



**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное бюджетное образовательное учреждение
высшего образования

«ИРКУТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ФГБОУ ВО «ИГУ»

Факультет бизнес-коммуникаций и информатики

Кафедра естественнонаучных дисциплин

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ

по дисциплине Б1.В.15 Шаблоны проектирования

направление подготовки 09.03.03 Прикладная информатика

направленность (профиль) Прикладная информатика (разработка программного обеспечения)

Одобен
УМК факультета бизнес-коммуникаций
и информатики

Разработан в соответствии с ФГОС ВО

с учетом требований проф. стандарта

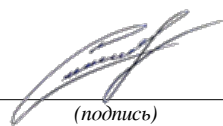
Председатель УМК

М.Г. Синчурина

ФИО, должность, ученая степень, звание

подпись, печать

Разработчики:



(подпись)

доцент

(занимаемая должность)

М.А. Сокольская

(инициалы, фамилия)

преподаватель

(занимаемая должность)

А.С. Воловиков

(инициалы, фамилия)



(подпись)

доцент

(занимаемая должность)

А.Г. Балахчи

(инициалы, фамилия)

Цель фонда оценочных средств. Оценочные средства предназначены для контроля и оценки образовательных достижений обучающихся, освоивших программу учебной дисциплины «Б1.В.15 Шаблоны проектирования». Перечень видов оценочных средств соответствует рабочей программе дисциплины.

Фонд оценочных средств включает контрольные материалы для проведения текущего контроля (в следующих формах: тест, практическое задание) и промежуточной аттестации в форме вопросов и заданий к зачету с оценкой.

Структура и содержание заданий – задания разработаны в соответствии с рабочей программой дисциплины «Б1.В.15 Шаблоны проектирования».

1. Паспорт фонда оценочных средств

Компетенция	Индикаторы компетенций	Результаты обучения
ПК-1 ППК-Р1 Способен разрабатывать и отлаживать программный код	ПК-1 ППК-Р1.1 Выполняет формализацию и алгоритмизация поставленных задач для разработки программного кода	• ПК-1 ППК-Р1.1. 3-1. Знает алгоритмы решения типичных задач, области и способы их применения • ПК-1 ППК-Р1.1. 3-2. Знает нотации и программное обеспечение для графического отображения алгоритмов • ПК-1 ППК-Р1.1. 3-3. Знает методы и приемы алгоритмизации поставленных задач • ПК-1 ППК-Р1.1. У-1. Умеет использовать методы и приемы формализации и алгоритмизации поставленных задач • ПК-1 ППК-Р1.1. У-2. Умеет применять алгоритмы решения типовых задач в соответствующих областях
	ПК-1 ППК-Р1.2 Формализует задачу ИТ отрасли в язык естественнонаучных дисциплин	• ПК-1 ППК-Р1.2. 3-1. Знает основные разделы математики, применяемые для анализа и моделирования непрерывных процессов и дискретных систем в прикладных задачах • ПК-1 ППК-Р1.2. У-1. Умеет выбирать адекватный математический аппарат для формализованного описания сущностей и отношений на основе ТЗ и бизнес-требований • ПК-1 ППК-Р1.2. В-1. Владеет навыками описания задач предметной области в виде формальных математических моделей, пригодных для последующей алгоритмизации и программной реализации
	ПК-1 ППК-Р1.3 Осуществляет обоснованный выбор методов и алгоритмов для программной реализации формальной математической модели	• ПК-1.3 ППК-Р1.5 3-1. Знает нормативно-технические документы (стандарты и регламенты), определяющие требования к оформлению программного кода • ПК-1.3 ППК-Р1.5 3-2. Знает основные стандарты оформления технической документации на компьютерное программное обеспечение • ПК-1.3 ППК-Р1.5 У-1. Умеет применять заданные стандарты и шаблоны для составления и оформления технической документации • ПК-1.3 ППК-Р1.5 У-2. Умеет применять нормативно-технические документы (стандарты и регламенты), определяющие требования к оформлению программного кода • ПК-1.3 ППК-Р1.5 У-3. Умеет применять инструментарий для создания и актуализации исходных текстов программ
	ПК-1 ППК-Р1.4 Разрабатывает программный код с использованием языков программирования	• ПК-1 ППК-Р1.4 3-1. Знает синтаксис выбранного языка программирования, особенности программирования на этом языке, стандартные библиотеки языка программирования • ПК-1 ППК-Р1.4 3-2. Знает методологии разработки компьютерного программного обеспечения • ПК-1 ППК-Р1.4 3-3. Знает технологии программирования • ПК-1 ППК-Р1.4 У-1. Умеет применять выбранные языки программирования для написания программного кода • ПК-1 ППК-Р1.4 У-2. Умеет использовать выбранную среду программирования • ПК-1 ППК-Р1.4 У-3. Умеет использовать возможности имеющейся технической и/или программной архитектуры для написания программного кода
	ПК-1 ППК-Р1.5 Оформляет программный код в соответствии с установленными требованиями	• ПК-1 ППК-Р1.5 3-1. Знает нормативно-технические документы (стандарты и регламенты), определяющие требования к оформлению программного кода • ПК-1 ППК-Р1.5 3-2. Знает основные стандарты оформления технической документации на компьютерное программное обеспечение • ПК-1 ППК-Р1.5 У-1. Умеет применять заданные стандарты и шаблоны для составления и оформления технической документации • ПК-1 ППК-Р1.5 У-2. Умеет применять нормативно-технические документы (стандарты и регламенты), определяющие требования к оформлению программного кода • ПК-1 ППК-Р1.5 У-3. Умеет применять инструментарий для создания и актуализации исходных текстов программ
	ПК-1 ППК-Р1.6 Работает с системой управления версиями программного кода	• ПК-1.4 ППК-Р1.6. 3-1. Знает возможности используемой системы управления версиями и вспомогательных инструментальных программных средств • ПК-1.4 ППК-Р1.6. 3-2. Знает установленный регламент использования системы управления версиями • ПК-1.4 ППК-Р1.6. У-1. Умеет регистрировать изменения исходного текста программного кода в системе управления версиями • ПК-1.4 ППК-Р1.6. У-2. Умеет сохранять изменения программного кода в соответствии с регламентом управления версиями • ПК-1.4 ППК-Р1.6. У-3. Умеет выполнять слияние, разделение и сравнение исходных текстов программного кода
	ПК-1 ППК-Р1.7 Проверяет и отлаживает программный код	• ПК-1.5 ППК-Р1.7 3-1. Знает методы и приемы отладки программного кода • ПК-1.5 ППК-Р1.7 3-2. Знает типы и форматы сообщений об ошибках, предупреждений • ПК-1.5 ППК-Р1.7 3-3. Знает способы использования технологических журналов, форматы и типы записей журналов • ПК-1.5 ППК-Р1.7 У-1. Умеет выявлять ошибки в программном коде • ПК-1.5 ППК-Р1.7 У-2. Умеет отлаживать программный код на уровне программных модулей • ПК-1.5 ППК-Р1.7 У-3. Умеет отлаживать программный код на уровне межмодульных взаимодействий и взаимодействий с окружением

Компетенция	Индикаторы компетенций	Результаты обучения
ПК-15 РГ-2 Способен организовать процессы разработки компьютерного программного обеспечения	ПК-15 РГ-2.1 Управляет проектированием компьютерного программного обеспечения	• ПК-15 РГ-2.1 З-1. Знает принципы построения архитектуры компьютерного программного обеспечения и виды архитектуры программного обеспечения • ПК-15 РГ-2.1 З-2. Знает методологии и средства проектирования компьютерного программного обеспечения • ПК-15 РГ-2.1 У-1. Умеет анализировать архитектуру компьютерного программного обеспечения и согласовывать ее с заинтересованными сторонами • ПК-15 РГ-2.1 У-2. Умеет распределять задания на проектирование компьютерного программного обеспечения, структуры базы данных, программных интерфейсов • ПК-15 РГ-2.1 У-2. Умеет оценивать качество проектирования компьютерного программного обеспечения, структуры базы данных, программных интерфейсов
	ПК-15 РГ-2.2 Управляет процессом разработки компьютерного программного обеспечения	• ПК-15 РГ-2.2 З-1. Знает методы и средства планирования и контроля (мониторинга) исполнения планов • ПК-15 РГ-2.2 З-2. Знает методы оценки качества плана разработки программного продукта (ресурсы, сроки, риски) • ПК-15 РГ-2.2 З-3. Знает основные принципы и методы управления персоналом • ПК-15 РГ-2.2 У-1. Умеет составлять планы процесса разработки программного продукта • ПК-15 РГ-2.2 У-2. Умеет оценивать качество плана разработки программного продукта (ресурсы, сроки, риски) • ПК-15 РГ-2.2 У-3. Умеет наблюдать за исполнением планов разработки программного продукта • ПК-15 РГ-2.2 У-4. Умеет корректировать план разработки программного продукта
	ПК-15 РГ-2.3 Управляет запросами на изменения, дефектами и проблемами в компьютерном программном обеспечении	• ПК-15 РГ-2.3 З-1. Знает методы и средства выявления дефектов, проблем и причин их возникновения в компьютерном программном обеспечении • ПК-15 РГ-2.3 З-2. Знает методы и средства управления запросами на изменения в компьютерном программном обеспечении • ПК-15 РГ-2.3 З-3. Знает методы планирования и документирования вносимых изменений в компьютерное программное обеспечение • ПК-15 РГ-2.3 У-1. Умеет оценивать запросы на изменения и предложенных решений по их осуществлению (по стоимости, трудоемкости, эффективности) в компьютерном программном обеспечении • ПК-15 РГ-2.3 У-2. Умеет принимать управленческие решения о реализации запросов на изменения (решений о необходимости и сроках внесения изменений в программное обеспечение и документацию) • ПК-15 РГ-2.3 У-3. Умеет контролировать исполнение принятых управленческих решений
	ПК-15 РГ-2.4 Управляет конфигурациями и выпусками программного продукта	• ПК-15 РГ-2.4 З-1. Знает нормативно-технические документы (стандарты и регламенты) по процессам управления конфигурациями, изменениями и выпусками программного продукта • ПК-15 РГ-2.4 З-2. Знает состав и методы использования коллективной среды разработки компьютерного программного обеспечения и системы контроля версий • ПК-15 РГ-2.4 З-3. Знает методы и средства верификации работоспособности выпусков программных продуктов • ПК-15 РГ-2.4 У-1. Умеет управлять версиями отдельных компонентов и программного продукта в целом • ПК-15.4 РГ-2.4 У-2. Умеет анализировать требования к выпуску новой версии программного продукта • ПК-15 РГ-2.4 У-3. Умеет определять перечень функциональных требований, реализуемых в новой версии программного продукта • ПК-15 РГ-2.4 У-4. Умеет контролировать выполнение разработки версии программного продукта

Компетенция	Индикаторы компетенций	Результаты обучения
ПК-2 ППК-Р2 Способен проверять работоспособность и проводить рефакторинг кода программного обеспечения	ПК-2 ППК-Р2.1 Разрабатывает тестовые наборы данных для проверки работоспособности компьютерного программного обеспечения	• ПК-2 ППК-Р2.1 З-1. Знает методы создания и документирования контрольных примеров и тестовых наборов данных • ПК-2 ППК-Р2.1 З-2. Знает требования к структуре и форматам хранения тестовых наборов данных • ПК-2 ППК-Р2.1 З-3. Знает правила, алгоритмы и технологии создания тестовых наборов данных • ПК-2 ППК-Р2.1 У-1. Умеет разрабатывать и оформлять контрольные примеры для проверки работоспособности компьютерного программного обеспечения • ПК-2 ППК-Р2.1 У-2. Умеет готовить тестовые наборы данных в соответствии с выбранной методикой тестирования компьютерного программного обеспечения."
	ПК-2 ППК-Р2.2 Проверяет работоспособность компьютерного программного обеспечения	• ПК-2 ППК-Р2.2 З-1. Знает методы и средства проверки работоспособности компьютерного программного обеспечения • ПК-2 ППК-Р2.2 З-2. Знает государственные стандарты испытания автоматизированных систем • ПК-2 ППК-Р2.2 З-3. Знает руководящие документы по стандартизации требований к документам автоматизированных систем • ПК-2 ППК-Р2.2 У-1. Умеет применять методы и средства проверки работоспособности компьютерного программного обеспечения • ПК-2 ППК-Р2.2 У-2. Умеет интерпретировать диагностические данные проверки работоспособности компьютерного программного обеспечения • ПК-2 ППК-Р2.2 У-3. Умеет анализировать значения полученных характеристик компьютерного программного обеспечения
	ПК-2 ППК-Р2.3 Исправляет дефекты программного кода, зафиксированных в базе данных дефектов	• ПК-2 ППК-Р2.3 З-1. Знает типичные ошибки, возникающие при разработке компьютерного программного обеспечения, методы их диагностики и исправления • ПК-2 ППК-Р2.3 З-2. Знает методы и приемы отладки программного кода • ПК-2 ППК-Р2.3 У-1. Умеет воспроизводить дефекты программного кода, зафиксированные в базе данных дефектов • ПК-2 ППК-Р2.3 У-2. Умеет выяснять причины возникновения дефектов программного кода • ПК-2 ППК-Р2.3 У-3. Умеет вносить изменений в программный код для устранения выявленных дефектов
	ПК-2 ППК-Р2.4 Выполняет рефакторинг и инспекцию программного кода (код-ревью)	• ПК-2 ППК-Р2.4 З-1. Знает методы и средства рефакторинга и инспекции программного кода • ПК-2 ППК-Р2.4 З-2. Знает нормативно-технические документы (стандарты и регламенты), регламентирующие требования к программному коду, порядок отражения изменений в системе управления версиями, порядок отражения результатов рефакторинга, оптимизации и инспекции в коллективной базе знаний • ПК-2 ППК-Р2.4 У-1. Умеет анализировать программный код на соответствие требованиям по читаемости и производительности • ПК-2 ППК-Р2.4 У-2. Умеет проводить инспекцию программного кода для поиска не обнаруженных на ранних стадиях разработки компьютерного программного обеспечения ошибок и критических мест • ПК-2 ППК-Р2.4 У-3. Умеет применять методы и средства рефакторинга и инспекции программного кода • ПК-2 ППК-Р2.4 У-4. Умеет публиковать результаты рефакторинга и инспекции в коллективной базе знаний
ПК-4 ППК-Р4 Способен проектировать и интегрировать базы данных в программные модули и компоненты	ПК-4 ППК-Р4.1 Пишет программный код с использованием языков определения и манипулирования данными в базах данных	• ПК-4 ППК-Р4.1 З-1. Знает архитектуры современных систем управления баз данных, включая SQL и noSQL • ПК-4 ППК-Р4.1 З-2. Знает синтаксис языка работы с выбранной базой данных, особенности программирования на этом языке • ПК-4 ППК-Р4.1 З-3. Знает современные среды программирования для работы с базами данных • ПК-4 ППК-Р4.1 У-1. Умеет применять выбранные языки работы с базами данных • ПК-4 ППК-Р4.1 У-2. Умеет использовать выбранную среду программирования для работы с данными в базе • ПК-4 ППК-Р4.1 У-3. Умеет использовать методы и средства выбранного языка программирования для работы с базами данных
	ПК-4 ППК-Р4.2 Проектирует базы данных для программных модулей и компонентов	• ПК-4 ППК-Р4.2 З-1. Знает современные подходы к проектированию реляционных и нереляционных баз данных • ПК-4 ППК-Р4.2 З-2. Знает нормативно-технические документы (стандарты и регламенты), определяющие требования к оформлению моделей баз данных выбранной архитектуры • ПК-4 ППК-Р4.2 У-1. Умеет выбирать тип базы данных в зависимости от решаемой задачи • ПК-4 ППК-Р4.2 У-2. Умеет проектировать и актуализировать структуру базы данных для программных моделей и компонентов • ПК-4 ППК-Р4.2 У-3. Умеет применять нормативно-технические документы (стандарты и регламенты), определяющие требования к оформлению модели баз данных • ПК-4 ППК-Р4.2 У-4. Умеет применять инструментарий для создания и актуализации моделей баз данных
	ПК-4 ППК-Р4.3 Оптимизирует производительность работы с базами данных	• ПК-4 ППК-Р4.3 З-1. Знает внутреннее устройство СУБД выбранной архитектуры • ПК-4 ППК-Р4.3 З-2. Знает методы и средства мониторинга и оптимизации производительности СУБД выбранной архитектуры • ПК-4 ППК-Р4.3 У-1. Умеет применять методы и средства мониторинга производительности запросов к базе данных • ПК-4 ППК-Р4.3 У-2. Умеет вырабатывать варианты оптимизации производительности запросов в базе данных • ПК-4 ППК-Р4.3 У-3. Умеет проводить оценку и обоснование рекомендуемых решений по оптимизации производительности запросов в базе данных

Компетенция	Индикаторы компетенций	Результаты обучения
ПК-5 ППК-Р5 Способен разрабатывать требования и проектировать программное обеспечение	ПК-5 ППК-Р5.1 Анализирует возможности реализации требований к компьютерному программному обеспечению	• ПК-5 ППК-Р5.1 З-1. Знает возможности существующей программно-технической архитектуры • ПК-5 ППК-Р5.1 З-2. Знает возможности современных и перспективных средств разработки программных продуктов, технических средств • ПК-5 ППК-Р5.1 З-3. Знает методологии разработки компьютерного программного обеспечения и технологии программирования • ПК-5 ППК-Р5.1 У-1. Умеет проводить сбор и систематизацию требований к компьютерному программному обеспечению • ПК-5 ППК-Р5.1 У-2. Умеет выявлять взаимосвязи и документировать требования к компьютерному программному обеспечению • ПК-5 ППК-Р5.1 У-3. Умеет вырабатывать варианты реализации требований к компьютерному программному обеспечению
	ПК-5 ППК-Р5.2 Разрабатывает технические спецификации на программные компоненты и их взаимодействие	• ПК-5 ППК-Р5.2 З-1. Знает методы и средства проектирования компьютерного программного обеспечения • ПК-5 ППК-Р5.2 З-2. Знает методы и средства проектирования программных интерфейсов • ПК-5 ППК-Р5.2 У-1. Умеет выбирать средства реализации требований к компьютерному программному обеспечению • ПК-5 ППК-Р5.2 У-2. Умеет вырабатывать варианты реализации компьютерного программного обеспечения • ПК-5 ППК-Р5.2 У-3. Умеет проводить оценку и обоснование рекомендуемых решений
	ПК-5 ППК-Р5.3 Проектирует компьютерное программное обеспечение	• ПК-5 ППК-Р5.3 З-1. Знает принципы построения и виды архитектуры компьютерного программного обеспечения • ПК-5 ППК-Р5.3 З-2. Знает типовые решения, библиотеки программных модулей, шаблоны, классы объектов, используемые при разработке компьютерного программного обеспечения • ПК-5 ППК-Р5.3 З-3. Знает нормативно-технические документы (стандарты), определяющие требования к технической документации на компьютерное программное обеспечение • ПК-5 ППК-Р5.3 З-4. Знает методы и средства проектирования компьютерного программного обеспечения • ПК-5 ППК-Р5.3 У-1. Умеет разрабатывать и изменять архитектуру компьютерного программного обеспечения и согласовывать ее с системным аналитиком и архитектором программного обеспечения • ПК-5 ППК-Р5.3 У-2. Умеет проектировать структуры данных • ПК-5 ППК-Р5.3 У-3. Умеет проектировать программные интерфейсы
	ПК-5 ППК-Р5.4 Выполняет логическое проектирование системы	• ПК-5 ППК-Р5.4 З-1. Знает устройство и функционирование ИТ-систем/продуктов • ПК-5 ППК-Р5.4 З-2. Знает методы моделирования и описания устройства и функционирования ИТ-систем/продуктов, их частей, обеспечения и окружения • ПК-5 ППК-Р5.4 З-3. Знает базовые технологии взаимодействия и интеграции систем и компонентов • ПК-5 ППК-Р5.4 З-4. Знает методы функциональной декомпозиции ИТ-систем • ПК-5 ППК-Р5.4 У-1. Умеет декомпонизировать ИТ-системы и ИТ-продукты на подсистемы и элементы поставки • ПК-5 ППК-Р5.4 У-2. Умеет описывать интерфейсы пользователя на логическом уровне • ПК-5 ППК-Р5.4 У-3. Умеет описывать интеграции со смежными системами на логическом уровне
ПК-6 ИП-2 Способен оптимизировать производительность программного обеспечения	ПК-6 ИП-2.1 Выполняет мониторинг производительности программного обеспечения	• ПК-6 ИП-2.1 З-1. Знает методы и средства мониторинга производительности компьютерного программного обеспечения • ПК-6 ИП-2.1 З-2. Знает метрики производительности программного обеспечения • ПК-6 ИП-2.1 З-3. Знает современные инструменты мониторинга производительности программного обеспечения • ПК-6 ИП-2.1 У-1. Умеет применять методы и средства мониторинга производительности компьютерного программного обеспечения • ПК-6 ИП-2.1 У-2. Умеет интерпретировать диагностические данные мониторинга производительности компьютерного программного обеспечения • ПК-6 ИП-2.1 У-3. Умеет определять "узкие места" программного обеспечения
	ПК-6 ИП-2.2 Выполняет оптимизацию программного кода	• ПК-6 ИП-2.2 З-1. Знает методы и средства оптимизации производительности компьютерного программного обеспечения • ПК-6 ИП-2.2 З-2. Знает современные инструменты оптимизации производительности программного обеспечения • ПК-6 ИП-2.2 У-1. Умеет оптимизировать программный код с использованием специализированных программных средств • ПК-6 ИП-2.2 У-2. Умеет вырабатывать варианты оптимизации производительности компьютерного программного обеспечения • ПК-6 ИП-2.2 У-3. Умеет проводить оценку и обоснование рекомендуемых решений по оптимизации производительности

2. Показатели и критерии оценивания компетенций, шкалы оценивания

2.1. Показатели и критерии оценивания компетенций

№ п/п	Раздел, тема	Код индикатора компетенции	Наименование ОС	
			ТК	ПА
1	Работа с чужим кодом	ПК-15 РГ-2.1, ПК-15 РГ-2.2, ПК-15 РГ-2.3, ПК-2 ППК-Р2.4, ПК-5 ППК-Р5.1, ПК-1 ППК-Р1.4, ПК-2 ППК-Р2.2, ПК-6 ИП-2.1, ПК-6 ИП-2.2, ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.5, ПК-1 ППК-Р1.6, ПК-1 ППК-Р1.7, ПК-15 РГ-2.4, ПК-4 ППК-Р4.1	Тест, Пз	Тест
2	Организация проекта	ПК-1 ППК-Р1.1, ПК-5 ППК-Р5.1, ПК-15 РГ-2.1, ПК-15 РГ-2.2, ПК-5 ППК-Р5.3, ПК-5 ППК-Р5.4, ПК-1 ППК-Р1.4, ПК-5 ППК-Р5.2	Тест, Пз	Тест
3	Одиночка (Singleton)	ПК-15 РГ-2.1, ПК-15 РГ-2.2, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.3, ПК-5 ППК-Р5.4, ПК-5 ППК-Р5.2, ПК-1 ППК-Р1.4, ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-2 ППК-Р2.4, ПК-6 ИП-2.1	Тест, Пз	Тест
4	Абстрактные классы	ПК-1 ППК-Р1.4, ПК-15 РГ-2.1, ПК-5 ППК-Р5.4, ПК-1 ППК-Р1.5, ПК-1 ППК-Р1.6, ПК-1 ППК-Р1.7, ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-2 ППК-Р2.4, ПК-5 ППК-Р5.3, ПК-6 ИП-2.1, ПК-6 ИП-2.2	Тест, Пз	Тест
5	Фабричный метод (Factory Method)	ПК-15 РГ-2.1, ПК-15 РГ-2.2, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.4, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-1 ППК-Р1.6, ПК-1 ППК-Р1.7, ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.4, ПК-6 ИП-2.2	Тест, Пз	Тест
6	Доменные модели	ПК-15 РГ-2.1, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.2, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-1 ППК-Р1.6, ПК-1 ППК-Р1.7, ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2	Тест, Пз	Тест
7	Фабрика (Factory)	ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.3, ПК-5 ППК-Р5.4, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-1 ППК-Р1.6, ПК-1 ППК-Р1.7, ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-2 ППК-Р2.4	Тест, Пз	Тест
8	Сериализация данных	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-4 ППК-Р4.2, ПК-5 ППК-Р5.4, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-1 ППК-Р1.6, ПК-1 ППК-Р1.7, ПК-2 ППК-Р2.4, ПК-6 ИП-2.1, ПК-6 ИП-2.2	Тест, Пз	Тест

№ п/п	Раздел, тема	Код индикатора компетенции	Наименование ОС	
			ТК	ПА
9	Десериализация данных	ПК-4 ППК-Р4.2, ПК-4 ППК-Р4.3, ПК-5 ППК-Р5.2, ПК-4 ППК-Р4.1	Тест, Пз	Тест
10	Складские обороты	ПК-4 ППК-Р4.2, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.2, ПК-4 ППК-Р4.1, ПК-4 ППК-Р4.3, ПК-6 ИП-2.1, ПК-6 ИП-2.2	Тест, Пз	Тест
11	Прототип (Prototype)	ПК-4 ППК-Р4.2, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-1 ППК-Р1.6, ПК-1 ППК-Р1.7, ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-2 ППК-Р2.4, ПК-4 ППК-Р4.1, ПК-4 ППК-Р4.3	Тест, Пз	Тест
12	RestApi (Dto, Swagger)	ПК-4 ППК-Р4.2, ПК-4 ППК-Р4.3, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-1 ППК-Р1.6, ПК-1 ППК-Р1.7, ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-2 ППК-Р2.4, ПК-4 ППК-Р4.1, ПК-6 ИП-2.1, ПК-6 ИП-2.2	Тест, Пз	Тест
13	Наблюдатель (Observer)	ПК-4 ППК-Р4.3, ПК-5 ППК-Р5.4, ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-1 ППК-Р1.6, ПК-1 ППК-Р1.7, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-2 ППК-Р2.4, ПК-4 ППК-Р4.1, ПК-6 ИП-2.1	Тест, Пз	Тест
14	Логирование	ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.4, ПК-15 РГ-2.1, ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-2 ППК-Р2.4, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.2, ПК-5 ППК-Р5.3, ПК-6 ИП-2.1, ПК-6 ИП-2.2	Тест, Пз	Тест
15	Развертывание приложения	ПК-15 РГ-2.4	Тест, Пз	Тест
16	Разделение ответственности (CQRS)	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-4 ППК-Р4.2, ПК-4 ППК-Р4.3, ПК-5 ППК-Р5.4, ПК-15 РГ-2.4	Тест, Пз	Тест

2.2. Критерии оценивания результатов обучения для текущего контроля успеваемости и промежуточной аттестации

Оценочное средство	Критерии оценивания	Шкала оценивания
Тест	Студентом даны правильные ответы на 91-100% заданий	Отлично
	Студентом даны правильные ответы на 81-90% заданий	Хорошо
	Студентом даны правильные ответы на 71-80% заданий	Удовлетворительно
	Студентом даны правильные ответы менее чем на 70% заданий	Неудовлетворительно
Практическое задание	Задание выполнено верно. Выбран оптимальный путь решения. Присутствует развернутое описание алгоритма решения	Отлично
	Задание выполнено верно. Допущены негрубые логические ошибки при описании алгоритма решения. Отсутствуют пояснения к решению задания	Хорошо
	Ход решения задания верный, но допущены ошибки приведшие к неправильному ответу	Удовлетворительно
	В работе получен неверный ответ, связанный с грубыми ошибками допущенными в ходе решения, либо решение отсутствует полностью	Неудовлетворительно

2.3. Оценочные средства для текущего контроля (примеры)

2.3.1. Материалы для компьютерного тестирования обучающихся

Общие критерии оценивания

Процент правильных ответов	Оценка
91% – 100%	5 (отлично)
81% – 90%	4 (хорошо)
71% – 80%	3 (удовлетворительно)
Менее 70%	2 (неудовлетворительно)

Соответствие вопросов теста индикаторам формируемых и оцениваемых компетенций

№ вопроса в тесте	Код индикатора компетенции
1	ПК-15 РГ-2.1, ПК-15 РГ-2.2, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.3, ПК-5 ППК-Р5.4
2	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3
3	ПК-4 ППК-Р4.3, ПК-5 ППК-Р5.4
4	ПК-15 РГ-2.1, ПК-15 РГ-2.2, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.3, ПК-5 ППК-Р5.4
5	ПК-15 РГ-2.1, ПК-15 РГ-2.2, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.4

№ вопроса в тесте	Код индикатора компетенции
6	ПК-4 ППК-Р4.2, ПК-4 ППК-Р4.3, ПК-5 ППК-Р5.4
7	ПК-4 ППК-Р4.2, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.2
8	ПК-15 РГ-2.1, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.2
9	ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.2, ПК-5 ППК-Р5.3, ПК-5 ППК-Р5.4
10	ПК-1 ППК-Р1.1, ПК-5 ППК-Р5.1
11	ПК-4 ППК-Р4.2, ПК-5 ППК-Р5.4
12	ПК-15 РГ-2.4
13	ПК-1 ППК-Р1.4, ПК-15 РГ-2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.4, ПК-6 ИП-2.1, ПК-6 ИП-2.2
14	ПК-4 ППК-Р4.2
15	ПК-4 ППК-Р4.2, ПК-4 ППК-Р4.3
16	ПК-1 ППК-Р1.4, ПК-15 РГ-2.1, ПК-5 ППК-Р5.4
17	ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.3, ПК-5 ППК-Р5.4
18	ПК-15 РГ-2.1, ПК-15 РГ-2.2, ПК-15 РГ-2.3, ПК-2 ППК-Р2.4, ПК-5 ППК-Р5.1
19	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3
20	ПК-4 ППК-Р4.2, ПК-4 ППК-Р4.3, ПК-5 ППК-Р5.2
21	ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.4

Ключ ответов

№ вопроса в тесте	Номер ответа (или ответ, или соответствие)
1	d
2	b
3	c
4	a
5	d
6	c
7	b
8	b
9	c
10	a
11	d
12	b
13	a
14	a
15	a
16	c
17	c

№ вопроса в тесте	Номер ответа (или ответ, или соответствие)
18	b
19	b
20	d
21	b

Перечень тестовых вопросов

№ 1. Задание с единичным выбором. Выберите один правильный ответ.

Какова основная идея подхода Domain-Driven Design (DDD)?

- a. Использование исключительно микросервисной архитектуры.
- b. Разделение приложения на клиентскую и серверную части.
- c. Максимальное сокращение количества классов в проекте.
- d. Проектирование программного обеспечения на основе глубокого анализа предметной области и построения модели, отражающей бизнес-процессы.

№ 2. Задание с единичным выбором. Выберите один правильный ответ.

Для чего используется шаблон проектирования "Общие данные"?

- a. Это распространенный вариант объединения различных систем
- b. Данный шаблон является простым вариантом интеграции различных система на основе взаимодействия базы данных

№ 3. Задание с единичным выбором. Выберите один правильный ответ.

В системе автоматизации сети ресторанов «Ромашка» после изменения статуса заказа на «Готов» необходимо автоматически: уведомить курьерскую службу; обновить информацию на рабочем месте администратора; изменить статус заказа на информационном табло кухни. Какое архитектурное решение наиболее соответствует применению шаблона «Наблюдатель» (Observer)?

- a. После изменения статуса необходимо вручную обновить данные во всех подсистемах.
- b. Каждый модуль самостоятельно периодически проверяет, изменился ли статус заказа.
- c. Объект «Заказ» автоматически уведомляет все подписанные на него модули при изменении своего состояния.
- d. Все действия выполняются одним большим методом, содержащим последовательные вызовы всех подсистем.

№ 4. Задание с единичным выбором. Выберите один правильный ответ.

Какова основная цель применения шаблона проектирования Singleton (Одиночка)? а) Создание большого количества экземпляров одного класса. б) Обеспечение существования только одного экземпляра класса и предоставление глобальной точки доступа к нему. в) Автоматическое создание объектов различных типов. г) Разделение приложения на клиентскую и серверную части.

- a. Обеспечение существования только одного экземпляра класса и предоставление глобальной точки доступа к нему.
- b. Создание большого количества экземпляров одного класса.
- c. Разделение приложения на клиентскую и серверную части.

d. Автоматическое создание объектов различных типов.

№ 5. Задание с единичным выбором. Выберите один правильный ответ.

В системе автоматизации сети ресторанов «Ромашка» необходимо реализовать создание документов различных типов: заказ поставщику, заявка на производство, маршрутный лист курьера и отчет по продажам. При этом в дальнейшем список документов может расширяться без изменения существующего кода. Какое решение наиболее соответствует применению шаблона «Фабричный метод» (Factory Method)?

a. Создавать документы непосредственно с помощью оператора new во всех местах программы.

b. Создать один экземпляр документа и изменять его тип во время работы программы.

c. Создать один класс, который с помощью большого оператора if-else или switch создает все виды документов.

d. Для каждого типа документа реализовать отдельный фабричный метод, который создает соответствующий объект, что позволит легко добавлять новые типы документов без изменения существующей логики.

№ 6. Задание с единичным выбором. Выберите один правильный ответ.

В системе автоматизации сети ресторанов «Ромашка» официанты и кассиры постоянно создают и изменяют заказы, а руководители ресторанов одновременно просматривают отчеты по продажам, остаткам на складе и загруженности курьеров. Для повышения производительности было принято решение разделить обработку операций изменения данных и операций их чтения. Какой архитектурный подход наиболее соответствует данному решению?

a. Каждая подсистема самостоятельно хранит собственную копию всей информации.

b. Все данные периодически копируются в отдельную базу данных вручную.

c. Для операций изменения данных (команд) и операций чтения данных (запросов) используются отдельные модели и механизмы обработки.

d. Все операции чтения и изменения данных выполняются одним сервисом и используют одну и ту же модель данных.

№ 7. Задание с единичным выбором. Выберите один правильный ответ.

В системе автоматизации сети ресторанов «Ромашка» после каждой операции поступления, списания или перемещения продукции необходимо автоматически обновлять остатки товаров на складе. Какое техническое решение наиболее корректно обеспечивает учет складских оборотов?

a. Хранить только текущие остатки товаров без информации об операциях.

b. После каждой операции создавать запись о складском обороте и на ее основе автоматически пересчитывать текущие остатки продукции.

c. Изменять остатки товаров только один раз в конце рабочего дня.

d. Пересчитывать остатки вручную администратором ресторана.

№ 8. Задание с единичным выбором. Выберите один правильный ответ.

При разработке системы автоматизации сети ресторанов «Ромашка» необходимо спроектировать доменную модель. Какой из перечисленных вариантов наиболее корректно отражает понятие доменной модели? а) Совокупность экранных форм пользовательского интерфейса, используемых администраторами ресторанов. б) Модель, описывающая ключевые бизнес-сущности (например, Ресторан, Заказ, Склад, Поставка, Курьер,

Сотрудник) и правила их взаимодействия в предметной области. в) Набор SQL-запросов для работы с базой данных. г) Структура каталогов и файлов программного проекта.

- а. Набор SQL-запросов для работы с базой данных.
- б. Модель, описывающая ключевые бизнес-сущности (например, Ресторан, Заказ, Склад, Поставка, Курьер, Сотрудник) и правила их взаимодействия в предметной области.
- с. Структура каталогов и файлов программного проекта.
- д. Совокупность экранных форм пользовательского интерфейса, используемых администраторами ресторанов.

№ 9. Задание с единственным выбором. Выберите один правильный ответ.

Какой из перечисленных объектов наиболее целесообразно реализовать с использованием шаблона Singleton?

- а. Запись о товаре на складе.
- б. Объект, представляющий отдельного пользователя системы.
- с. Менеджер конфигурации приложения, обеспечивающий доступ к настройкам программы.
- д. Экземпляр заказа интернет-магазина.

№ 10. Задание с единственным выбором. Выберите один правильный ответ.

Какой из принципов SOLID утверждает, что класс должен иметь только одну причину для изменения?

- а. Single Responsibility Principle (SRP)
- б. Open/Closed Principle (OCP)
- с. Interface Segregation Principle (ISP)
- д. Dependency Inversion Principle (DIP)

№ 11. Задание с единственным выбором. Выберите один правильный ответ.

При разработке системы автоматизации сети ресторанов «Ромашка» необходимо передать информацию о заказе из ресторана в центральную систему обработки. Для этого объект «Заказ» преобразуется в формат JSON и отправляется по сети. Какова основная цель сериализации данных в данной ситуации?

- а. Оптимизация производительности процессора при обработке заказа.
- б. Шифрование данных перед передачей по сети
- с. Создание резервной копии базы данных
- д. Преобразование объекта в формат, пригодный для хранения или передачи с возможностью последующего восстановления объекта.

№ 12. Задание с единственным выбором. Выберите один правильный ответ.

Команда завершила разработку системы автоматизации сети ресторанов «Ромашка». Перед передачей программного обеспечения заказчику необходимо обеспечить его запуск на сервере ресторана с сохранением всех настроек и возможностью дальнейшего обновления. Что включает в себя процесс развертывания программного приложения?

- а. Только компиляцию исходного кода.
- б. Установку и настройку программного обеспечения в целевой среде, включая конфигурацию, подключение необходимых компонентов и подготовку приложения к эксплуатации.
- с. Исправление ошибок, обнаруженных в процессе разработки.

d. Создание резервной копии исходного кода.

№ 13. Задание с единственным выбором. Выберите один правильный ответ.

Почему внесение изменений в чужой программный код часто оказывается сложнее разработки новой программы?

a. Неизвестны структура программы, логика реализации и взаимосвязи между ее компонентами, поэтому любое изменение может повлиять на уже работающий функционал.

b. Все ответы верны.

c. Чужой код всегда содержит ошибки.

d. Чужой код невозможно изменить без полного переписывания.

№ 14. Задание с единственным выбором. Выберите один правильный ответ.

В системе автоматизации сети ресторанов «Ромашка» администратор ежедневно создает меню для каждого ресторана. В большинстве случаев новое меню отличается от меню предыдущего дня лишь несколькими позициями и ценами. Какое решение наиболее соответствует применению шаблона «Прототип» (Prototype)? а) Каждый раз создавать новое меню вручную, заполняя все блюда заново. б) Создать копию существующего меню и изменить только те позиции, которые отличаются от предыдущего дня. в) Хранить только одно меню для всех ресторанов. г) Удалять старое меню перед созданием нового.

a. Создать копию существующего меню и изменить только те позиции, которые отличаются от предыдущего дня.

b. Каждый раз создавать новое меню вручную, заполняя все блюда заново.

c. Удалять старое меню перед созданием нового.

d. Хранить только одно меню для всех ресторанов.

№ 15. Задание с единственным выбором. Выберите один правильный ответ.

Можно ли использовать несколько Декораторов для одного объекта?

a. Да

b. Нет

№ 16. Задание с единственным выбором. Выберите один правильный ответ.

Каково основное назначение абстрактного класса в объектно-ориентированном программировании?

a. Создание объекта, экземпляры которого используются непосредственно в программе.

b. Хранение только статических методов и констант.

c. Определение общего интерфейса и базовой реализации для группы связанных классов с последующим наследованием.

d. Обеспечение существования единственного экземпляра класса.

№ 17. Задание с единственным выбором. Выберите один правильный ответ.

При разработке системы автоматизации сети ресторанов «Ромашка» необходимо создавать различные типы отчетов: финансовый отчет, отчет по складу, отчет по продажам и отчет по работе курьерской службы. В дальнейшем планируется добавление новых видов отчетов без изменения существующего кода. Какое решение наиболее соответствует использованию шаблона «Фабрика»?

a. Создавать каждый отчет непосредственно с помощью оператора new в разных частях программы.

b. Использовать один объект отчета и изменять его тип во время выполнения программы.

c. Реализовать класс-фабрику, который отвечает за создание объектов отчетов, а клиентский код работает только с их общим интерфейсом.

d. Создавать все отчеты в одном большом классе с использованием множества условных операторов.

№ 18. Задание с единичным выбором. Выберите один правильный ответ.

Какова основная цель анализа существующего программного кода перед внесением изменений?

a. Удалить неиспользуемые файлы проекта

b. Определить структуру программы, понять взаимосвязи между компонентами и оценить влияние изменений на существующий функционал.

c. Полностью переписать программу с использованием нового языка программирования.

d. Увеличить количество строк программного кода.

№ 19. Задание с единичным выбором. Выберите один правильный ответ.

Что такое CQRS?

a. Это высокоскоростная система передачи данных

b. Это архитектурный шаблон проектирования с разделением ответственности за команды и запросы

№ 20. Задание с единичным выбором. Выберите один правильный ответ.

В системе автоматизации сети ресторанов «Ромашка» центральный сервер получает от ресторана данные о новом заказе в формате JSON. Для дальнейшей обработки необходимо восстановить объект «Заказ» в программе. Какой процесс выполняется в данном случае?

a. Архивация данных.

b. Репликация данных.

c. Компиляция данных.

d. Десериализация данных.

№ 21. Задание с единичным выбором. Выберите один правильный ответ.

В системе автоматизации сети ресторанов «Ромашка» используется собственный модуль логирования, однако было принято решение перейти на стороннюю библиотеку, интерфейс которой отличается от существующего. При этом изменение кода всех модулей системы нежелательно. Какое решение наиболее соответствует применению шаблона «Адаптер» (Adapter)?

a. Отказаться от использования сторонней библиотеки и оставить старую реализацию.

b. Создать класс-адаптер, который преобразует вызовы существующего интерфейса логирования в интерфейс новой библиотеки.

c. Одновременно использовать две независимые системы логирования без их взаимодействия.

d. Полностью переписать все модули системы для работы с новой библиотекой логирования.

2.3.2. Практические задания для оценки компетенции «ПК-1 ППК-Р1.4»

№ 1. Кейс №1. Работа с чужим кодом: модернизация приложения «Календарь» (сам.

работа).

Кейс №1. Работа с чужим кодом: модернизация приложения «Календарь» (сам. работа)

В указанном примере (календарь) заменить название месяцев / дней недели на русский вариант

Добавить функционал:

- Добавить возможность формирования календаря по произвольному периоду. Сейчас строится от текущего месяца и года.

- Добавить отображение праздничных дней в календаре. Фон: **светло коричневый**. Цвет текста: **темно серый**.

- Изменить отображение календаря. Начало недели с **понедельника**. Окончание: **суббота, воскресенье**.

- Обязательно зафиксировать время, затраченное на изменение (в часах).

№ 2. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" Практическое занятие 1. Организация проекта.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка"
Практическое занятие 1. Организация проекта
Самостоятельная работа

- Необходимо создать собственный Git репозиторий.
 - В репозиторий добавить файл readme.md в формате Markdown
- Можно использовать готовый online редакторы: <https://dillinger.io/>
- Добавить каталог Docs и в него включить приложенный файл Technicaltask.md
 - В качестве результата, прислать ссылку на Git репо
 - Придумать и написать собственные правила оформления исходного кода и структуры проекта на основе [Шаблона](#) .

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка зада

№ 3. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 2. "Шаблон Одиночка (Sigleton) " .

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .
Практическое занятие № 2. "Шаблон Одиночка (Sigleton) "

Необходимо;

1. Доработать класс **settings_manager**. Добавить / изменить метод convert. Данный метод должен формировать объект от класса **Settings.py** из загруженного словаря.
2. Добавить модульный тест, который проверяет загрузку данных в объект от класса **Settings.py**. Проверить загрузку всех свойств.
3. Доработать класс **settings_manager**. Сделать универсальную загрузку из файла, который может располагаться в любом каталоге.
4. Добавить модульный тест который проверит загрузку настроек из файла в другом каталоге и другого наименования.
5. Доработать класс **Settings.py** добавить следующие ограничения:
ИНН : 12 симв

Счет 11 симв

Корреспондентский счет 11 симв

БИК 9 симв

Наименование

Вид собственности 5 симв

6. Список файлов из занятия от 07/02/2024 в объявлении

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка зада

№ 4. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 3. Абстрактные классы.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 3. Абстрактные классы.

Самостоятельная работа

Необходимо доработать текущий проект. Добавить следующее:

1 Добавить модели данных:**номенклатура, единица измерения, группа номенклатура, организация, склад**

2 Все модели должны располагаться в каталоге `src/models[name]_model`

3 Все модели должны наследоваться от класса `abstract_reference`

4 Модель единица измерения имеет два поля; базовая единица измерения, коэффициент пересчета.

Например для единицы измерения 1 Кг - базовая единица измерения будет - 1 гр с коэффициентом пересчета = 1. Для 1 кг - коэффициент пересчет будет 1000. Для единицы измерения 1 грамм - базовая единица будет 1 грамм и коэффициент = 1

Пример:

```
base_range = range_model("грамм",1)
new_range = range_model("кг", 1000, base_range)
```

5 Модель организация должна содержать в себе:**ИНН, БИК, Счет, Форма собственности.**

6 В Модель организации добавить в конструктор класс `settings.py` и организовать копирование данных полей

Пример:

```
manager = settings_manager()
manager.open("settings.json")
company = company_model(manager.settings)
```

7 Необходимо добавить автотесты которые бы показывали корректное создание всех моделей со всеми вариантами параметров.

8 Модель номенклатура должна содержать дополнительное поле: полное наименование с длиной 255 символов.

9. Модель номенклатуры должна включать в себя другие модели; группа номенклатуры, единица измерения

10 Обычное наименование (класс `abstract_reference`) имеет ограничение на поле наименование - 50 символов.

11 Добавить автотест который продемонстрирует загрузку параметров настроек и далее

создание объекта по модели организация

12 Добавить автотест который продемонстрирует работу с моделью единица измерения

13 Все обработки ошибок должны быть реализованы через внутренние классы: `argument_exception`, `operation_exception`, `error_proxy`

Задание сдается каждым учащимся в собственной репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания.

№ 5. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 3. Фабричный метод (Factory Method).

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 4. Фабричный метод (Factory Method)

Самостоятельная работа

- Доработать метод **create** класса **start_service** Метод должен формировать и сохранять данные по:

- Номенклатуре
- Единицам измерения
- Группам

- Создать собственный рецепт в формате **Markdown**. Включить в каталог **Docs** своего проекта.

- Необходимо создать **модели** для описания рецептов

- Доработать класс **start_service / create_receipts** (формирование рецептов) и доработать метод **create** для генерации данных. Добавить формирование моделей (собственный рецепт, рецепт на занятии).

- Доработать класс **storage_repository** Добавить структуру для хранения / описания рецептов.

- Добавить модульные тесты для проверки наличия данных

- Номенклатуре
- Единицам измерения
- Группам
- Рецепты

Задание сдается каждым учащимся в собственной репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания.

№ 6. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 5. Доменные модели.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 5. Доменные модели

Самостоятельная работа

Необходимо создать по [аналогии](#) рецепт приготовления блюда. Нужно указать список ингредиентов.

Объем / вес. Обязательно указать пошаговый список и время приготовления.
Формат **Markdown** / **HTML**

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания.

№ 7. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 6. Фабрика (Factory).

**Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .
Практическое занятие № 6. Фабрика (Factory)**

Самостоятельная работа

Необходимо

- Доработать в класс **settings_manager** Добавить загрузку данных Расширить класс **settings_model**. Добавить поле **response_format** Следующие значения

- CSV
- Markdown
- Json
- XML

- Доработать класс **factory_entities**. Инкапсулировать в него настройки. Добавить еще один метод, который будет создавать нужный ответ в зависимости от настроек (**create_default**).

- Создать классы наследники для формирования response структур в форматах: **Markdown**, **XML** и т.д. Подключить в класс **factory_entities**

- Написать модульные тест для проверки формирования данных . Проверка на корректность состава данных.

- Результаты сформированных данных (в виде файлов) включить в репо. Можно написать автотест, который сформирует файлы (группы, единицы измерений, номенклатура, рецепты)

6. Доработать main.py Добавить методы которые формирую данные в указанном формате (**Markdown**, **XML** и т.д.) и для указанного типа (**Номенклатура, Единица измерения и т.д.**)

7. Приложить в качестве результата screenshot с выводом данных из RestApi

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 8. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие №7. Сериализация данных.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие №7. Сериализация данных

Самостоятельная работа

Необходимо:

- Создать абстрактный класс **abstract_convertor**. В нем создать абстрактный метод Convert. Данный метод должен возвращать словарь в формате ключ - наименование поля, значение - данные поля. На вход данный метод должен получить любой объект.
- Создать реализацию от абстрактного класса для формирования словаря по простым типам: numeric, str - **basic_convertor**
- Создать реализацию от абстрактного класса для формирования словаря по типу DateTime - **datetime_convertor**
- Создать реализацию от абстрактного класса для формирования словаря по типу Reference - **reference_convertor**
- Создать набор модульных тестов, которые проверяют конвертицию простых и составных типов.
- Создать класс **convert_factory** по шаблону "Фабрика". Данный класс должен принимать любой объект любого типа и формировать набор словарей согласно исходной структуры используя ; **basic_convertor, datetime_convertor, reference_convertor**
- Написать модульный тест для проверки работы **convert_factory** методом строго в **Json** формате.

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 9. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 8. Десериализация данных.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 8. Десериализация данных

Самостоятельная работа

- Необходимо разработать механизм десериализации данных в формате Json в текущие модели данных.
- Номерклатура, Единица измерения, Рецепт и т.д.
- Желательно, выполнить реализацию с использованием рекурсии и шаблона Фабрика.
- Добавить автотесты, которые будут сравнивать загруженные модели с сформированными моделями в Начальной загрузке ([start_manager](#))

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 10. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 10. Прототип (Prototype).

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 10. Прототип (Prototype)

Самостоятельная работа

В целях реализации различных вариантов фильтрации необходимо

1 Создать перечисление с вариантами:

минимум

- EQUALS (Полное совпадение)
- LIKE (Вхождение строки)

2 Разработать DTO модель данных с помощью которой будет проводиться фильтрация данных по одной из DOMAIN моделей:

- Номенклатура
 - Группа
- Единица измерения

- Рецепт

3 Фильтровать по полям

- Наименование
- Уникальный код

3 Доработать шаблон "Прототип" который будет фильтровать данные в зависимости от модели DTO. **Желательно**, на все DOMAIN модели использовать один прототип.

4 Предусмотреть поиск во вложенных структурах: единица измерения - базовая единица (группа - группа владельца) и т.д.

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 11. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 11. RestApi (Dto, Swagger).

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 11. RestApi (Dto, Swagger)

Самостоятельная работа

- Создать GET запрос на получение справочных данных. В запросе в Url необходимо указать тип доменной сущности: номенклатура, единица измерения, рецепт и т.д

Так же, необходимо указывать формат передаваемых данных: XML, JSON, CSV

- Создать GET запрос на получение Оборотно-сальдовой ведомости (ОСВ) за указанный период.

- Подключить swagger к проекту согласно [Легенде](#) .

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 12. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 12. Наблюдатель (Observer).

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 12. Наблюдатель (Observer)

Самостоятельная работа

Необходимо

1. Создать класс **reference_service** в котором реализовать

- Добавление
- Изменение
- Удаление

Номенклатура, Единица измерения, Группа, Склад

2 В классе **reference_service** использовать класс прототип для поиска информации.

3 Подключить новый сервис к Web методам

- /api/[reference_type] **GETPUT** (добавление)
- /api/[reference_type] **PATCH** (изменение)
- /api/[reference_type] **DELETE** (удаление)

4 Добавить в проект реализацию шаблона наблюдатель. Добавить следующие пост обработчики:

- - При удалении. **Проверять**, наличие номенклатуры в рецепте или в сохраненных данных. Если есть такая номенклатура, то блокировать удаление. Аналогичное поведение реализовать для остальных типов.
- - При изменении, менять свойства номенклатуры : рецепт, сохраненные обороты и т.д.
- - Необходимо пересчитывать остатки по дате блокировки на основе
- - Записывать изменения в настройках (**appsettings.json**)

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

2.3.3. Практические задания для оценки компетенции «ПК-1 ППК-Р1.5»

№ 13. Кейс №1. Работа с чужим кодом: модернизация приложения «Календарь» (сам. работа).

Кейс №1. Работа с чужим кодом: модернизация приложения «Календарь» (сам. работа)

В указанном примере (календарь) заменить название месяцев / дней недели на русский вариант

Добавить функционал:

- Добавить возможность формирования календаря по произвольному периоду. Сейчас строится от текущего месяца и года.
- Добавить отображение праздничных дней в календаре. Фон: **светло коричневый**. Цвет текста: **темно серый**.
- Изменить отображение календаря. Начало недели с **понедельника**. Окончание: **суббота, воскресенье**.
- Обязательно зафиксировать время, затраченное на изменение (в часах).

№ 14. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" Практическое занятие 1. Организация проекта.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка"

Практическое занятие 1. Организация проекта

Самостоятельная работа

- Необходимо создать собственный Git репозиторий.
 - В репозиторий добавить файл readme.md в формате Markdown
- Можно использовать готовый online редакторы: <https://dillinger.io/>
- Добавить каталог Docs и в него включить приложенный файл Technicaltask.md
 - В качестве результата, прислать ссылку на Git репо
 - Придумать и написать собственные правила оформления исходного кода и структуры проекта на основе [Шаблона](#) .

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка зада

№ 15. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 2. "Шаблон Одиночка (Sigletone) " .

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 2. "Шаблон Одиночка (Sigletone) "

Необходимо;

1. Доработать класс **settings_manager**. Добавить / изменить метод convert. Данный метод должен формировать объект от класса **Settings.py** из загруженного словаря.
2. Добавить модульный тест, который проверяет загрузку данных в объект от класса **Settings.py**. Проверить загрузку всех свойств.
3. Доработать класс **settings_manager**. Сделать универсальную загрузку из файла, который может располагаться в любом каталоге.
4. Добавить модульный тест который проверит загрузку настроек из файла в другом каталоге и другого наименования.
5. Доработать класс **Settings.py** добавить следующие ограничения:
ИНН : 12 симв
Счет 11 симв
Корреспондентский счет 11 симв
БИК 9 симв
Наименование
Вид собственности 5 симв
6. Список файлов из занятия от 07/02/2024 в объявлении

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка зада

№ 16. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 3. Абстрактные классы.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 3. Абстрактные классы.

Самостоятельная работа

Необходимо доработать текущий проект. Добавить следующее:

1. Добавить модели данных: **номенклатура, единица измерения, группа номенклатура, организация, склад**

2 Все модели должны располагаться в каталоге **src/models[name]_model**

3 Все модели должны наследоваться от класса abstract_reference

4 Модель единица измерения имеет два поля; базовая единица измерения, коэффициент пересчета.

Например для единицы измерения 1 Кг - базовая единица измерения будет - 1 гр с коэффициентом пересчета = 1. Для 1 кг - коэффициент пересчет будет 1000. Для единицы измерения 1 грамм - базовая единица будет 1 грамм и коэффициент = 1

Пример:

```
base_range = range_model("грамм",1)
new_range = range_model("кг", 1000, base_range)
```

5 Модель организация должна содержать в себе: **ИНН, БИК, Счет, Форма собственности.**

6 В Модель организации добавить в конструктор класс **settings.py** и организовать копирование данных полей

Пример:

```
manager = settings_manager()
manager.open("settings.json")
company = company_model(manager.settings)
```

7 Необходимо добавить автотесты которые бы показывали корректное создание всех моделей со всеми вариантами параметров.

8 Модель номенклатура должна содержать дополнительное поле: полное наименование с длиной 255 символов.

9. Модель номенклатуры должна включать в себя другие модели; группа номенклатуры, единица измерения

10 Обычное наименование (класс **abstract_reference**) имеет ограничение на поле наименование - 50 символов.

11 Добавить автотест который продемонстрирует загрузку параметров настроек и далее создание объекта по модели организация

12 Добавить автотест который продемонстрирует работу с моделью единица измерения

13 Все обработки ошибок должны быть реализованы через внутренние классы: **argument_exception, operation_exception, error_proxu**

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания.

№ 17. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 3. Фабричный метод (Factory Method).

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 4. Фабричный метод (Factory Method)

Самостоятельная работа

- Доработать метод **create** класса **start_service** Метод должен формировать и сохранять данные по:

- Номенклатуре
- Единицам измерения
- Группам
- Создать собственный рецепт в формате **Markdown**. Включить в каталог **Docs** своего проекта.
 - Необходимо создать **модели** для описания рецептов
 - Доработать класс **start_service / create_receipts** (формирование рецептов) и доработать метод **create** для генерации данных. Добавить формирование моделей (собственный рецепт, рецепт на занятии).
 - Доработать класс **storage_repository** Добавить структуру для хранения / описания рецептов.
- Добавить модульные тесты для проверки наличия данных
 - Номенклатуре
 - Единицам измерения
 - Группам
 - Рецепты

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания.

№ 18. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 5. Доменные модели.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка".

Практическое занятие № 5. Доменные модели

Самостоятельная работа

Необходимо создать по [анalogии](#) рецепт приготовления блюда. Нужно указать список ингредиентов.

Объем / вес. Обязательно указать пошаговый список и время приготовления.

Формат **Markdown / HTML**

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания.

№ 19. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 6. Фабрика (Factory).

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка".

Практическое занятие № 6. Фабрика (Factory)

Самостоятельная работа

Необходимо

- Доработать в класс **settings_manager** Добавить загрузку данных Расширить класс settings_model. Добавить поле **response_format** Следующие значения
- CSV

- Markdown
- Json
- XML

- Доработать класс **factory_entities**. Инкапсулировать в него настройки. Добавить еще один метод, который будет создавать нужный ответ в зависимости от настроек (**create_default**).

- Создать классы наследники для формирования response структур в форматах: **Markdown**, **XML** и т.д. Подключить в класс **factory_entities**

- Написать модульные тест для проверки формирования данных . Проверка на корректность состава данных.

- Результаты сформированных данных (в виде файлов) включить в репо. Можно написать автотест, который сформирует файлы (группы, единицы измерений, номенклатура, рецепты)

6. Доработать main.py Добавить методы которые формирую данные в указанному формате (**Markdown**, **XML** и т.д.) и для указанного типа (**Номенклатура**, **Единица измерения** и т.д.)

7. Приложить в качестве результата screenshot с выводом данных из RestApi

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 20. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие №7. Сериализация данных.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .
Практическое занятие №7. Сериализация данных

Самостоятельная работа

Необходимо:

- Создать абстрактный класс **abstract_convertor**. В нем создать абстрактный метод Convert. Данный метод должен возвращать словарь в формате ключ - наименование поля, значение - данные поля. На вход данный метод должен получить любой объект.
- Создать реализацию от абстрактного класса для формирования словаря по простым типам: numeric, str - **basic_convertor**
- Создать реализацию от абстрактного класса для формирования словаря по типу DateTime - **datetime_convertor**
- Создать реализация от абстрактного класса для формирования словаря по типу Reference - **reference_convertor**
- Создать набор модульных тестов, которые проверяют конвертицию простых и составных типов.
- Создать класс **convert_factory** по шаблону "Фабрика". Данный класс должен принимать любой объект любого типа и формировать набор словарей согласно исходной структуры используя ; **basic_convertor**, **datetime_convertor**, **reference_convertor**

- Написать модульный тест для проверки работы **convert_factory** методом строго в **Json** формате.

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 21. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 8. Десериализация данных.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 8. Десериализация данных

Самостоятельная работа

- Необходимо разработать механизм десериализации данных в формате Json в текущие модели данных.
- Номенклатура, Единица измерения, Рецепт и т.д.
- Желательно, выполнить реализацию с использованием рекурсии и шаблона Фабрика.
- Добавить автотесты, которые будут сравнивать загруженные модели с сформированными моделями в Начальной загрузке ([start_manager](#))

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 22. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 10. Прототип (Prototype).

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 10. Прототип (Prototype)

Самостоятельная работа

В целях реализации различных вариантов фильтрации необходимо

1 Создать перечисление с вариантами:

минимум

- EQUALS (Полное совпадение)
- LIKE (Вхождение строки)

2 Разработать DTO модель данных с помощью которой будет проводиться фильтрация данных по одной из DOMAIN моделей:

- Номенклатура
 - Группа
- Единица измерения
 - Рецепт

3 Фильтровать по полям

- Наименование
- Уникальный код

3 Доработать шаблон "Прототип" который будет фильтровать данные в зависимости от модели DTO. **Желательно**, на все DOMAIN модели использовать один прототип.

4 Предусмотреть поиск во вложенных структурах: единица измерения - базовая единица (группа - группа владельца) и т.д.

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 23. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 11. RestApi (Dto, Swagger).

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 11. RestApi (Dto, Swagger)

Самостоятельная работа

- Создать GET запрос на получение справочных данных. В запросе в Url необходимо указать тип доменной сущности: номенклатура, единица измерения, рецепт и т.д

Так же, необходимо указывать формат передаваемых данных: XML, JSON, CSV

- Создать GET запрос на получение Оборотно-сальдовой ведомости (ОСВ) за указанный период.

- Подключить swagger к проекту согласно [Легенде](#) .

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 24. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 12. Наблюдатель (Observer).

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 12. Наблюдатель (Observer)

Самостоятельная работа

Необходимо

1. Создать класс **reference_service** в котором реализовать

- Добавление
- Изменение
- Удаление

Номенклатура, Единица измерения, Группа, Склад

2 В классе **reference_service** использовать класс прототип для поиска информации.

3 Подключить новый сервис к Web методам

- /api/[reference_type] **GETPUT** (добавление)
- /api/[reference_type] **PATCH** (изменение)
- /api/[reference_type] **DELETE** (удаление)

4 Добавить в проект реализацию шаблона наблюдатель. Добавить следующие пост обработчики:

— - При удалении. **Проверять**, наличие номенклатуры в рецепте или в сохраненных данных. Если есть такая номенклатура, то блокировать удаление. Аналогичное поведение реализовать для остальных типов.

— - При изменении, менять свойства номенклатуры : рецепт, сохраненные обороты и т.д.

- - Необходимо пересчитывать остатки по дате блокировки на основе
- - Записывать изменения в настройках (**appsettings.json**)

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

2.3.4. Практические задания для оценки компетенции «ПК-1 ППК-Р1.6»

№ 25. Кейс №1. Работа с чужим кодом: модернизация приложения «Календарь» (сам. работа).

Кейс №1. Работа с чужим кодом: модернизация приложения «Календарь» (сам. работа)

В указанном примере (календарь) заменить название месяцев / дней недели на русский вариант

Добавить функционал:

- Добавить возможность формирования календаря по произвольному периоду. Сейчас строится от текущего месяца и года.
- Добавить отображение праздничных дней в календаре. Фон: **светло коричневый**. Цвет текста: **темно серый**.
- Изменить отображение календаря. Начало недели с **понедельника**. Окончание: **суббота, воскресенье**.
- Обязательно зафиксировать время, затраченное на изменение (в часах).

№ 26. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" Практическое занятие 1. Организация проекта.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка"

Практическое занятие 1. Организация проекта

Самостоятельная работа

- Необходимо создать собственный Git репозиторий.
 - В репозиторий добавить файл readme.md в формате Markdown
- Можно использовать готовый online редакторы: <https://dillinger.io/>
- Добавить каталог Docs и в него включить приложенный файл Technicaltask.md
 - В качестве результата, прислать ссылку на Git репо
 - Придумать и написать собственные правила оформления исходного кода и структуры проекта на основе [Шаблона](#) .

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка зада

№ 27. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 2. "Шаблон Одиночка (Sigletone) " .

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 2. "Шаблон Одиночка (Sigletone) "

Необходимо;

1. Доработать класс **settings_manager**. Добавить / изменить метод convert. Данный метод должен формировать объект от класса **Settings.py** из загруженного словаря.

2. Добавить модульный тест, который проверяет загрузку данных в объект от класса Settings.py. Проверить загрузку всех свойств.
3. Доработать класс **settings_manager**. Сделать универсальную загрузку из файла, который может располагаться в любом каталоге.
4. Добавить модульный тест который проверит загрузку настроек из файла в другом каталоге и другого наименования.
5. Доработать класс **Settings.py** добавить следующие ограничения:

ИНН : 12 симв

Счет 11 симв

Корреспондентский счет 11 симв

БИК 9 симв

Наименование

Вид собственности 5 симв

6. Список файлов из занятия от 07/02/2024 в объявлении

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка зада

№ 28. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 3. Абстрактные классы.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 3. Абстрактные классы.

Самостоятельная работа

Необходимо доработать текущий проект. Добавить следующее:

- 1 Добавить модели данных:**номенклатура, единица измерения, группа номенклатура, организация, склад**

2 Все модели должны располагаться в каталоге **src/models[name]_model**

3 Все модели должны наследоваться от класса abstract_reference

4 Модель единица измерения имеет два поля; базовая единица измерения, коэффициент пересчета.

Например для единицы измерения 1 Кг - базовая единица измерения будет - 1 гр с коэффициентом пересчета = 1. Для 1 кг - коэффициент пересчет будет 1000. Для единицы измерения 1 грамм - базовая единица будет 1 грамм и коэффициент = 1

Пример:

```
base_range = range_model("грамм",1)
```

```
new_range = range_model("кг", 1000, base_range)
```

5 Модель организация должна содержать в себе:**ИНН, БИК, Счет, Форма собственности.**

6 В Модель организации добавить в конструктор класс **settings.py** и организовать копирование данных полей

Пример:

```
manager = settings_manager()
```

```
manager.open("settings.json")
```

```
company = company_model(manager.settings)
```

7 Необходимо добавить автотесты которые бы показывали корректное создание всех моделей со всеми вариантами параметров.

8 Модель номенклатура должна содержать дополнительное поле: полное наименование с длиной 255 символов.

9. Модель номенклатуры должна включать в себя другие модели; группа номенклатуры, единица измерения

10 Обычное наименование (класс **abstract_reference**) имеет ограничение на поле наименование - 50 символов.

11 Добавить автотест который продемонстрирует загрузку параметров настроек и далее создание объекта по модели организация

12 Добавить автотест который продемонстрирует работу с моделью единица измерения

13 Все обработки ошибок должны быть реализованы через внутренние классы: `argument_exception`, `operation_exception`, `error_proxy`

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания.

№ 29. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 3. Фабричный метод (Factory Method).

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка".

Практическое занятие № 4. Фабричный метод (Factory Method)

Самостоятельная работа

- Доработать метод **create** класса **start_service** Метод должен формировать и сохранять данные по:

- Номенклатуре
- Единицам измерения
- Группам

- Создать собственный рецепт в формате **Markdown**. Включить в каталог **Docs** своего проекта.

- Необходимо создать **модели** для описания рецептов

- Доработать класс **start_service** / **create_receipts** (формирование рецептов) и доработать метод **create** для генерации данных. Добавить формирование моделей (собственный рецепт, рецепт на занятии).

- Доработать класс **storage_repository** Добавить структуру для хранения / описания рецептов.

- Добавить модульные тесты для проверки наличия данных

- Номенклатуре
- Единицам измерения
- Группам
- Рецепты

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания.

№ 30. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практи-

ческое занятие № 5. Доменные модели.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 5. Доменные модели

Самостоятельная работа

Необходимо создать по [аналогии](#) рецепт приготовления блюда. Нужно указать список ингредиентов.

Объем / вес. Обязательно указать пошаговый список и время приготовления.

Формат **Markdown** / **HTML**

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания.

№ 31. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 6. Фабрика (Factory).

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 6. Фабрика (Factory)

Самостоятельная работа

Необходимо

- Доработать в класс **settings_manager** Добавить загрузку данных Расширить класс **settings_model**. Добавить поле **response_format** Следующие значения

- CSV
- Markdown
- Json
- XML

- Доработать класс **factory_entities**. Инкапсулировать в него настройки. Добавить еще один метод, который будет создавать нужный ответ в зависимости от настроек (**create_default**).

- Создать классы наследники для формирования response структур в форматах: **Markdown**, **XML** и т.д. Подключить в класс **factory_entities**

- Написать модульные тест для проверки формирования данных . Проверка на корректность состава данных.

- Результаты сформированных данных (в виде файлов) включить в репо. Можно написать автотест, который сформирует файлы (группы, единицы измерений, номенклатура, рецепты)

6. Доработать **main.py** Добавить методы которые формирую данные в указанном формате (**Markdown**, **XML** и т.д.) и для указанного типа (**Номенклатура, Единица измерения и т.д.**)

7. Приложить в качестве результата screenshot с выводом данных из RestApi

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 32. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 8. Десериализация данных.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 8. Десериализация данных

Самостоятельная работа

- Необходимо разработать механизм десериализации данных в формате Json в текущие модели данных.
- Номенклатура, Единица измерения, Рецепт и т.д.
- Желательно, выполнить реализацию с использованием рекурсии и шаблона Фабрика.
- Добавить автотесты, которые будут сравнивать загруженные модели с сформированными моделями в Начальной загрузке ([start_manager](#))

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 33. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 10. Прототип (Prototype).

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 10. Прототип (Prototype)

Самостоятельная работа

В целях реализации различных вариантов фильтрации необходимо

1 Создать перечисление с вариантами:

минимум

- EQUALS (Полное совпадение)
- LIKE (Вхождение строки)

2 Разработать DTO модель данных с помощью которой будет проводиться фильтрация данных по одной из DOMAIN моделей:

- Номенклатура
 - Группа
- Единица измерения
 - Рецепт

3 Фильтровать по полям

- Наименование
- Уникальный код

3 Доработать шаблон "Прототип" который будет фильтровать данные в зависимости от модели DTO. **Желательно**, на все DOMAIN модели использовать один прототип.

4 Предусмотреть поиск во вложенных структурах: единица измерения - базовая

единица (группа - группа владелец) и т.д.

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 34. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 12. Наблюдатель (Observer).

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 12. Наблюдатель (Observer)

Самостоятельная работа

Необходимо

1. Создать класс **reference_service** в котором реализовать

- Добавление
- Изменение
- Удаление

Номенклатура, Единица измерения, Группа, Склад

2 В классе **reference_service** использовать класс прототип для поиска информации.

3 Подключить новый сервис к Web методам

- /api/[reference_type] **GETPUT** (добавление)
- /api/[reference_type]**PATCH** (изменение)
- /api/[reference_type]**DELETE** (удаление)

4 Добавить в проект реализацию шаблона наблюдатель. Добавить следующие пост обработки:

— - При удалении. **Проверять**, наличие номенклатуры в рецепте или в сохраненных данных. Если есть такая номенклатура, то блокировать удаление. Аналогичное поведение реализовать для остальных типов.

— - При изменении, менять свойства номенклатуры : рецепт, сохраненные обороты и т.д.

— - Необходимо пересчитывать остатки по дате блокировки на основе

— - Записывать изменения в настройках (**appsettings.json**)

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

2.3.5. Практические задания для оценки компетенции «ПК-1 ППК-Р1.7»

№ 35. Кейс №1. Работа с чужим кодом: модернизация приложения «Календарь» (сам. работа).

Кейс №1. Работа с чужим кодом: модернизация приложения «Календарь» (сам. работа)

В указанном примере (календарь) заменить название месяцев / дней недели на русский вариант

Добавить функционал:

- Добавить возможность формирования календаря по произвольному периоду. Сейчас строится от текущего месяца и года.

- Добавить отображение праздничных дней в календаре. Фон: **светло коричневый**. Цвет текста: **темно серый**.

- Изменить отображение календаря. Начало недели с **понедельника**. Окончание: **суббота, воскресенье**.

- Обязательно зафиксировать время, затраченное на изменение (в часах).

№ 36. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" Практическое занятие 1. Организация проекта.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка"

Практическое занятие 1. Организация проекта

Самостоятельная работа

- Необходимо создать собственный Git репозиторий.
 - В репозиторий добавить файл readme.md в формате Markdown
- Можно использовать готовый online редакторы: <https://dillinger.io/>
- Добавить каталог Docs и в него включить приложенный файл Technicaltask.md
 - В качестве результата, прислать ссылку на Git репо
 - Придумать и написать собственные правила оформления исходного кода и структуры проекта на основе [Шаблона](#) .

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка зада

№ 37. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 2. "Шаблон Одиночка (Sigletone) "

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 2. "Шаблон Одиночка (Sigletone) "

Необходимо;

1. Доработать класс **settings_manager**. Добавить / изменить метод convert. Данный метод должен формировать объект от класса **Settings.py** из загруженного словаря.
2. Добавить модульный тест, который проверяет загрузку данных в объект от класса **Settings.py**. Проверить загрузку всех свойств.
3. Доработать класс **settings_manager**. Сделать универсальную загрузку из файла, который может располагаться в любом каталоге.
4. Добавить модульный тест который проверит загрузку настроек из файла в другом каталоге и другого наименования.
5. Доработать класс **Settings.py** добавить следующие ограничения:
ИНН : 12 симв
Счет 11 симв
Корреспондентский счет 11 симв
БИК 9 симв
Наименование
Вид собственности 5 симв
6. Список файлов из занятия от 07/02/2024 в объявлении

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка зада

№ 38. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практи-

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 3. Абстрактные классы.

Самостоятельная работа

Необходимо доработать текущий проект. Добавить следующее:

1 Добавить модели данных:**номенклатура, единица измерения, группа номенклатуры, организация, склад**

2 Все модели должны располагаться в каталоге `src/models[name]_model`

3 Все модели должны наследоваться от класса `abstract_reference`

4 Модель единица измерения имеет два поля; базовая единица измерения, коэффициент пересчета.

Например для единицы измерения 1 Кг - базовая единица измерения будет - 1 гр с коэффициентом пересчета = 1. Для 1 кг - коэффициент пересчет будет 1000. Для единицы измерения 1 грамм - базовая единица будет 1 грамм и коэффициент = 1

Пример:

```
base_range = range_model("грамм",1)
new_range = range_model("кг", 1000, base_range)
```

5 Модель организация должна содержать в себе:**ИНН, БИК, Счет, Форма собственности.**

6 В Модель организации добавить в конструктор класс `settings.py` и организовать копирование данных полей

Пример:

```
manager = settings_manager()
manager.open("settings.json")
company = company_model(manager.settings)
```

7 Необходимо добавить автотесты которые бы показывали корректное создание всех моделей со всеми вариантами параметров.

8 Модель номенклатура должна содержать дополнительное поле: полное наименование с длиной 255 символов.

9. Модель номенклатуры должна включать в себя другие модели; группа номенклатуры, единица измерения

10 Обычное наименование (класс `abstract_reference`) имеет ограничение на поле наименование - 50 символов.

11 Добавить автотест который продемонстрирует загрузку параметров настроек и далее создание объекта по модели организация

12 Добавить автотест который продемонстрирует работу с моделью единица измерения

13 Все обработки ошибок должны быть реализованы через внутренние классы: `argument_exception`, `operation_exception`, `error_proxy`

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания.

ческое занятие № 3. Фабричный метод (Factory Method).

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 4. Фабричный метод (Factory Method)

Самостоятельная работа

- Доработать метод **create** класса **start_service** Метод должен формировать и сохранять данные по:

- Номенклатуре
- Единицам измерения
- Группам

- Создать собственный рецепт в формате **Markdown**. Включить в каталог **Docs** своего проекта.

- Необходимо создать **модели** для описания рецептов

- Доработать класс **start_service** / **create_receipts** (формирование рецептов) и доработать метод **create** для генерации данных. Добавить формирование моделей (собственный рецепт, рецепт на занятии).

- Доработать класс **storage_repository** Добавить структуру для хранения / описания рецептов.

- Добавить модульные тесты для проверки наличия данных

- Номенклатуре
- Единицам измерения
- Группам
- Рецепты

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания.

№ 40. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 5. Доменные модели.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 5. Доменные модели

Самостоятельная работа

Необходимо создать по [аналогии](#) рецепт приготовления блюда. Нужно указать список ингредиентов.

Объем / вес. Обязательно указать пошаговый список и время приготовления.

Формат **Markdown** / **HTML**

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания.

№ 41. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 6. Фабрика (Factory).

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .
Практическое занятие № 6. Фабрика (Factory)

Самостоятельная работа

Необходимо

- Доработать в класс **settings_manager** Добавить загрузку данных Расширить класс **settings_model**. Добавить поле **response_format** Следующие значения
 - CSV
 - Markdown
 - Json
 - XML
- Доработать класс **factory_entities**. Инкапсулировать в него настройки. Добавить еще один метод, который будет создавать нужный ответ в зависимости от настроек (**create_default**).
- Создать классы наследники для формирования response структур в форматах: **Markdown**, **XML** и т.д. Подключить в класс **factory_entities**
- Написать модульные тест для проверки формирования данных . Проверка на корректность состава данных.
- Результаты сформированных данных (в виде файлов) включить в репо. Можно написать автотест, который сформирует файлы (группы, единицы измерений, номенклатура, рецепты)

6. Доработать **main.py** Добавить методы которые формирую данные в указанном формате (**Markdown**, **XML** и т.д.) и для указанного типа (**Номенклатура, Единица измерения и т.д.**)

7. Приложить в качестве результата screenshot с выводом данных из RestApi

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 42. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие №7. Сериализация данных.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .
Практическое занятие №7. Сериализация данных

Самостоятельная работа

Необходимо:

- Создать абстрактный класс **abstract_convertor**. В нем создать абстрактный метод **Convert** Данный метод должен возвращать словарь в формате ключ - наименование поля, значение - данные поля. На вход данный метод должен получить любой объект.
- Создать реализацию от абстрактного класса для формирования словаря по простым типам: **numeric**, **str** - **basic_convertor**
- Создать реализацию от абстрактного класса для формировать словаря по типу **DateTime** -

datetime_convertor

- Создать реализация от абстрактного класса для формирования словаря по типу **Reference - reference_convertor**
- Создать набор модульных тестов, которые проверяют конвертицию простых и составных типов.
- Создать класс **convert_factory** по шаблону "Фабрика". Данный класс должен принимать любой объект любого типа и формировать набор словарей согласно исходной структуры используя ; **basic_convertor, datetime_convertor, reference_convertor**
- Написать модульный тест для проверки работы **convert_factory** методом строго в **Json** формате.

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 43. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 8. Десериализация данных.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 8. Десериализация данных

Самостоятельная работа

- Необходимо разработать механизм десериализации данных в формате Json в текущие модели данных.
- Номенклатура, Единица измерения, Рецепт и т.д.
- Желательно, выполнить реализацию с использованием рекурсии и шаблона Фабрика.
- Добавить автотесты, которые будут сравнивать загруженные модели с сформированными моделями в Начальной загрузке ([start_manager](#))

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 44. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 10. Прототип (Prototype).

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 10. Прототип (Prototype)

Самостоятельная работа

В целях реализации различных вариантов фильтрации необходимо

1 Создать перечисление с вариантами:

минимум

- EQUALS (Полное совпадение)
- LIKE (Вхождение строки)

2 Разработать DTO модель данных с помощью которой будет проводиться фильтрация данных по одной из DOMAIN моделей:

- Номенклатура
 - Группа
- Единица измерения

- Рецепт
- 3 Фильтровать по полям
- Наименование
- Уникальный код

3 Доработать шаблон "Прототип" который будет фильтровать данные в зависимости от модели DTO. **Желательно**, на все DOMAIN модели использовать один прототип.

4 Предусмотреть поиск во вложенных структурах: единица измерения - базовая единица (группа - группа владельцев) и т.д.

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

2.3.6. Практические задания для оценки компетенции «ПК-15 РГ-2.1»

№ 45. Кейс №1. Работа с чужим кодом: модернизация приложения «Календарь» (сам. работа).

Кейс №1. Работа с чужим кодом: модернизация приложения «Календарь» (сам. работа)

В указанном примере (календарь) заменить название месяцев / дней недели на русский вариант

Добавить функционал:

- Добавить возможность формирования календаря по произвольному периоду. Сейчас строится от текущего месяца и года.
- Добавить отображение праздничных дней в календаре. Фон: **светло коричневый**. Цвет текста: **темно серый**.
- Изменить отображение календаря. Начало недели с **понедельника**. Окончание: **суббота, воскресенье**.
- Обязательно зафиксировать время, затраченное на изменение (в часах).

№ 46. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 2. "Шаблон Одиночка (Sigletone) " .

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 2. "Шаблон Одиночка (Sigletone) "

Необходимо;

1. Доработать класс **settings_manager**. Добавить / изменить метод convert. Данный метод должен формировать объект от класса **Settings.py** из загруженного словаря.
2. Добавить модульный тест, который проверяет загрузку данных в объект от класса **Settings.py**. Проверить загрузку всех свойств.
3. Доработать класс **settings_manager**. Сделать универсальную загрузку из файла, который может располагаться в любом каталоге.
4. Добавить модульный тест который проверит загрузку настроек из файла в другом каталоге и другого наименования.
5. Доработать класс **Settings.py** добавить следующие ограничения:
ИНН : 12 симв
Счет 11 симв
Корреспондентский счет 11 симв

БИК 9 симв

Наименование

Вид собственности 5 симв

6. Список файлов из занятия от 07/02/2024 в объявлении

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка зада

№ 47. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 9. Складские обороты.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 9. Складские обороты

Самостоятельная работа

Необходимо

- Добавить модели: Склад, Транзакция . Модель транзакция включает в себя следующие поля:

- Дата, Уникальный номер, Номенклатура, Склад, Количество, Единица измерения
- Доработать данные в файл default.json и в Стартовый сервис ([start_manager](#)) для формирования списка складских транзакций.

- Добавить метод / класс / структуру для расчета Оборотно-сальдовой ведомости

Возвращает таблицу вида:

- Начальный остаток на дату начала, Номенклатура, Единица измерения, Приход, Расход, Конечный остаток на дату окончания. Все обороты считаются в **базовых** единицах измерения

- Обязательно учитывать при подготовки отчета ОСВ изменения в справочниках. Например, если добавили новую номенклатуру, то в ОСВ данная номенклатура так же должна появиться.

- Добавить набор модульных / интеграционных тестов. Результат сформированного отчета ОСВ в виде картинке приложить к решению.

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 48. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 13. Логирование.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 13. Логирование

Самостоятельная работа

Необходимо реализовать в системе **reference_service**

- Складские операции: **storage_service**
- Вызовы Web методов: **main.py**
- Изменение параметров настроек: **settings.json**

2 Логирование необходимо разделить на следующие категории

- DEBUG
- INFO
- ERROR

3 Данные категории должны настраиваться в **settings.json**

- Добавить настройку "Минимальный уровень логирования". DEBUG - самый подробный вариант. В него включены варианты INFO, ERROR. INFO по аналогии - это INFO сообщения и ERROR сообщения.

4 Логирование может быть выполнено в отдельный файл или в консоль в зависимости от **settings.json**

5 Для реализации логирования обязательно использовать шаблон проектирования "Наблюдатель"

Задание сдается каждым учащимся в собственной репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

2.3.7. Практические задания для оценки компетенции «ПК-2 ППК-Р2.1»

№ 49. Кейс №1. Работа с чужим кодом: модернизация приложения «Календарь» (сам. работа).

Кейс №1. Работа с чужим кодом: модернизация приложения «Календарь» (сам. работа)

В указанном примере (календарь) заменить название месяцев / дней недели на русский вариант

Добавить функционал:

- Добавить возможность формирования календаря по произвольному периоду. Сейчас строится от текущего месяца и года.

- Добавить отображение праздничных дней в календаре. Фон: **светло коричневый**. Цвет текста: **темно серый**.

- Изменить отображение календаря. Начало недели с **понедельника**. Окончание: **суббота, воскресенье**.

- Обязательно зафиксировать время, затраченное на изменение (в часах).

№ 50. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" Практическое занятие 1. Организация проекта.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка"

Практическое занятие 1. Организация проекта

Самостоятельная работа

- Необходимо создать собственный Git репозиторий.

- В репозиторий добавить файл readme.md в формате Markdown

Можно использовать готовый online редакторы: <https://dillinger.io/>

- Добавить каталог Docs и в него включить приложенный файл Technicaltask.md

- В качестве результата, прислать ссылку на Git репо

- Придумать и написать собственные правила оформления исходного кода и структуры проекта на основе [Шаблона](#) .

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка зада

№ 51. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 2. "Шаблон Одиночка (Sigletone) ".

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 2. "Шаблон Одиночка (Sigletone) "

Необходимо;

1. Доработать класс **settings_manager**. Добавить / изменить метод `convert`. Данный метод должен формировать объект от класса **Settings.py** из загруженного словаря.
2. Добавить модульный тест, который проверяет загрузку данных в объект от класса **Settings.py**. Проверить загрузку всех свойств.
3. Доработать класс **settings_manager**. Сделать универсальную загрузку из файла, который может располагаться в любом каталоге.
4. Добавить модульный тест который проверит загрузку настроек из файла в другом каталоге и другого наименования.
5. Доработать класс **Settings.py** добавить следующие ограничения:

ИНН : 12 симв

Счет 11 симв

Корреспондентский счет 11 симв

БИК 9 симв

Наименование

Вид собственности 5 симв

6. Список файлов из занятия от 07/02/2024 в объявлении

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка зада

№ 52. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 3. Абстрактные классы.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 3. Абстрактные классы.

Самостоятельная работа

Необходимо доработать текущий проект. Добавить следующее:

1. Добавить модели данных: **номенклатура, единица измерения, группа номенклатура, организация, склад**
2. Все модели должны располагаться в каталоге `src/models[name]_model`
3. Все модели должны наследоваться от класса **`abstract_reference`**
4. Модель единица измерения имеет два поля; базовая единица измерения, коэффициент пересчета.

Например для единицы измерения 1 Кг - базовая единица измерения будет - 1 гр с коэффициентом пересчета = 1. Для 1 кг - коэффициент пересчет будет 1000. Для единицы измерения 1 грамм - базовая единица будет 1 грамм и коэффициент = 1

Пример:

```
base_range = range_model("грамм",1)
new_range = range_model("кг", 1000, base_range)
```

5. Модель организация должна содержать в себе: **ИНН, БИК, Счет, Форма собственности.**

6 В Модель организации добавить в конструктор класс **settings.py** и организовать копирование данных полей

Пример:

```
manager = settings_manager()
manager.open("settings.json")
company = company_model(manager.settings)
```

7 Необходимо добавить автотесты которые бы показывали корректное создание всех моделей со всеми вариантами параметров.

8 Модель номенклатура должна содержать дополнительное поле: полное наименование с длиной 255 символов.

9. Модель номенклатуры должна включать в себя другие модели; группа номенклатуры, единица измерения

10 Обычное наименование (класс **abstract_reference**) имеет ограничение на поле наименование - 50 символов.

11 Добавить автотест который продемонстрирует загрузку параметров настроек и далее создание объекта по модели организация

12 Добавить автотест который продемонстрирует работу с моделью единица измерения

13 Все обработки ошибок должны быть реализованы через внутренние классы: `argument_exception`, `operation_exception`, `error_proxy`

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания.

№ 53. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 3. Фабричный метод (Factory Method).

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 4. Фабричный метод (Factory Method)

Самостоятельная работа

- Доработать метод **create** класса **start_service** Метод должен формировать и сохранять данные по:

- Номенклатуре
- Единицам измерения
- Группам

- Создать собственный рецепт в формате **Markdown**. Включить в каталог **Docs** своего проекта.

- Необходимо создать **модели** для описания рецептов

- Доработать класс **start_service / create_receipts** (формирование рецептов) и доработать метод **create**

для генерации данных. Добавить формирование моделей (собственный рецепт, рецепт на занятии).

- Доработать класс **storage_repository** Добавить структуру для хранения / описания рецептов.

- Добавить модульные тесты для проверки наличия данных
- Номенклатуре
- Единицам измерения
- Группам
- Рецепты

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания.

№ 54. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 5. Доменные модели.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка".

Практическое занятие № 5. Доменные модели

Самостоятельная работа

Необходимо создать по [аналогии](#) рецепт приготовления блюда. Нужно указать список ингредиентов.

Объем / вес. Обязательно указать пошаговый список и время приготовления.

Формат **Markdown / HTML**

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания.

№ 55. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 8. Десериализация данных.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка".

Практическое занятие № 8. Десериализация данных

Самостоятельная работа

- Необходимо разработать механизм десериализации данных в формате Json в текущие модели данных.

- Номенклатура, Единица измерения, Рецепт и т.д.

- Желательно, выполнить реализацию с использованием рекурсии и шаблона Фабрика.

- Добавить автотесты, которые будут сравнивать загруженные модели с сформированными моделями в Начальной загрузке ([start_manager](#))

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 56. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 10. Прототип (Prototype).

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка".

Практическое занятие № 10. Прототип (Prototype)

Самостоятельная работа

В целях реализации различных вариантов фильтрации необходимо

1 Создать перечисление с вариантами:

минимум

- EQUALS (Полное совпадение)
- LIKE (Вхождение строки)

2 Разработать DTO модель данных с помощью которой будет проводиться фильтрация данных по одной из DOMAIN моделей:

- Номенклатура
 - Группа
- Единица измерения
 - Рецепт

3 Фильтровать по полям

- Наименование
- Уникальный код

3 Доработать шаблон "Прототип" который будет фильтровать данные в зависимости от модели DTO. **Желательно**, на все DOMAIN модели использовать один прототип.

4 Предусмотреть поиск во вложенных структурах: единица измерения - базовая единица (группа - группа владельца) и т.д.

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 57. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 13. Логирование.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка".

Практическое занятие № 13. Логирование

Самостоятельная работа

Необходимо реализовать в системе `reference_service`

- Складские операции: `storage_service`
- Вызовы Web методов: `main.py`
- Изменение параметров настроек: `settings.json`

2 Логирование необходимо разделить на следующие категории

- DEBUG
- INFO
- ERROR

3 Данные категории должны настраиваться в `settings.json`

- Добавить настройку "Минимальный уровень логирования". DEBUG - самый подробный вариант. В него включены варианты INFO, ERROR. INFO по аналогии - это INFO сообщения и ERROR сообщения.

4 Логирование может быть выполнено в отдельный файл или в консоль в зависимости от `settings.json`

5 Для реализации логирования обязательно использовать шаблон проектирования

"Наблюдатель"

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задани

2.3.8. Практические задания для оценки компетенции «ПК-2 ППК-Р2.2»

№ 58. Кейс №1. Работа с чужим кодом: модернизация приложения «Календарь» (сам. работа).

Кейс №1. Работа с чужим кодом: модернизация приложения «Календарь» (сам. работа)

В указанном примере (календарь) заменить название месяцев / дней недели на русский вариант

Добавить функционал:

- Добавить возможность формирования календаря по произвольному периоду. Сейчас строится от текущего месяца и года.

- Добавить отображение праздничных дней в календаре. Фон: **светло коричневый**. Цвет текста: **темно серый**.

- Изменить отображение календаря. Начало недели с **понедельника**. Окончание: **суббота, воскресенье**.

- Обязательно зафиксировать время, затраченное на изменение (в часах).

№ 59. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" Практическое занятие 1. Организация проекта.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка"

Практическое занятие 1. Организация проекта

Самостоятельная работа

- Необходимо создать собственный Git репозиторий.
 - В репозиторий добавить файл readme.md в формате Markdown
- Можно использовать готовый online редакторы: <https://dillinger.io/>
- Добавить каталог Docs и в него включить приложенный файл Technicaltask.md
 - В качестве результата, прислать ссылку на Git репо
 - Придумать и написать собственные правила оформления исходного кода и структуры проекта на основе [Шаблона](#) .

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка зада

№ 60. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 2. "Шаблон Одиночка (Sigletone) " .

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 2. "Шаблон Одиночка (Sigletone) "

Необходимо;

1. Доработать класс **settings_manager**. Добавить / изменить метод convert. Данный метод

должен формировать объект от класса **Settings.py** из загруженного словаря.

2. Добавить модульный тест, который проверяет загрузку данных в объект от класса **Settings.py**. Проверить загрузку всех свойств.

3. Доработать класс **settings_manager**. Сделать универсальную загрузку из файла, который может располагаться в любом каталоге.

4. Добавить модульный тест который проверит загрузку настроек из файла в другом каталоге и другого наименования.

5. Доработать класс **Settings.py** добавить следующие ограничения:

ИНН : 12 симв

Счет 11 симв

Корреспондентский счет 11 симв

БИК 9 симв

Наименование

Вид собственности 5 симв

6. Список файлов из занятия от 07/02/2024 в объявлении

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка зада

№ 61. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 3. Абстрактные классы.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 3. Абстрактные классы.

Самостоятельная работа

Необходимо доработать текущий проект. Добавить следующее:

1 Добавить модели данных:**номенклатура, единица измерения, группа номенклатура, организация, склад**

2 Все модели должны располагаться в каталоге **src/models[name]_model**

3 Все модели должны наследоваться от класса abstract_reference

4 Модель единица измерения имеет два поля; базовая единица измерения, коэффициент пересчета.

Например для единицы измерения 1 Кг - базовая единица измерения будет - 1 гр с коэффициентом пересчета = 1. Для 1 кг - коэффициент пересчет будет 1000. Для единицы измерения 1 грамм - базовая единица будет 1 грамм и коэффициент = 1

Пример:

```
base_range = range_model("грамм",1)
new_range = range_model("кг", 1000, base_range)
```

5 Модель организация должна содержать в себе:**ИНН, БИК, Счет, Форма собственности.**

6 В Модель организации добавить в конструктор класс **settings.py** и организовать копирование данных полей

Пример:

```
manager = settings_manager()
manager.open("settings.json")
company = company_model(manager.settings)
```

7 Необходимо добавить автотесты которые бы показывали корректное создание всех моделей

со всеми вариантами параметров.

8 Модель номенклатура должна содержать дополнительное поле: полное наименование с длиной 255 символов.

9. Модель номенклатуры должна включать в себя другие модели; группа номенклатуры, единица измерения

10 Обычное наименование (класс **abstract_reference**) имеет ограничение на поле наименование - 50 символов.

11 Добавить автотест который продемонстрирует загрузку параметров настроек и далее создание объекта по модели организация

12 Добавить автотест который продемонстрирует работу с моделью единица измерения

13 Все обработки ошибок должны быть реализованы через внутренние классы: `argument_exception`, `operation_exception`, `error_proxy`

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания.

№ 62. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 3. Фабричный метод (Factory Method).

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка".

Практическое занятие № 4. Фабричный метод (Factory Method)

Самостоятельная работа

- Доработать метод **create** класса **start_service** Метод должен формировать и сохранять данные по:

- Номенклатуре
- Единицам измерения
- Группам

- Создать собственный рецепт в формате **Markdown**. Включить в каталог **Docs** своего проекта.

- Необходимо создать **модели** для описания рецептов

- Доработать класс **start_service** / **create_receipts** (формирование рецептов) и доработать метод **create** для генерации данных. Добавить формирование моделей (собственный рецепт, рецепт на занятии).

- Доработать класс **storage_repository** Добавить структуру для хранения / описания рецептов.

- Добавить модульные тесты для проверки наличия данных

- Номенклатуре
- Единицам измерения
- Группам
- Рецепты

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания.

№ 63. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 5. Доменные модели.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка".

Практическое занятие № 5. Доменные модели

Самостоятельная работа

Необходимо создать по [анalogии](#) рецепт приготовления блюда. Нужно указать список ингредиентов.

Объем / вес. Обязательно указать пошаговый список и время приготовления.

Формат **Markdown / HTML**

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания.

№ 64. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 6. Фабрика (Factory).

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка".

Практическое занятие № 6. Фабрика (Factory)

Самостоятельная работа

Необходимо

- Доработать в класс **settings_manager** Добавить загрузку данных Расширить класс **settings_model**. Добавить поле **response_format** Следующие значения

- CSV
- Markdown
- Json
- XML

- Доработать класс **factory_entities**. Инкапсулировать в него настройки. Добавить еще один метод, который будет создавать нужный ответ в зависимости от настроек (**create_default**).

- Создать классы наследники для формирования response структур в форматах: **Markdown**, **XML** и т.д. Подключить в класс **factory_entities**

- Написать модульные тест для проверки формирования данных . Проверка на корректность состава данных.

- Результаты сформированных данных (в виде файлов) включить в репо. Можно написать автотест, который сформирует файлы (группы, единицы измерений, номенклатура, рецепты)

6. Доработать **main.py** Добавить методы которые формирую данные в указанном формате (**Markdown**, **XML** и т.д.) и для указанного типа (**Номенклатура, Единица измерения и т.д.**)

7. Приложить в качестве результата screenshot с выводом данных из RestApi

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 65. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие №7. Сериализация данных.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие №7. Сериализация данных

Самостоятельная работа

Необходимо:

- Создать абстрактный класс **abstract_convertor**. В нем создать абстрактный метод Convert. Данный метод должен возвращать словарь в формате ключ - наименование поля, значение - данные поля. На вход данный метод должен получить любой объект.
- Создать реализацию от абстрактного класса для формирования словаря по простым типам: numeric, str - **basic_convertor**
- Создать реализацию от абстрактного класса для формирования словаря по типу DateTime - **datetime_convertor**
- Создать реализацию от абстрактного класса для формирования словаря по типу Reference - **reference_convertor**
- Создать набор модульных тестов, которые проверяют конвертицию простых и составных типов.
- Создать класс **convert_factory** по шаблону "Фабрика". Данный класс должен принимать любой объект любого типа и формировать набор словарей согласно исходной структуры используя ; **basic_convertor, datetime_convertor, reference_convertor**
- Написать модульный тест для проверки работы **convert_factory** методом строго в **Json** формате.

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 66. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 8. Десериализация данных.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 8. Десериализация данных

Самостоятельная работа

- Необходимо разработать механизм десериализации данных в формате Json в текущие модели данных.
- Номенклатура, Единица измерения, Рецепт и т.д.
- Желательно, выполнить реализацию с использованием рекурсии и шаблона Фабрика.
- Добавить автотесты, которые будут сравнивать загруженные модели с сформированными моделями в Начальной загрузке ([start_manager](#))

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 67. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практи-

ческое занятие № 10. Прототип (Prototype).

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 10. Прототип (Prototype)

Самостоятельная работа

В целях реализации различных вариантов фильтрации необходимо

1 Создать перечисление с вариантами:

минимум

- EQUALS (Полное совпадение)
- LIKE (Вхождение строки)

2 Разработать DTO модель данных с помощью которой будет проводиться фильтрация данных по одной из DOMAIN моделей:

- Номенклатура
 - Группа
- Единица измерения
 - Рецепт

3 Фильтровать по полям

- Наименование
- Уникальный код

3 Доработать шаблон "Прототип" который будет фильтровать данные в зависимости от модели DTO. **Желательно**, на все DOMAIN модели использовать один прототип.

4 Предусмотреть поиск во вложенных структурах: единица измерения - базовая единица (группа - группа владелец) и т.д.

Задание сдается каждым учащимся в собственной репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 68. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 11. RestApi (Dto, Swagger).

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 11. RestApi (Dto, Swagger)

Самостоятельная работа

- Создать GET запрос на получение справочных данных. В запросе в Url необходимо указать тип доменной сущности: номенклатура, единица измерения, рецепт и т.д

Так же, необходимо указывать формат передаваемых данных: XML, JSON, CSV

- Создать GET запрос на получение Оборотно-сальдовой ведомости (ОСВ) за указанный период.

- Подключить swagger к проекту согласно [Легенде](#) .

Задание сдается каждым учащимся в собственной репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 69. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 12. Наблюдатель (Observer).

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка".
Практическое занятие № 12. Наблюдатель (Observer)

Самостоятельная работа

Необходимо

1. Создать класс **reference_service** в котором реализовать

- Добавление
- Изменение
- Удаление

Номенклатура, Единица измерения, Группа, Склад

2 В классе **reference_service** использовать класс прототип для поиска информации.

3 Подключить новый сервис к Web методам

- /api/[reference_type] **GETPUT** (добавление)
- /api/[reference_type] **PATCH** (изменение)
- /api/[reference_type] **DELETE** (удаление)

4 Добавить в проект реализацию шаблона наблюдатель. Добавить следующие пост обработки:

— - При удалении. **Проверять**, наличие номенклатуры в рецепте или в сохраненных данных. Если есть такая номенклатура, то блокировать удаление. Аналогичное поведение реализовать для остальных типов.

— - При изменении, менять свойства номенклатуры : рецепт, сохраненные обороты и т.д.

— - Необходимо пересчитывать остатки по дате блокировки на основе

— - Записывать изменения в настройках (**appsettings.json**)

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 70. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 13. Логирование.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка".
Практическое занятие № 13. Логирование

Самостоятельная работа

Необходимо реализовать в системе **reference_service**

- Складские операции: **storage_service**
- Вызовы Web методов: **main.py**
- Изменение параметров настроек: **settings.json**

2 Логирование необходимо разделить на следующие категории

- DEBUG
- INFO
- ERROR

3 Данные категории должны настраиваться в **settings.json**

- Добавить настройку "Минимальный уровень логирования". DEBUG - самый подробный вариант. В него включены варианты INFO, ERROR. INFO по аналогии - это INFO сообщения и ERROR сообщения.

4 Логирование может быть выполнено в отдельный файл или в консоль в зависимости от **settings.json**

5 Для реализации логирования обязательно использовать шаблон проектирования **"Наблюдатель"**

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

2.3.9. Практические задания для оценки компетенции «ПК-2 ППК-Р2.3»

№ 71. Кейс №1. Работа с чужим кодом: модернизация приложения «Календарь» (сам. работа).

Кейс №1. Работа с чужим кодом: модернизация приложения «Календарь» (сам. работа)

В указанном примере (календарь) заменить название месяцев / дней недели на русский вариант

Добавить функционал:

- Добавить возможность формирования календаря по произвольному периоду. Сейчас строится от текущего месяца и года.

- Добавить отображение праздничных дней в календаре. Фон: **светло коричневый**. Цвет текста: **темно серый**.

- Изменить отображение календаря. Начало недели с **понедельника**. Окончание: **суббота, воскресенье**.

- Обязательно зафиксировать время, затраченное на изменение (в часах).

№ 72. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" Практическое занятие 1. Организация проекта.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка"

Практическое занятие 1. Организация проекта

Самостоятельная работа

- Необходимо создать собственный Git репозиторий.

- В репозиторий добавить файл readme.md в формате Markdown

Можно использовать готовый online редакторы: <https://dillinger.io/>

- Добавить каталог Docs и в него включить приложенный файл Technicaltask.md

- В качестве результата, прислать ссылку на Git репо

- Придумать и написать собственные правила оформления исходного кода и структуры проекта на основе [Шаблона](#).

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка зада

№ 73. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 2. "Шаблон Одиночка (Sigletone) ".

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 2. "Шаблон Одиночка (Sigletone) "

Необходимо;

1. Доработать класс **settings_manager**. Добавить / изменить метод **convert**. Данный метод должен формировать объект от класса **Settings.py** из загруженного словаря.
2. Добавить модульный тест, который проверяет загрузку данных в объект от класса **Settings.py**. Проверить загрузку всех свойств.
3. Доработать класс **settings_manager**. Сделать универсальную загрузку из файла, который может располагаться в любом каталоге.
4. Добавить модульный тест который проверит загрузку настроек из файла в другом каталоге и другого наименования.
5. Доработать класс **Settings.py** добавить следующие ограничения:

ИНН : 12 симв

Счет 11 симв

Корреспондентский счет 11 симв

БИК 9 симв

Наименование

Вид собственности 5 симв

6. Список файлов из занятия от 07/02/2024 в объявлении

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка зада

№ 74. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 3. Абстрактные классы.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 3. Абстрактные классы.

Самостоятельная работа

Необходимо доработать текущий проект. Добавить следующее:

1. Добавить модели данных: **номенклатура, единица измерения, группа номенклатура, организация, склад**
2. Все модели должны располагаться в каталоге **src/models[name]_model**
3. Все модели должны наследоваться от класса **abstract_reference**
4. Модель единица измерения имеет два поля; базовая единица измерения, коэффициент пересчета.

Например для единицы измерения 1 Кг - базовая единица измерения будет - 1 гр с коэффициентом пересчета = 1. Для 1 кг - коэффициент пересчет будет 1000. Для единицы измерения 1 грамм - базовая единица будет 1 грамм и коэффициент = 1

Пример:

```
base_range = range_model("грамм",1)
```

```
new_range = range_model("кг", 1000, base_range)
```

5. Модель организация должна содержать в себе: **ИНН, БИК, Счет, Форма собственности.**

6 В Модель организации добавить в конструктор класс **settings.py** и организовать копирование данных полей

Пример:

```
manager = settings_manager()
manager.open("settings.json")
company = company_model(manager.settings)
```

7 Необходимо добавить автотесты которые бы показывали корректное создание всех моделей со всеми вариантами параметров.

8 Модель номенклатура должна содержать дополнительное поле: полное наименование с длиной 255 символов.

9. Модель номенклатуры должна включать в себя другие модели; группа номенклатуры, единица измерения

10 Обычное наименование (класс **abstract_reference**) имеет ограничение на поле наименование - 50 символов.

11 Добавить автотест который продемонстрирует загрузку параметров настроек и далее создание объекта по модели организация

12 Добавить автотест который продемонстрирует работу с моделью единица измерения

13 Все обработки ошибок должны быть реализованы через внутренние классы: `argument_exception`, `operation_exception`, `error_proxy`

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания.

№ 75. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 3. Фабричный метод (Factory Method).

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 4. Фабричный метод (Factory Method)

Самостоятельная работа

- Доработать метод **create** класса **start_service** Метод должен формировать и сохранять данные по:

- Номенклатуре
- Единицам измерения
- Группам

- Создать собственный рецепт в формате **Markdown**. Включить в каталог **Docs** своего проекта.

- Необходимо создать **модели** для описания рецептов

- Доработать класс **start_service / create_receipts** (формирование рецептов) и доработать метод **create**

для генерации данных. Добавить формирование моделей (собственный рецепт, рецепт на занятии).

- Доработать класс **storage_repository** Добавить структуру для хранения / описания рецептов.

- Добавить модульные тесты для проверки наличия данных
- Номенклатуре
- Единицам измерения
- Группам
- Рецепты

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания.

№ 76. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 5. Доменные модели.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка".

Практическое занятие № 5. Доменные модели

Самостоятельная работа

Необходимо создать по [анalogии](#) рецепт приготовления блюда. Нужно указать список ингредиентов.

Объем / вес. Обязательно указать пошаговый список и время приготовления.

Формат **Markdown / HTML**

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания.

№ 77. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 8. Десериализация данных.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка".

Практическое занятие № 8. Десериализация данных

Самостоятельная работа

- Необходимо разработать механизм десериализации данных в формате Json в текущие модели данных.

- Номенклатура, Единица измерения, Рецепт и т.д.

- Желательно, выполнить реализацию с использованием рекурсии и шаблона Фабрика.

- Добавить автотесты, которые будут сравнивать загруженные модели с сформированными моделями в Начальной загрузке ([start_manager](#))

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 78. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 12. Наблюдатель (Observer).

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка".

Практическое занятие № 12. Наблюдатель (Observer)

Самостоятельная работа

Необходимо

1. Создать класс **reference_service** в котором реализовать

- Добавление
- Изменение
- Удаление

Номенклатура, Единица измерения, Группа, Склад

2 В классе **reference_service** использовать класс прототип для поиска информации.

3 Подключить новый сервис к Web методам

- /api/[reference_type] **GETPUT** (добавление)
- /api/[reference_type] **PATCH** (изменение)
- /api/[reference_type] **DELETE** (удаление)

4 Добавить в проект реализацию шаблона наблюдатель. Добавить следующие пост обработчики:

— - При удалении. **Проверять**, наличие номенклатуры в рецепте или в сохраненных данных. Если есть такая номенклатура, то блокировать удаление. Аналогичное поведение реализовать для остальных типов.

— - При изменении, менять свойства номенклатуры : рецепт, сохраненные обороты и т.д.

— - Необходимо пересчитывать остатки по дате блокировки на основе

— - Записывать изменения в настройках (**appsettings.json**)

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 79. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 13. Логирование.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 13. Логирование

Самостоятельная работа

Необходимо реализовать в системе **reference_service**

- Складские операции: **storage_service**
- Вызовы Web методов: **main.py**
- Изменение параметров настроек: **settings.json**

2 Логирование необходимо разделить на следующие категории

- DEBUG
- INFO
- ERROR

3 Данные категории должны настраиваться в **settings.json**

- Добавить настройку "Минимальный уровень логирования". DEBUG - самый подробный вариант. В него включены варианты INFO, ERROR. INFO по аналогии - это INFO сообщения и ERROR сообщения.

4 Логирование может быть выполнено в отдельный файл или в консоль в зависимости от **settings.json**

5 Для реализации логирования обязательно использовать шаблон проектирования "Наблюдатель"

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

2.3.10. Практические задания для оценки компетенции «ПК-2 ППК-Р2.4»

№ 80. Кейс №1. Работа с чужим кодом: модернизация приложения «Календарь» (сам. работа).

Кейс №1. Работа с чужим кодом: модернизация приложения «Календарь» (сам. работа)

В указанном примере (календарь) заменить название месяцев / дней недели на русский вариант

Добавить функционал:

- Добавить возможность формирования календаря по произвольному периоду. Сейчас строится от текущего месяца и года.

- Добавить отображение праздничных дней в календаре. Фон: **светло коричневый**. Цвет текста: **темно серый**.

- Изменить отображение календаря. Начало недели с **понедельника**. Окончание: **суббота, воскресенье**.

- Обязательно зафиксировать время, затраченное на изменение (в часах).

№ 81. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" Практическое занятие 1. Организация проекта.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка"

Практическое занятие 1. Организация проекта

Самостоятельная работа

- Необходимо создать собственный Git репозиторий.
 - В репозиторий добавить файл readme.md в формате Markdown
- Можно использовать готовый online редакторы: <https://dillinger.io/>
- Добавить каталог Docs и в него включить приложенный файл Technicaltask.md
 - В качестве результата, прислать ссылку на Git репо
 - Придумать и написать собственные правила оформления исходного кода и структуры проекта на основе [Шаблона](#) .

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 82. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 2. "Шаблон Одиночка (Sigleton) "

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .
Практическое занятие № 2. "Шаблон Одиночка (Sigleton) "

Необходимо;

1. Доработать класс **settings_manager**. Добавить / изменить метод **convert**. Данный метод должен формировать объект от класса **Settings.py** из загруженного словаря.
2. Добавить модульный тест, который проверяет загрузку данных в объект от класса **Settings.py**. Проверить загрузку всех свойств.
3. Доработать класс **settings_manager**. Сделать универсальную загрузку из файла, который может располагаться в любом каталоге.
4. Добавить модульный тест который проверит загрузку настроек из файла в другом каталоге и другого наименования.
5. Доработать класс **Settings.py** добавить следующие ограничения:

ИНН : 12 симв

Счет 11 симв

Корреспондентский счет 11 симв

БИК 9 симв

Наименование

Вид собственности 5 симв

6. Список файлов из занятия от 07/02/2024 в объявлении

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка зада

№ 83. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 3. Абстрактные классы.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .
Практическое занятие № 3. Абстрактные классы.

Самостоятельная работа

Необходимо доработать текущий проект. Добавить следующее:

- 1 Добавить модели данных:**номенклатура, единица измерения, группа номенклатура, организация, склад**

2 Все модели должны располагаться в каталоге **src/models[name]_model**

3 Все модели должны наследоваться от класса **abstract_reference**

4 Модель единица измерения имеет два поля; базовая единица измерения, коэффициент пересчета.

Например для единицы измерения 1 Кг - базовая единица измерения будет - 1 гр с коэффициентом пересчета = 1. Для 1 кг - коэффициент пересчет будет 1000. Для единицы измерения 1 грамм - базовая единица будет 1 грамм и коэффициент = 1

Пример:

```
base_range = range_model("грамм",1)
```

```
new_range = range_model("кг", 1000, base_range)
```

5 Модель организация должна содержать в себе:**ИНН, БИК, Счет, Форма собственности.**

6 В Модель организации добавить в конструктор класс **settings.py** и организовать копирование данных полей

Пример:

```
manager = settings_manager()
manager.open("settings.json")
company = company_model(manager.settings)
```

7 Необходимо добавить автотесты которые бы показывали корректное создание всех моделей со всеми вариантами параметров.

8 Модель номенклатура должна содержать дополнительное поле: полное наименование с длиной 255 символов.

9. Модель номенклатуры должна включать в себя другие модели; группа номенклатуры, единица измерения

10 Обычное наименование (класс **abstract_reference**) имеет ограничение на поле наименование - 50 символов.

11 Добавить автотест который продемонстрирует загрузку параметров настроек и далее создание объекта по модели организация

12 Добавить автотест который продемонстрирует работу с моделью единица измерения

13 Все обработки ошибок должны быть реализованы через внутренние классы: `argument_exception`, `operation_exception`, `error_proxy`

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания.

№ 84. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 3. Фабричный метод (Factory Method).

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 4. Фабричный метод (Factory Method)

Самостоятельная работа

- Доработать метод **create** класса **start_service** Метод должен формировать и сохранять данные по:

- Номенклатуре
- Единицам измерения
- Группам

- Создать собственный рецепт в формате **Markdown**. Включить в каталог **Docs** своего проекта.

- Необходимо создать **модели** для описания рецептов

- Доработать класс **start_service / create_receipts** (формирование рецептов) и доработать метод **create** для генерации данных. Добавить формирование моделей (собственный рецепт, рецепт на занятии).

- Доработать класс **storage_repository** Добавить структуру для хранения / описания рецептов.

- Добавить модульные тесты для проверки наличия данных

- Номенклатуре
- Единицам измерения
- Группам

- Рецепты

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания.

№ 85. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 5. Доменные модели.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 5. Доменные модели

Самостоятельная работа

Необходимо создать по [анalogии](#) рецепт приготовления блюда. Нужно указать список ингредиентов.

Объем / вес. Обязательно указать пошаговый список и время приготовления.

Формат **Markdown / HTML**

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания.

№ 86. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 6. Фабрика (Factory).

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 6. Фабрика (Factory)

Самостоятельная работа

Необходимо

- Доработать в класс **settings_manager** Добавить загрузку данных Расширить класс settings_model. Добавить поле **response_format** Следующие значения

- CSV
- Markdown
- Json
- XML

- Доработать класс **factory_entities**. Инкапсулировать в него настройки. Добавить еще один метод, который будет создавать нужный ответ в зависимости от настроек (**create_default**).

- Создать классы наследники для формирования response структур в форматах: **Markdown, XML и т.д.** Подключить в класс **factory_entities**

- Написать модульные тест для проверки формирования данных . Проверка на корректность состава данных.

- Результаты сформированных данных (в виде файлов) включить в репо. Можно написать автотест, который сформирует файлы (группы, единицы измерений, номенклатура, рецепты)

6. Доработать main.py Добавить методы которые формирую данные в указанном формате

(Markdown, , XML и т.д.) и для указанного типа (Номенклатура, Единица измерения и т.д.)

7. Приложить в качестве результата screenshot с выводом данных из RestApi

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 87. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие №7. Сериализация данных.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие №7. Сериализация данных

Самостоятельная работа

Необходимо:

- Создать абстрактный класс **abstract_convertor**. В нем создать абстрактный метод Convert. Данный метод должен возвращать словарь в формате ключ - наименование поля, значение - данные поля. На вход данный метод должен получить любой объект.
- Создать реализацию от абстрактного класса для формирования словаря по простым типам: numeric, str - **basic_convertor**
- Создать реализацию от абстрактного класса для формирования словаря по типу DateTime - **datetime_convertor**
- Создать реализацию от абстрактного класса для формирования словаря по типу Reference - **reference_convertor**
- Создать набор модульных тестов, которые проверяют конвертицию простых и составных типов.
- Создать класс **convert_factory** по шаблону "Фабрика". Данный класс должен принимать любой объект любого типа и формировать набор словарей согласно исходной структуры используя ; **basic_convertor, datetime_convertor, reference_convertor**
- Написать модульный тест для проверки работы **convert_factory** методом строго в **Json** формате.

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 88. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 8. Десериализация данных.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 8. Десериализация данных Самостоятельная работа

- Необходимо разработать механизм десериализации данных в формате Json в текущие модели данных.
- Номенклатура, Единица измерения, Рецепт и т.д.
- Желательно, выполнить реализацию с использованием рекурсии и шаблона Фабрика.
- Добавить автотесты, которые будут сравнивать загруженные модели с сформированными моделями в Начальной загрузке ([start_manager](#))

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 89. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 10. Прототип (Prototype).

**Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .
Практическое занятие № 10. Прототип (Prototype)**

Самостоятельная работа

В целях реализации различных вариантов фильтрации необходимо

1 Создать перечисление с вариантами:

минимум

- EQUALS (Полное совпадение)
- LIKE (Вхождение строки)

2 Разработать DTO модель данных с помощью которой будет проводиться фильтрация данных по одной из DOMAIN моделей:

- Номенклатура
 - Группа
- Единица измерения
 - Рецепт

3 Фильтровать по полям

- Наименование
- Уникальный код

3 Доработать шаблон "Прототип" который будет фильтровать данные в зависимости от модели DTO. **Желательно**, на все DOMAIN модели использовать один прототип.

4 Предусмотреть поиск во вложенных структурах: единица измерения - базовая единица (группа - группа владельца) и т.д.

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 90. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 12. Наблюдатель (Observer).

**Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .
Практическое занятие № 12. Наблюдатель (Observer)**

Самостоятельная работа

Необходимо

1. Создать класс **reference_service** в котором реализовать

- Добавление
- Изменение
- Удаление

Номенклатура, Единица измерения, Группа, Склад

2 В классе **reference_service** использовать класс прототип для поиска информации.

3 Подключить новый сервис к Web методам

- /api/[reference_type] **GETPUT** (добавление)
- /api/[reference_type]**PATCH** (изменение)
- /api/[reference_type]**DELETE** (удаление)

4 Добавить в проект реализацию шаблона наблюдатель. Добавить следующие пост обработчики:

- - При удалении. **Проверять**, наличие номенклатуры в рецепте или в сохраненных данных. Если есть такая номенклатура, то блокировать удаление. Аналогичное поведение реализовать для остальных типов.
- - При изменении, менять свойства номенклатуры : рецепт, сохраненные обороты и т.д.
- - Необходимо пересчитывать остатки по дате блокировки на основе
- - Записывать изменения в настройках (**appsettings.json**)

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 91. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 13. Логирование.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 13. Логирование

Самостоятельная работа

Необходимо реализовать в системе **reference_service**

- Складские операции: **storage_service**
- Вызовы Web методов: **main.py**
- Изменение параметров настроек: **settings.json**

2 Логирование необходимо разделить на следующие категории

- DEBUG
- INFO
- ERROR

3 Данные категории должны настраиваться в **settings.json**

- Добавить настройку "Минимальный уровень логирования". DEBUG - самый подробный вариант. В него включены варианты INFO, ERROR. INFO по аналогии - это INFO сообщения и ERROR сообщения.

4 Логирование может быть выполнено в отдельный файл или в консоль в зависимости от **settings.json**

5 Для реализации логирования обязательно использовать шаблон проектирования "**Наблюдатель**"

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

2.3.11. Практические задания для оценки компетенции «ПК-4 ППК-Р4.1»

№ 92. Кейс №1. Работа с чужим кодом: модернизация приложения «Календарь» (сам. работа).

Кейс №1. Работа с чужим кодом: модернизация приложения «Календарь» (сам. работа)

В указанном примере (календарь) заменить название месяцев / дней недели на русский вариант

Добавить функционал:

- Добавить возможность формирования календаря по произвольному периоду. Сейчас строится от текущего месяца и года.

- Добавить отображение праздничных дней в календаре. Фон: **светло коричневый**. Цвет текста: **темно серый**.

- Изменить отображение календаря. Начало недели с **понедельника**. Окончание: **суббота, воскресенье**.

- Обязательно зафиксировать время, затраченное на изменение (в часах).

№ 93. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие №7. Сериализация данных.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие №7. Сериализация данных

Самостоятельная работа

Необходимо:

- Создать абстрактный класс **abstract_convertor**. В нем создать абстрактный метод Convert. Данный метод должен возвращать словарь в формате ключ - наименование поля, значение - данные поля. На вход данный метод должен получить любой объект.
- Создать реализацию от абстрактного класса для формирования словаря по простым типам: numeric, str - **basic_convertor**
- Создать реализацию от абстрактного класса для формирования словаря по типу DateTime - **datetime_convertor**
- Создать реализацию от абстрактного класса для формирования словаря по типу Reference - **reference_convertor**
- Создать набор модульных тестов, которые проверяют конвертицию простых и составных типов.
- Создать класс **convert_factory** по шаблону "Фабрика". Данный класс должен принимать любой объект любого типа и формировать набор словарей согласно исходной структуры используя ; **basic_convertor, datetime_convertor, reference_convertor**
- Написать модульный тест для проверки работы **convert_factory** методом строго в **Json** формате.

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 94. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практи-

ческое занятие № 8. Десериализация данных.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 8. Десериализация данных

Самостоятельная работа

- Необходимо разработать механизм десериализации данных в формате Json в текущие модели данных.
- Номенклатура, Единица измерения, Рецепт и т.д.
- Желательно, выполнить реализацию с использованием рекурсии и шаблона Фабрика.
- Добавить автотесты, которые будут сравнивать загруженные модели с сформированными моделями в Начальной загрузке ([start_manager](#))

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 95. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 9. Складские обороты.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 9. Складские обороты

Самостоятельная работа

Необходимо

- Добавить модели: Склад, Транзакция . Модель транзакция включает в себя следующие поля:
 - Дата, Уникальный номер, Номенклатура, Склад, Количество, Единица измерения
 - Доработать данные в файл default.json и в Стартовый сервис ([start_manager](#)) для формирования списка складских транзакций.
 - Добавить метод / класс / структуру для расчета Оборотно-сальдовой ведомости
- Возвращает таблицу вида:
 - Начальный остаток на дату начала, Номенклатура, Единица измерения, Приход, Расход, Конечный остаток на дату окончания. Все обороты считаются в **базовых** единицах измерения
 - Обязательно учитывать при подготовки отчета ОСВ изменения в справочниках. Например, если добавили новую номенклатуру, то в ОСВ данная номенклатура так же должна появиться.
 - Добавить набор модульных / интеграционных тестов. Результат сформированного отчета ОСВ в виде картинке приложить к решению.

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 96. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 10. Прототип (Prototype).

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 10. Прототип (Prototype)

Самостоятельная работа

В целях реализации различных вариантов фильтрации необходимо

1 Создать перечисление с вариантами:

минимум

- EQUALS (Полное совпадение)
- LIKE (Вхождение строки)

2 Разработать DTO модель данных с помощью которой будет проводиться фильтрация данных по одной из DOMAIN моделей:

- Номенклатура
 - Группа
- Единица измерения
 - Рецепт

3 Фильтровать по полям

- Наименование
- Уникальный код

3 Доработать шаблон "Прототип" который будет фильтровать данные в зависимости от модели DTO. **Желательно**, на все DOMAIN модели использовать один прототип.

4 Предусмотреть поиск во вложенных структурах: единица измерения - базовая единица (группа - группа владельцев) и т.д.

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 97. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 11. RestApi (Dto, Swagger).

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка".

Практическое занятие № 11. RestApi (Dto, Swagger)

Самостоятельная работа

- Создать GET запрос на получение справочных данных. В запросе в Url необходимо указать тип доменной сущности: номенклатура, единица измерения, рецепт и т.д

Так же, необходимо указывать формат передаваемых данных: XML, JSON, CSV

- Создать GET запрос на получение Оборотно-сальдовой ведомости (ОСВ) за указанный период.

- Подключить swagger к проекту согласно [Легенде](#) .

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 98. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 13. Логирование.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .
Практическое занятие № 13. Логирование

Самостоятельная работа

Необходимо реализовать в системе **reference_service**

- Складские операции: **storage_service**
- Вызовы Web методов: **main.py**
- Изменение параметров настроек: **settings.json**

2 Логирование необходимо разделить на следующие категории

- DEBUG
- INFO
- ERROR

3 Данные категории должны настраиваться в **settings.json**

- Добавить настройку "Минимальный уровень логирования". DEBUG - самый подробный вариант. В него включены варианты INFO, ERROR. INFO по аналогии - это INFO сообщения и ERROR сообщения.

4 Логирование может быть выполнено в отдельный файл или в консоль в зависимости от **settings.json**

5 Для реализации логирования обязательно использовать шаблон проектирования "Наблюдатель"

Задание сдается каждым учащимся в собственной репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

2.3.12. Практические задания для оценки компетенции «ПК-5 ППК-Р5.1»

№ 99. Кейс №1. Работа с чужим кодом: модернизация приложения «Календарь» (сам. работа).

Кейс №1. Работа с чужим кодом: модернизация приложения «Календарь» (сам. работа)

В указанном примере (календарь) заменить название месяцев / дней недели на русский вариант

Добавить функционал:

- Добавить возможность формирования календаря по произвольному периоду. Сейчас строится от текущего месяца и года.

- Добавить отображение праздничных дней в календаре. Фон: **светло коричневый**. Цвет текста: **темно серый**.

- Изменить отображение календаря. Начало недели с **понедельника**. Окончание: **суббота, воскресенье**.

- Обязательно зафиксировать время, затраченное на изменение (в часах).

2.3.13. Практические задания для оценки компетенции «ПК-6 ИП-2.1»

№ 100. Кейс №1. Работа с чужим кодом: модернизация приложения «Календарь» (сам. работа).

Кейс №1. Работа с чужим кодом: модернизация приложения «Календарь» (сам. работа)

В указанном примере (календарь) заменить название месяцев / дней недели на русский вариант

Добавить функционал:

- Добавить возможность формирования календаря по произвольному периоду. Сейчас строится от текущего месяца и года.

- Добавить отображение праздничных дней в календаре. Фон: **светло коричневый**. Цвет текста: **темно серый**.

- Изменить отображение календаря. Начало недели с **понедельника**. Окончание: **суббота, воскресенье**.

- Обязательно зафиксировать время, затраченное на изменение (в часах).

№ 101. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 2. "Шаблон Одиночка (Sigletone) ".

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 2. "Шаблон Одиночка (Sigletone) "

Необходимо;

1. Доработать класс **settings_manager**. Добавить / изменить метод `convert`. Данный метод должен формировать объект от класса **Settings.py** из загруженного словаря.
2. Добавить модульный тест, который проверяет загрузку данных в объект от класса **Settings.py**. Проверить загрузку всех свойств.
3. Доработать класс **settings_manager**. Сделать универсальную загрузку из файла, который может располагаться в любом каталоге.
4. Добавить модульный тест который проверит загрузку настроек из файла в другом каталоге и другого наименования.
5. Доработать класс **Settings.py** добавить следующие ограничения:

ИНН : 12 симв

Счет 11 симв

Корреспондентский счет 11 симв

БИК 9 симв

Наименование

Вид собственности 5 симв

6. Список файлов из занятия от 07/02/2024 в объявлении

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка зада

№ 102. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 3. Абстрактные классы.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 3. Абстрактные классы.

Самостоятельная работа

Необходимо доработать текущий проект. Добавить следующее:

1 Добавить модели данных: **номенклатура, единица измерения, группа номенклатуры, организация, склад**

2 Все модели должны располагаться в каталоге `src/models[name]_model`

3 Все модели должны наследоваться от класса `abstract_reference`

4 Модель единица измерения имеет два поля; базовая единица измерения, коэффициент пересчета.

Например для единицы измерения 1 Кг - базовая единица измерения будет - 1 гр с коэффициентом пересчета = 1. Для 1 кг - коэффициент пересчет будет 1000. Для единицы измерения 1 грамм - базовая единица будет 1 грамм и коэффициент = 1

Пример:

```
base_range = range_model("грамм",1)
new_range = range_model("кг", 1000, base_range)
```

5 Модель организация должна содержать в себе: **ИНН, БИК, Счет, Форма собственности.**

6 В Модель организации добавить в конструктор класс `settings.py` и организовать копирование данных полей

Пример:

```
manager = settings_manager()
manager.open("settings.json")
company = company_model(manager.settings)
```

7 Необходимо добавить автотесты которые бы показывали корректное создание всех моделей со всеми вариантами параметров.

8 Модель номенклатура должна содержать дополнительное поле: полное наименование с длиной 255 символов.

9. Модель номенклатуры должна включать в себя другие модели; группа номенклатуры, единица измерения

10 Обычное наименование (класс `abstract_reference`) имеет ограничение на поле наименование - 50 символов.

11 Добавить автотест который продемонстрирует загрузку параметров настроек и далее создание объекта по модели организация

12 Добавить автотест который продемонстрирует работу с моделью единица измерения

13 Все обработки ошибок должны быть реализованы через внутренние классы: `argument_exception`, `operation_exception`, `error_proxy`

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания.

№ 103. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 3. Фабричный метод (Factory Method).

**Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка".
Практическое занятие № 4. Фабричный метод (Factory Method)**

Самостоятельная работа

- Доработать метод **create** класса **start_service** Метод должен формировать и сохранять данные по:

- Номенклатуре
- Единицам измерения
- Группам

- Создать собственный рецепт в формате **Markdown**. Включить в каталог **Docs** своего проекта.

- Необходимо создать **модели** для описания рецептов

- Доработать класс **start_service / create_receipts** (формирование рецептов) и доработать метод **create** для генерации данных. Добавить формирование моделей (собственный рецепт, рецепт на занятии).

- Доработать класс **storage_repository** Добавить структуру для хранения / описания рецептов.

- Добавить модульные тесты для проверки наличия данных

- Номенклатуре
- Единицам измерения
- Группам
- Рецепты

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания.

№ 104. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 5. Доменные модели.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка".

Практическое занятие № 5. Доменные модели

Самостоятельная работа

Необходимо создать по [аналогии](#) рецепт приготовления блюда. Нужно указать список ингредиентов.

Объем / вес. Обязательно указать пошаговый список и время приготовления.

Формат **Markdown / HTML**

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания.

№ 105. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 6. Фабрика (Factory).

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка".

Практическое занятие № 6. Фабрика (Factory)

Самостоятельная работа

Необходимо

- Доработать в класс **settings_manager** Добавить загрузку данных Расширить класс **settings_model**. Добавить поле **response_format** Следующие значения
 - CSV
 - Markdown
 - Json
 - XML
 - Доработать класс **factory_entities**. Инкапсулировать в него настройки. Добавить еще один метод, который будет создавать нужный ответ в зависимости от настроек (**create_default**).
 - Создать классы наследники для формирования response структур в форматах: **Markdown**, **XML** и т.д. Подключить в класс **factory_entities**
 - Написать модульные тест для проверки формирования данных . Проверка на корректность состава данных.
 - Результаты сформированных данных (в виде файлов) включить в репо. Можно написать автотест, который сформирует файлы (группы, единицы измерений, номенклатура, рецепты)
6. Доработать **main.py** Добавить методы которые формирую данные в указанном формате (**Markdown**, **XML** и т.д.) и для указанного типа (**Номенклатура, Единица измерения и т.д.**)
7. Приложить в качестве результата screenshot с выводом данных из RestApi

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 106. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие №7. Сериализация данных.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие №7. Сериализация данных

Самостоятельная работа

Необходимо:

- Создать абстрактный класс **abstract_convertor**. В нем создать абстрактный метод **Convert** Данный метод должен возвращать словарь в формате ключ - наименование поля, значение - данные поля. На вход данный метод должен получить любой объект.
- Создать реализацию от абстрактного класса для формирования словаря по простым типам: **numeric**, **str** - **basic_convertor**
- Создать реализацию от абстрактного класса для формирования словаря по типу **DateTime** - **datetime_convertor**
- Создать реализация от абстрактного класса для формирования словаря по типу **Reference** - **reference_convertor**

- Создать набор модульных тестов, которые проверяют конвертицию простых и составных типов.
- Создать класс **convert_factory** по шаблону "Фабрика". Данный класс должен принимать любой объект любого типа и формировать набор словарей согласно исходной структуры используя ; **basic_convertor**, **datetime_convertor**, **reference_convertor**
- Написать модульный тест для проверки работы **convert_factory** методом строго в **Json** формате.

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 107. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 8. Десериализация данных.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 8. Десериализация данных

Самостоятельная работа

- Необходимо разработать механизм десериализации данных в формате Json в текущие модели данных.
- Номерклатура, Единица измерения, Рецепт и т.д.
- Желательно, выполнить реализацию с использованием рекурсии и шаблона Фабрика.
- Добавить автотесты, которые будут сравнивать загруженные модели с сформированными моделями в Начальной загрузке ([start_manager](#))

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 108. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 11. RestApi (Dto, Swagger).

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 11. RestApi (Dto, Swagger)

Самостоятельная работа

- Создать GET запрос на получение справочных данных. В запросе в Url необходимо указать тип доменной сущности: номенклатура, единица измерения, рецепт и т.д
- Так же, необходимо указывать формат передаваемых данных: XML, JSON, CSV
- Создать GET запрос на получение Оборотно-сальдовой ведомости (ОСВ) за указанный период.
- Подключить swagger к проекту согласно [Легенде](#) .

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 109. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 12. Наблюдатель (Observer).

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .
Практическое занятие № 12. Наблюдатель (Observer)
Самостоятельная работа

Необходимо

1. Создать класс **reference_service** в котором реализовать

- Добавление
- Изменение
- Удаление

Номенклатура, Единица измерения, Группа, Склад

2 В классе **reference_service** использовать класс прототип для поиска информации.

3 Подключить новый сервис к Web методам

- /api/[reference_type] **GETPUT** (добавление)
- /api/[reference_type] **PATCH** (изменение)
- /api/[reference_type] **DELETE** (удаление)

4 Добавить в проект реализацию шаблона наблюдатель. Добавить следующие пост обработчики:

— - При удалении. **Проверить**, наличие номенклатуры в рецепте или в сохраненных данных. Если есть такая номенклатура, то блокировать удаление. Аналогичное поведение реализовать для остальных типов.

— - При изменении, менять свойства номенклатуры : рецепт, сохраненные обороты и т.д.

— - Необходимо пересчитывать остатки по дате блокировки на основе

— - Записывать изменения в настройках (**appsettings.json**)

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

2.3.14. Практические задания для оценки компетенции «ПК-6 ИП-2.2»

№ 110. Кейс №1. Работа с чужим кодом: модернизация приложения «Календарь» (сам. работа).

Кейс №1. Работа с чужим кодом: модернизация приложения «Календарь» (сам. работа)

В указанном примере (календарь) заменить название месяцев / дней недели на русский вариант

Добавить функционал:

- Добавить возможность формирования календаря по произвольному периоду. Сейчас строится от текущего месяца и года.

- Добавить отображение праздничных дней в календаре. Фон: **светло коричневый**. Цвет текста: **темно серый**.

- Изменить отображение календаря. Начало недели с **понедельника**. Окончание: **суббота, воскресенье**.

- Обязательно зафиксировать время, затраченное на изменение (в часах).

№ 111. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 2. "Шаблон Одиночка (Sigletone) ".

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .
Практическое занятие № 2. "Шаблон Одиночка (Sigletone) "

Необходимо;

1. Доработать класс **settings_manager**. Добавить / изменить метод `convert`. Данный метод должен формировать объект от класса **Settings.py** из загруженного словаря.
2. Добавить модульный тест, который проверяет загрузку данных в объект от класса **Settings.py**. Проверить загрузку всех свойств.
3. Доработать класс **settings_manager**. Сделать универсальную загрузку из файла, который может располагаться в любом каталоге.
4. Добавить модульный тест который проверит загрузку настроек из файла в другом каталоге и другого наименования.
5. Доработать класс **Settings.py** добавить следующие ограничения:

ИНН : 12 симв

Счет 11 симв

Корреспондентский счет 11 симв

БИК 9 симв

Наименование

Вид собственности 5 симв

6. Список файлов из занятия от 07/02/2024 в объявлении

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка зада

№ 112. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 3. Абстрактные классы.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .
Практическое занятие № 3. Абстрактные классы.

Самостоятельная работа

Необходимо доработать текущий проект. Добавить следующее:

- 1 Добавить модели данных: **номенклатура, единица измерения, группа номенклатура, организация, склад**
- 2 Все модели должны располагаться в каталоге **src/models[name]_model**
- 3 Все модели должны наследоваться от класса **abstract_reference**
- 4 Модель единица измерения имеет два поля; базовая единица измерения, коэффициент пересчета.

Например для единицы измерения 1 Кг - базовая единица измерения будет - 1 гр с коэффициентом пересчета = 1. Для 1 кг - коэффициент пересчет будет 1000. Для единицы измерения 1 грамм - базовая единица будет 1 грамм и коэффициент = 1

Пример:

```
base_range = range_model("грамм",1)
```

```
new_range = range_model("кг", 1000, base_range)
```

- 5 Модель организация должна содержать в себе: **ИНН, БИК, Счет, Форма собственности.**

- 6 В Модель организации добавить в конструктор класс **settings.py** и организовать копирование данных полей

Пример:

```
manager = settings_manager()
manager.open("settings.json")
company = company_model(manager.settings)
```

7 Необходимо добавить автотесты которые бы показывали корректное создание всех моделей со всеми вариантами параметров.

8 Модель номенклатура должна содержать дополнительное поле: полное наименование с длиной 255 символов.

9. Модель номенклатуры должна включать в себя другие модели; группа номенклатуры, единица измерения

10 Обычное наименование (класс **abstract_reference**) имеет ограничение на поле наименование - 50 символов.

11 Добавить автотест который продемонстрирует загрузку параметров настроек и далее создание объекта по модели организация

12 Добавить автотест который продемонстрирует работу с моделью единица измерения

13 Все обработки ошибок должны быть реализованы через внутренние классы: `argument_exception`, `operation_exception`, `error_proxy`

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания.

№ 113. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 5. Доменные модели.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка".

Практическое занятие № 5. Доменные модели

Самостоятельная работа

Необходимо создать по [аналогии](#) рецепт приготовления блюда. Нужно указать список ингредиентов.

Объем / вес. Обязательно указать пошаговый список и время приготовления.

Формат **Markdown** / **HTML**

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания.

№ 114. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 6. Фабрика (Factory).

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка".

Практическое занятие № 6. Фабрика (Factory)

Самостоятельная работа

Необходимо

- Доработать в класс **settings_manager** Добавить загрузку данных Расширить класс **settings_model**. Добавить поле **response_format** Следующие значения
- CSV
- Markdown
- Json
- XML

- Доработать класс **factory_entities**. Инкапсулировать в него настройки. Добавить еще один метод, который будет создавать нужный ответ в зависимости от настроек (**create_default**).

- Создать классы наследники для формирования response структур в форматах: **Markdown**, **XML** и т.д. Подключить в класс **factory_entities**

- Написать модульные тест для проверки формирования данных . Проверка на корректность состава данных.

- Результаты сформированных данных (в виде файлов) включить в репо. Можно написать автотест, который сформирует файлы (группы, единицы измерений, номенклатура, рецепты)

6. Доработать **main.py** Добавить методы которые формирую данные в указанному формате (**Markdown**, **XML** и т.д.) и для указанного типа (**Номенклатура, Единица измерения и т.д.**)

7. Приложить в качестве результата **screenshot** с выводом данных из **RestApi**

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 115. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 8. Десериализация данных.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 8. Десериализация данных

Самостоятельная работа

- Необходимо разработать механизм десериализации данных в формате **Json** в текущие модели данных.

- Номенклатура, Единица измерения, Рецепт и т.д.

- Желательно, выполнить реализацию с использованием рекурсии и шаблона Фабрика.

- Добавить автотесты, которые будут сравнивать загруженные модели с сформированными моделями в Начальной загрузке ([start_manager](#))

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 116. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 12. Наблюдатель (Observer).

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .
Практическое занятие № 12. Наблюдатель (Observer)
Самостоятельная работа

Необходимо

1. Создать класс **reference_service** в котором реализовать

- Добавление
- Изменение
- Удаление

Номенклатура, Единица измерения, Группа, Склад

2 В классе **reference_service** использовать класс прототип для поиска информации.

3 Подключить новый сервис к Web методам

- /api/[reference_type] **GETPUT** (добавление)
- /api/[reference_type] **PATCH** (изменение)
- /api/[reference_type] **DELETE** (удаление)

4 Добавить в проект реализацию шаблона наблюдатель. Добавить следующие пост обработчики:

— - При удалении. **Проверить**, наличие номенклатуры в рецепте или в сохраненных данных. Если есть такая номенклатура, то блокировать удаление. Аналогичное поведение реализовать для остальных типов.

— - При изменении, менять свойства номенклатуры : рецепт, сохраненные обороты и т.д.

— - Необходимо пересчитывать остатки по дате блокировки на основе

— - Записывать изменения в настройках (**appsettings.json**)

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

2.3.15. Практические задания для оценки компетенции «ПК-5 ППК-Р5.4»

№ 117. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" Практическое занятие 1. Организация проекта.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка"
Практическое занятие 1. Организация проекта
Самостоятельная работа

- Необходимо создать собственный Git репозиторий.
- В репозиторий добавить файл readme.md в формате Markdown

Можно использовать готовый online редакторы: <https://dillinger.io/>

- Добавить каталог Docs и в него включить приложенный файл Technicaltask.md
- В качестве результата, прислать ссылку на Git репо
- Придумать и написать собственные правила оформления исходного кода и структуры проекта на основе [Шаблона](#) .

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка зада

2.3.16. Практические задания для оценки компетенции «ПК-1 ППК-Р1.1»

№ 118. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 3. Абстрактные классы.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка".

Практическое занятие № 3. Абстрактные классы.

Самостоятельная работа

Необходимо доработать текущий проект. Добавить следующее:

1 Добавить модели данных: **номенклатура, единица измерения, группа номенклатуры, организация, склад**

2 Все модели должны располагаться в каталоге `src/models[name]_model`

3 Все модели должны наследоваться от класса `abstract_reference`

4 Модель единица измерения имеет два поля; базовая единица измерения, коэффициент пересчета.

Например для единицы измерения 1 Кг - базовая единица измерения будет - 1 гр с коэффициентом пересчета = 1. Для 1 кг - коэффициент пересчет будет 1000. Для единицы измерения 1 грамм - базовая единица будет 1 грамм и коэффициент = 1

Пример:

```
base_range = range_model("грамм",1)
new_range = range_model("кг", 1000, base_range)
```

5 Модель организация должна содержать в себе: **ИНН, БИК, Счет, Форма собственности.**

6 В Модель организации добавить в конструктор класс `settings.py` и организовать копирование данных полей

Пример:

```
manager = settings_manager()
manager.open("settings.json")
company = company_model(manager.settings)
```

7 Необходимо добавить автотесты которые бы показывали корректное создание всех моделей со всеми вариантами параметров.

8 Модель номенклатура должна содержать дополнительное поле: полное наименование с длиной 255 символов.

9. Модель номенклатуры должна включать в себя другие модели; группа номенклатуры, единица измерения

10 Обычное наименование (класс `abstract_reference`) имеет ограничение на поле наименование - 50 символов.

11 Добавить автотест который продемонстрирует загрузку параметров настроек и далее создание объекта по модели организация

12 Добавить автотест который продемонстрирует работу с моделью единица измерения

13 Все обработки ошибок должны быть реализованы через внутренние классы: `argument_exception`, `operation_exception`, `error_proxy`

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания.

№ 119. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 3. Фабричный метод (Factory Method).

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка".

Практическое занятие № 4. Фабричный метод (Factory Method)

Самостоятельная работа

- Доработать метод **create** класса **start_service** Метод должен формировать и сохранять данные по:

- Номенклатуре
- Единицам измерения
- Группам

- Создать собственный рецепт в формате **Markdown**. Включить в каталог **Docs** своего проекта.

- Необходимо создать **модели** для описания рецептов

- Доработать класс **start_service / create_receipts** (формирование рецептов) и доработать метод **create** для генерации данных. Добавить формирование моделей (собственный рецепт, рецепт на занятии).

- Доработать класс **storage_repository** Добавить структуру для хранения / описания рецептов.

- Добавить модульные тесты для проверки наличия данных

- Номенклатуре
- Единицам измерения
- Группам
- Рецепты

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания.

№ 120. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 5. Доменные модели.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка".

Практическое занятие № 5. Доменные модели

Самостоятельная работа

Необходимо создать по [аналогии](#) рецепт приготовления блюда. Нужно указать список ингредиентов.

Объем / вес. Обязательно указать пошаговый список и время приготовления.

Формат **Markdown / HTML**

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания.

2.3.17. Практические задания для оценки компетенции «ПК-1 ППК-Р1.2»

№ 121. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 3. Абстрактные классы.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка".

Практическое занятие № 3. Абстрактные классы.

Самостоятельная работа

Необходимо доработать текущий проект. Добавить следующее:

1 Добавить модели данных: **номенклатура, единица измерения, группа номенклатуры, организация, склад**

2 Все модели должны располагаться в каталоге `src/models[name]_model`

3 Все модели должны наследоваться от класса `abstract_reference`

4 Модель единица измерения имеет два поля; базовая единица измерения, коэффициент пересчета.

Например для единицы измерения 1 Кг - базовая единица измерения будет - 1 гр с коэффициентом пересчета = 1. Для 1 кг - коэффициент пересчет будет 1000. Для единицы измерения 1 грамм - базовая единица будет 1 грамм и коэффициент = 1

Пример:

```
base_range = range_model("грамм",1)
new_range = range_model("кг", 1000, base_range)
```

5 Модель организация должна содержать в себе: **ИНН, БИК, Счет, Форма собственности.**

6 В Модель организации добавить в конструктор класс `settings.py` и организовать копирование данных полей

Пример:

```
manager = settings_manager()
manager.open("settings.json")
company = company_model(manager.settings)
```

7 Необходимо добавить автотесты которые бы показывали корректное создание всех моделей со всеми вариантами параметров.

8 Модель номенклатура должна содержать дополнительное поле: полное наименование с длиной 255 символов.

9. Модель номенклатуры должна включать в себя другие модели; группа номенклатуры, единица измерения

10 Обычное наименование (класс `abstract_reference`) имеет ограничение на поле наименование - 50 символов.

11 Добавить автотест который продемонстрирует загрузку параметров настроек и далее создание объекта по модели организация

12 Добавить автотест который продемонстрирует работу с моделью единица измерения

13 Все обработки ошибок должны быть реализованы через внутренние классы: `argument_exception`, `operation_exception`, `error_proxy`

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания.

№ 122. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 5. Доменные модели.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка".

Практическое занятие № 5. Доменные модели

Самостоятельная работа

Необходимо создать по [анalogии](#) рецепт приготовления блюда. Нужно указать список ингредиентов.

Объем / вес. Обязательно указать пошаговый список и время приготовления.

Формат **Markdown / HTML**

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания.

2.3.18. Практические задания для оценки компетенции «ПК-1 ППК-Р1.3»

№ 123. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 3. Абстрактные классы.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка".

Практическое занятие № 3. Абстрактные классы.

Самостоятельная работа

Необходимо доработать текущий проект. Добавить следующее:

1 Добавить модели данных: **номенклатура, единица измерения, группа номенклатура, организация, склад**

2 Все модели должны располагаться в каталоге `src/models[name]_model`

3 Все модели должны наследоваться от класса `abstract_reference`

4 Модель единица измерения имеет два поля; базовая единица измерения, коэффициент пересчета.

Например для единицы измерения 1 Кг - базовая единица измерения будет - 1 гр с коэффициентом пересчета = 1. Для 1 кг - коэффициент пересчет будет 1000. Для единицы измерения 1 грамм - базовая единица будет 1 грамм и коэффициент = 1

Пример:

```
base_range = range_model("грамм",1)
new_range = range_model("кг", 1000, base_range)
```

5 Модель организация должна содержать в себе: **ИНН, БИК, Счет, Форма собственности.**

6 В Модель организации добавить в конструктор класс `settings.py` и организовать копирование данных полей

Пример:

```
manager = settings_manager()
manager.open("settings.json")
company = company_model(manager.settings)
```

7 Необходимо добавить автотесты которые бы показывали корректное создание всех моделей

со всеми вариантами параметров.

8 Модель номенклатура должна содержать дополнительное поле: полное наименование с длиной 255 символов.

9. Модель номенклатуры должна включать в себя другие модели; группа номенклатуры, единица измерения

10 Обычное наименование (класс **abstract_reference**) имеет ограничение на поле наименование - 50 символов.

11 Добавить автотест который продемонстрирует загрузку параметров настроек и далее создание объекта по модели организация

12 Добавить автотест который продемонстрирует работу с моделью единица измерения

13 Все обработки ошибок должны быть реализованы через внутренние классы: `argument_exception`, `operation_exception`, `error_proxy`

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания.

№ 124. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 5. Доменные модели.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка".

Практическое занятие № 5. Доменные модели

Самостоятельная работа

Необходимо создать по [анalogии](#) рецепт приготовления блюда. Нужно указать список ингредиентов.

Объем / вес. Обязательно указать пошаговый список и время приготовления.

Формат **Markdown / HTML**

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания.

2.3.19. Практические задания для оценки компетенции «ПК-5 ППК-Р5.3»

№ 125. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 3. Абстрактные классы.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка".

Практическое занятие № 3. Абстрактные классы.

Самостоятельная работа

Необходимо доработать текущий проект. Добавить следующее:

1 Добавить модели данных: **номенклатура, единица измерения, группа номенклатура, организация, склад**

2 Все модели должны располагаться в каталоге `src/models[name]_model`

3 Все модели должны наследоваться от класса `abstract_reference`

4 Модель единица измерения имеет два поля; базовая единица измерения, коэффициент пересчета.

Например для единицы измерения 1 Кг - базовая единица измерения будет - 1 гр с коэффициентом пересчета = 1. Для 1 кг - коэффициент пересчет будет 1000. Для единицы измерения 1 грамм - базовая единица будет 1 грамм и коэффициент = 1

Пример:

```
base_range = range_model("грамм",1)
new_range = range_model("кг", 1000, base_range)
```

5 Модель организация должна содержать в себе: **ИНН, БИК, Счет, Форма собственности.**

6 В Модель организации добавить в конструктор класс **settings.py** и организовать копирование данных полей

Пример:

```
manager = settings_manager()
manager.open("settings.json")
company = company_model(manager.settings)
```

7 Необходимо добавить автотесты которые бы показывали корректное создание всех моделей со всеми вариантами параметров.

8 Модель номенклатура должна содержать дополнительное поле: полное наименование с длиной 255 символов.

9. Модель номенклатуры должна включать в себя другие модели; группа номенклатуры, единица измерения

10 Обычное наименование (класс **abstract_reference**) имеет ограничение на поле наименование - 50 символов.

11 Добавить автотест который продемонстрирует загрузку параметров настроек и далее создание объекта по модели организация

12 Добавить автотест который продемонстрирует работу с моделью единица измерения

13 Все обработки ошибок должны быть реализованы через внутренние классы: **argument_exception, operation_exception, error_proxy**

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания.

2.3.20. Практические задания для оценки компетенции «ПК-4 ППК-Р4.2»

№ 126. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие №7. Сериализация данных.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка".

Практическое занятие №7. Сериализация данных

Самостоятельная работа

Необходимо:

- Создать абстрактный класс **abstract_covertor**. В нем создать абстрактный метод **Convert** Данный метод должен возвращать словарь в формате ключ - наименование поля, значение - данные поля. На вход данный метод должен получить любой объект.
- Создать реализацию от абстрактного класса для формирования словаря по простым типам:

numeric, str - **basic_convertor**

- Создать реализацию от абстрактного класса для формирования словаря по типу DateTime - **datetime_convertor**
- Создать реализация от абстрактного класса для формирования словаря по типу Reference - **reference_convertor**
- Создать набор модульных тестов, которые проверяют конвертицию простых и составных типов.
- Создать класс **convert_factory** по шаблону "Фабрика". Данный класс должен принимать любой объект любого типа и формировать набор словарей согласно исходной структуры используя ; **basic_convertor, datetime_convertor, reference_convertor**
- Написать модульный тест для проверки работы **convert_factory** методом строго в **Json** формате.

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 127. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 8. Десериализация данных.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 8. Десериализация данных

Самостоятельная работа

- Необходимо разработать механизм десериализации данных в формате Json в текущие модели данных.
- Номенклатура, Единица измерения, Рецепт и т.д.
- Желательно, выполнить реализацию с использованием рекурсии и шаблона Фабрика.
- Добавить автотесты, которые будут сравнивать загруженные модели с сформированными моделями в Начальной загрузке ([start_manager](#))

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 128. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 9. Складские обороты.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 9. Складские обороты

Самостоятельная работа

Необходимо

- Добавить модели: Склад, Транзакция . Модель транзакция включает в себя следующие поля:
 - Дата, Уникальный номер, Номенклатура, Склад, Количество, Единица измерения
 - Доработать данные в файл default.json и в Стартовый сервис ([start_manager](#)) для формирования списка складских транзакций.
 - Добавить метод / класс / структуру для расчета Оборотно-сальдовой ведомости
- Возвращает таблицу вида:

- Начальный остаток на дату начала, Номенклатура, Единица измерения, Приход, Расход, Конечный остаток на дату окончания Все обороты считаются в **базовых** единицах

измерения

- Обязательно учитывать при подготовки отчета ОСВ изменения в справочниках. Например, если добавили новую номенклатуру, то в ОСВ данная номенклатура так же должна появиться.

- Добавить набор модульных / интеграционных тестов. Результат сформированного отчета ОСВ в виде картинке приложить к решению.

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 129. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 10. Прототип (Prototype).

**Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка".
Практическое занятие № 10. Прототип (Prototype)**

Самостоятельная работа

В целях реализации различных вариантов фильтрации необходимо

1 Создать перечисление с вариантами:

минимум

- EQUALS (Полное совпадение)
- LIKE (Вхождение строки)

2 Разработать DTO модель данных с помощью которой будет проводиться фильтрация данных по одной из DOMAIN моделей:

- Номенклатура
 - Группа
- Единица измерения
 - Рецепт

3 Фильтровать по полям

- Наименование
- Уникальный код

3 Доработать шаблон "Прототип" который будет фильтровать данные в зависимости от модели DTO. **Желательно**, на все DOMAIN модели использовать один прототип.

4 Предусмотреть поиск во вложенных структурах: единица измерения - базовая единица (группа - группа владелец) и т.д.

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 130. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 11. RestApi (Dto, Swagger).

**Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка".
Практическое занятие № 11. RestApi (Dto, Swagger)**

Самостоятельная работа

- Создать GET запрос на получение справочных данных. В запросе в Url необходимо указать тип доменной сущности: номенклатура, единица измерения, рецепт и т.д
- Так же, необходимо указывать формат передаваемых данных: XML, JSON, CSV
- Создать GET запрос на получение Оборотно-сальдовой ведомости (ОСВ) за указанный период.
- Подключить swagger к проекту согласно [Легенде](#) .

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 131. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 13. Логирование.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 13. Логирование

Самостоятельная работа

Необходимо реализовать в системе **reference_service**

- Складские операции: **storage_service**
- Вызовы Web методов: **main.py**
- Изменение параметров настроек: **settings.json**

2 Логирование необходимо разделить на следующие категории

- DEBUG
- INFO
- ERROR

3 Данные категории должны настраиваться в **settings.json**

- Добавить настройку "Минимальный уровень логирования". DEBUG - самый подробный вариант. В него включены варианты INFO, ERROR. INFO по аналогии - это INFO сообщения и ERROR сообщения.

4 Логирование может быть выполнено в отдельный файл или в консоль в зависимости от **settings.json**

5 Для реализации логирования обязательно использовать шаблон проектирования "Наблюдатель"

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

2.3.21. Практические задания для оценки компетенции «ПК-4 ППК-Р4.3»

№ 132. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие №7. Сериализация данных.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие №7. Сериализация данных

Самостоятельная работа

Необходимо:

- Создать абстрактный класс **abstract_convertor**. В нем создать абстрактный метод Convert. Данный метод должен возвращать словарь в формате ключ - наименование поля, значение - данные поля. На вход данный метод должен получить любой объект.
- Создать реализацию от абстрактного класса для формирования словаря по простым типам: numeric, str - **basic_convertor**
- Создать реализацию от абстрактного класса для формирования словаря по типу DateTime - **datetime_convertor**
- Создать реализацию от абстрактного класса для формирования словаря по типу Reference - **reference_convertor**
- Создать набор модульных тестов, которые проверяют конвертицию простых и составных типов.
- Создать класс **convert_factory** по шаблону "Фабрика". Данный класс должен принимать любой объект любого типа и формировать набор словарей согласно исходной структуры используя ; **basic_convertor, datetime_convertor, reference_convertor**
- Написать модульный тест для проверки работы **convert_factory** методом строго в **Json** формате.

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 133. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 8. Десериализация данных.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 8. Десериализация данных

Самостоятельная работа

- Необходимо разработать механизм десериализации данных в формате Json в текущие модели данных.
- Номерклатура, Единица измерения, Рецепт и т.д.
- Желательно, выполнить реализацию с использованием рекурсии и шаблона Фабрика.
- Добавить автотесты, которые будут сравнивать загруженные модели с сформированными моделями в Начальной загрузке ([start_manager](#))

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 134. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" . Практическое занятие № 9. Складские обороты.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 9. Складские обороты

Самостоятельная работа

Необходимо

- Добавить модели: Склад, Транзакция . Модель транзакция включает в себя следующие

поля:

- Дата, Уникальный номер, Номенклатура, Склад, Количество, Единица измерения
- Доработать данные в файл default.json и в Стартовый сервис ([start_manager](#)) для формирования списка складских транзакций.

- Добавить метод / класс / структуру для расчета Оборотно-сальдовой ведомости

Возвращает таблицу вида:

- Начальный остаток на дату начала, Номенклатура, Единица измерения, Приход, Расход, Конечный остаток на дату окончания. Все обороты считаются в **базовых** единицах измерения

- Обязательно учитывать при подготовки отчета ОСВ изменения в справочниках. Например, если добавили новую номенклатуру, то в ОСВ данная номенклатура так же должна появиться.

- Добавить набор модульных / интеграционных тестов. Результат сформированного отчета ОСВ в виде картинке приложить к решению.

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 135. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 10. Прототип (Prototype).

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 10. Прототип (Prototype)

Самостоятельная работа

В целях реализации различных вариантов фильтрации необходимо

1 Создать перечисление с вариантами:

минимум

- EQUALS (Полное совпадение)
- LIKE (Вхождение строки)

2 Разработать DTO модель данных с помощью которой будет проводиться фильтрация данных по одной из DOMAIN моделей:

- Номенклатура
 - Группа
- Единица измерения

- Рецепт

3 Фильтровать по полям

- Наименование
- Уникальный код

3 Доработать шаблон "Прототип" который будет фильтровать данные в зависимости от модели DTO. **Желательно**, на все DOMAIN модели использовать один прототип.

4 Предусмотреть поиск во вложенных структурах: единица измерения - базовая единица (группа - группа владельца) и т.д.

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 136. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 11. RestApi (Dto, Swagger).

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 11. RestApi (Dto, Swagger)

Самостоятельная работа

- Создать GET запрос на получение справочных данных. В запросе в Url необходимо указать тип доменной сущности: номенклатура, единица измерения, рецепт и т.д

Так же, необходимо указывать формат передаваемых данных: XML, JSON, CSV

- Создать GET запрос на получение Оборотно-сальдовой ведомости (ОСВ) за указанный период.

- Подключить swagger к проекту согласно [Легенде](#) .

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 137. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 13. Логирование.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 13. Логирование

Самостоятельная работа

Необходимо реализовать в систем `reference_service`

- Складские операции: `storage_service`
- Вызовы Web методов: `main.py`
- Изменение параметров настроек: `settings.json`

2 Логирование необходимо разделить на следующие категории

- DEBUG
- INFO
- ERROR

3 Данные категории должны настраиваться в `settings.json`

- Добавить настройку "Минимальный уровень логирования". DEBUG - самый подробный вариант. В него включены варианты INFO, ERROR. INFO по аналогии - это INFO сообщения и ERROR сообщения.

4 Логирование может быть выполнено в отдельный файл или в консоль в зависимости от `settings.json`

5 Для реализации логирования обязательно использовать шаблон проектирования "Наблюдатель"

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на

PullRequest. Далее, производится проверка задани

2.3.22. Практические задания для оценки компетенции «ПК-15 РГ-2.2»

№ 138. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 9. Складские обороты.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка".

Практическое занятие № 9. Складские обороты

Самостоятельная работа

Необходимо

- Добавить модели: Склад, Транзакция . Модель транзакция включает в себя следующие поля:

- Дата, Уникальный номер, Номенклатура, Склад, Количество, Единица измерения
- Доработать данные в файл default.json и в Стартовый сервис ([start_manager](#)) для формирования списка складских транзакций.

- Добавить метод / класс / структуру для расчета Оборотно-сальдовой ведомости

Возвращает таблицу вида:

- Начальный остаток на дату начала, Номенклатура, Единица измерения, Приход, Расход, Конечный остаток на дату окончания. Все обороты считаются в **базовых** единицах измерения

- Обязательно учитывать при подготовки отчета ОСВ изменения в справочниках. Например, если добавили новую номенклатуру, то в ОСВ данная номенклатура так же должна появиться.

- Добавить набор модульных / интеграционных тестов. Результат сформированного отчета ОСВ в виде картинке приложить к решению.

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

2.3.23. Практические задания для оценки компетенции «ПК-15 РГ-2.3»

№ 139. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 9. Складские обороты.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка".

Практическое занятие № 9. Складские обороты

Самостоятельная работа

Необходимо

- Добавить модели: Склад, Транзакция . Модель транзакция включает в себя следующие поля:

- Дата, Уникальный номер, Номенклатура, Склад, Количество, Единица измерения
- Доработать данные в файл default.json и в Стартовый сервис ([start_manager](#)) для формирования списка складских транзакций.

- Добавить метод / класс / структуру для расчета Оборотно-сальдовой ведомости

Возвращает таблицу вида:

- Начальный остаток на дату начала, Номенклатура, Единица измерения, Приход, Расход, Конечный остаток на дату окончания. Все обороты считаются в **базовых** единицах измерения
- Обязательно учитывать при подготовке отчета ОСВ изменения в справочниках. Например, если добавили новую номенклатуру, то в ОСВ данная номенклатура так же должна появиться.
- Добавить набор модульных / интеграционных тестов. Результат сформированного отчета ОСВ в виде картинке приложить к решению.

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

2.3.24. Практические задания для оценки компетенции «ПК-15 РГ-2.4»

№ 140. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 9. Складские обороты.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 9. Складские обороты

Самостоятельная работа

Необходимо

- Добавить модели: Склад, Транзакция . Модель транзакция включает в себя следующие поля:
 - Дата, Уникальный номер, Номенклатура, Склад, Количество, Единица измерения
 - Доработать данные в файл default.json и в Стартовый сервис ([start_manager](#)) для формирования списка складских транзакций.
 - Добавить метод / класс / структуру для расчета Оборотно-сальдовой ведомости

Возвращает таблицу вида:

- Начальный остаток на дату начала, Номенклатура, Единица измерения, Приход, Расход, Конечный остаток на дату окончания. Все обороты считаются в **базовых** единицах измерения
- Обязательно учитывать при подготовке отчета ОСВ изменения в справочниках. Например, если добавили новую номенклатуру, то в ОСВ данная номенклатура так же должна появиться.
- Добавить набор модульных / интеграционных тестов. Результат сформированного отчета ОСВ в виде картинке приложить к решению.

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

№ 141. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 14. Развертывание приложений.

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка" .

Практическое занятие № 14. Развертывание приложений

Самостоятельная работа

- Необходимо доработать docker compose, Dockerfile и настройки NGinx. Необходимо разделить на два хоста систему.
- Первый хост работает со всеми GET запросами
- Второй с остальными запросами.
- Необходимо добавить сервис для чтения и записи данных в базу данных.
- Доработать **два метода** (или доработать существующие)
 - POST api\save
 - POST api\load
- Добавить настройки (settings.json) для хранения строки подключения к базу данных и режима работы с хранилищем данных.
 - Записывать \ считывать из базы данных
 - Записывать \ считывать из файла
 - Настройки для подключения к базе данных сделать с использованием **Environment** (.env файл)
- Используя файлы с запросами к сервисам (8001, 8002 порты, bat / sh файлы / подготовленные в прошлом задании) запустить запрос на изменение любой сущности (например: наименование в номенклатуре), далее, запустить сохранение данные (POST api\save) на одном из сервисов, а на другом сервисе сформировать POST api\load и потом GET запрос.

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задани

№ 142. Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 15. Разделение ответственности (CQRS).

Кейс №2 (сквозной). "Автоматизация сети ресторанов Ромашка". Практическое занятие № 15. Разделение ответственности (CQRS)

Самостоятельная работа

Необходимо

- Создать отдельное приложение с настройками. В настройках указывать URL и тип запроса (POST / GET и прочее)
- Запустить собственное приложение в формате: 1 образ на чтение данных и 1 образ на запись данных.
- Запустить собственную утилиту и выполнить нагрузку. Далее провести измерение метрик.
- Увеличить нагрузку. Увеличивать до тех пор пока не проявиться деградация приложения.
- Запустить собственное приложение в формате: 3 образа на чтение данных и 1 образ на запись данных.
- Запустить собственную утилиту и выполнить нагрузку с предыдущими настройками. Снять основную метрику.

Задание сдается каждым учащимся в собственном репозитории (Git) в виде ссылки на PullRequest. Далее, производится проверка задания

3. Промежуточная аттестация

3.1. Методические материалы, определяющие процедуру оценивания знаний, умений, навыков и опыта деятельности

Зачет является заключительным этапом процесса формирования компетенций обучающегося при изучении дисциплины и имеет целью проверку и оценку знаний обучающегося по теории, и применению полученных знаний, умений и навыков при решении практических задач.

Зачет проводится по расписанию, сформированному учебно-методическим управлением, в сроки, предусмотренные календарным учебным графиком. Зачет принимается преподавателем, ведущим лекционные занятия.

Зачет проводится только при предъявлении обучающимся зачетной книжки и при условии выполнения всех контрольных мероприятий, предусмотренных учебным планом и рабочей программой дисциплины. Обучающимся на зачете представляется право выбрать один из билетов. Время подготовки к ответу составляет 30 минут. По истечении установленного времени обучающийся должен ответить на вопросы билета. Результаты зачета оцениваются по четырехбалльной системе и заносятся в зачетно-экзаменационную ведомость и зачетную книжку. В зачетную книжку заносятся только положительные оценки. Подписанный преподавателем экземпляр ведомости сдается не позднее следующего дня в деканат.

В случае неявки обучающегося на зачет в зачетно-экзаменационную ведомость делается отметка «не явка». Обучающиеся, не прошедшие промежуточную аттестацию по дисциплине, должны ликвидировать академическую задолженность в установленном локальными нормативными актами порядке.

3.2. Вопросы к зачету с оценкой

№	Вопрос	Код компетенции
1.	Что такое шаблон проектирования Клиент-Сервер и как он работает?	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-6 ИП-2.1, ПК-6 ИП-2.2, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-2 ППК-Р2.4
2.	Что такое процессорное разделение нагрузки и как он работает в архитектуре Клиент-Сервер?	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5
3.	Какие могут быть риски и ограничения в создании приложения на основе архитектуры Клиент-Сервер?	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-2 ППК-Р2.4
4.	Что такое многоуровневый шаблон проектирования?	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-4 ППК-Р4.1, ПК-4 ППК-Р4.2, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.2, ПК-5 ППК-Р5.3, ПК-6 ИП-2.1, ПК-6 ИП-2.2, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-2 ППК-Р2.4, ПК-15 РГ-2.1

№	Вопрос	Код компетенции
5.	Какие основные компоненты входят в многоуровневую архитектуру?	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-4 ППК-Р4.1, ПК-4 ППК-Р4.2, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.2, ПК-5 ППК-Р5.3, ПК-6 ИП-2.1, ПК-6 ИП-2.2, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-2 ППК-Р2.4, ПК-15 РГ-2.1
6.	Какие преимущества имеет многоуровневый подход в разработке программного обеспечения?	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-4 ППК-Р4.1, ПК-4 ППК-Р4.2, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.2, ПК-5 ППК-Р5.3, ПК-6 ИП-2.1, ПК-6 ИП-2.2, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-2 ППК-Р2.4, ПК-15 РГ-2.1
7.	Что такое шаблон проектирования Одиночка?	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3
8.	Какие проблемы решает шаблон проектирования Одиночка?	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3
9.	Какая разница между Одиночкой и Глобальной переменной?	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3
10.	Что такое архитектура MVC и какие ее основные компоненты?	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-4 ППК-Р4.1, ПК-4 ППК-Р4.2, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.2, ПК-5 ППК-Р5.3, ПК-6 ИП-2.1, ПК-6 ИП-2.2, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-2 ППК-Р2.4, ПК-15 РГ-2.1
11.	Какие преимущества может предоставить использование архитектуры MVC?	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-4 ППК-Р4.1, ПК-4 ППК-Р4.2, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.2, ПК-5 ППК-Р5.3, ПК-6 ИП-2.1, ПК-6 ИП-2.2, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-2 ППК-Р2.4, ПК-15 РГ-2.1
12.	Как взаимодействуют между собой компоненты архитектуры MVC?	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-4 ППК-Р4.1, ПК-4 ППК-Р4.2, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.2, ПК-5 ППК-Р5.3, ПК-6 ИП-2.1, ПК-6 ИП-2.2, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-2 ППК-Р2.4, ПК-15 РГ-2.1

№	Вопрос	Код компетенции
13.	Что такое шаблон проектирования Фабричный метод?	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-4 ППК-Р4.1, ПК-4 ППК-Р4.2, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.2, ПК-5 ППК-Р5.3, ПК-6 ИП-2.1, ПК-6 ИП-2.2, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-2 ППК-Р2.4, ПК-15 РГ-2.1
14.	В каких случаях следует использовать Фабричный метод?	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-4 ППК-Р4.1, ПК-4 ППК-Р4.2, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.2, ПК-5 ППК-Р5.3, ПК-6 ИП-2.1, ПК-6 ИП-2.2, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-2 ППК-Р2.4, ПК-15 РГ-2.1
15.	Как работает Фабричный метод в контексте наследования?	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-4 ППК-Р4.1, ПК-4 ППК-Р4.2, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.2, ПК-5 ППК-Р5.3, ПК-6 ИП-2.1, ПК-6 ИП-2.2, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-2 ППК-Р2.4, ПК-15 РГ-2.1
16.	Какие альтернативные подходы существуют для создания объектов?	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-4 ППК-Р4.1, ПК-4 ППК-Р4.2, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.2, ПК-5 ППК-Р5.3, ПК-6 ИП-2.1, ПК-6 ИП-2.2, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-2 ППК-Р2.4, ПК-15 РГ-2.1
17.	Что такое SOA (архитектура сервисно-ориентированного приложения)?	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3
18.	Какие преимущества имеют SOA приложения по сравнению с традиционными монолитными приложениями?	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3
19.	Какие компоненты входят в SOA архитектуру приложений?	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3
20.	Какова роль веб-сервисов в SOA архитектуре приложений?	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3
21.	Что такое шаблон проектирования Фабрика?	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-4 ППК-Р4.1, ПК-4 ППК-Р4.2, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.2, ПК-5 ППК-Р5.3, ПК-6 ИП-2.1, ПК-6 ИП-2.2, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-2 ППК-Р2.4, ПК-15 РГ-2.1

№	Вопрос	Код компетенции
22.	Какова цель использования данного шаблона?	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-4 ППК-Р4.1, ПК-4 ППК-Р4.2, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.2, ПК-5 ППК-Р5.3, ПК-6 ИП-2.1, ПК-6 ИП-2.2, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-2 ППК-Р2.4, ПК-15 РГ-2.1
23.	Можно ли определить свои собственные компоненты?	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-4 ППК-Р4.1, ПК-4 ППК-Р4.2, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.2, ПК-5 ППК-Р5.3, ПК-6 ИП-2.1, ПК-6 ИП-2.2, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-2 ППК-Р2.4, ПК-15 РГ-2.1
24.	Какие преимущества в использовании этого шаблона проектирования?	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-4 ППК-Р4.1, ПК-4 ППК-Р4.2, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.2, ПК-5 ППК-Р5.3, ПК-6 ИП-2.1, ПК-6 ИП-2.2, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-2 ППК-Р2.4, ПК-15 РГ-2.1
25.	Что такое шаблон проектирования "Общие данные"?	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3
26.	С какой целью применяется шаблон "Общие данные"?	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3
27.	Что такое архитектура "Издатель - подписчик"?	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-6 ИП-2.1, ПК-6 ИП-2.2, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-2 ППК-Р2.4
28.	Какие принципы лежат в основе архитектуры "Издатель - подписчик"?	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-6 ИП-2.1, ПК-6 ИП-2.2, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-2 ППК-Р2.4
29.	Какая роль играют подписчики архитектуры "Издатель - подписчик"?	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-6 ИП-2.1, ПК-6 ИП-2.2, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-2 ППК-Р2.4
30.	Какие преимущества имеет эта архитектура "Издатель - подписчик" по сравнению с другими подходами?	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-6 ИП-2.1, ПК-6 ИП-2.2, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-2 ППК-Р2.4

№	Вопрос	Код компетенции
31.	Что такое Событийно-ориентированная архитектура (СОА) и как она отличается от других подходов в архитектуре ПО?	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-4 ППК-Р4.1, ПК-4 ППК-Р4.2, ПК-15 РГ-2.1
32.	Какие основные компоненты входят в архитектуру СОА?	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-4 ППК-Р4.1, ПК-4 ППК-Р4.2, ПК-6 ИП-2.1, ПК-6 ИП-2.2, ПК-15 РГ-2.1
33.	Какие типы сообщений могут использоваться в архитектуре СОА?	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-4 ППК-Р4.1, ПК-4 ППК-Р4.2, ПК-2 ППК-Р2.4, ПК-15 РГ-2.1
34.	Как происходит мониторинг и отладка приложений, созданных на основе архитектуры СОА?	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-4 ППК-Р4.1, ПК-4 ППК-Р4.2, ПК-2 ППК-Р2.4, ПК-15 РГ-2.1
35.	Что такое шаблон проектирования Мост?	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-4 ППК-Р4.1, ПК-4 ППК-Р4.2, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.2, ПК-5 ППК-Р5.3, ПК-6 ИП-2.1, ПК-6 ИП-2.2, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-2 ППК-Р2.4, ПК-15 РГ-2.1
36.	Какие проблемы решает шаблон Мост?	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-4 ППК-Р4.1, ПК-4 ППК-Р4.2, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.2, ПК-5 ППК-Р5.3, ПК-6 ИП-2.1, ПК-6 ИП-2.2, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-2 ППК-Р2.4, ПК-15 РГ-2.1
37.	Как устранить жесткую связь между абстракцией и реализацией?	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-4 ППК-Р4.1, ПК-4 ППК-Р4.2, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.2, ПК-5 ППК-Р5.3, ПК-6 ИП-2.1, ПК-6 ИП-2.2, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-2 ППК-Р2.4, ПК-15 РГ-2.1
38.	Какая задача может быть решена с помощью шаблона Мост?	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-4 ППК-Р4.1, ПК-4 ППК-Р4.2, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.2, ПК-5 ППК-Р5.3, ПК-6 ИП-2.1, ПК-6 ИП-2.2, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-2 ППК-Р2.4, ПК-15 РГ-2.1

№	Вопрос	Код компетенции
39.	Что такое шаблон проектирования Декоратор?	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-4 ППК-Р4.1, ПК-4 ППК-Р4.2, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.2, ПК-5 ППК-Р5.3, ПК-6 ИП-2.1, ПК-6 ИП-2.2, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-2 ППК-Р2.4, ПК-15 РГ-2.1
40.	Какие проблемы решает шаблон Декоратор?	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-4 ППК-Р4.1, ПК-4 ППК-Р4.2, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.2, ПК-5 ППК-Р5.3, ПК-6 ИП-2.1, ПК-6 ИП-2.2, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-2 ППК-Р2.4, ПК-15 РГ-2.1
41.	Можно ли использовать несколько Декораторов для одного объекта?	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-4 ППК-Р4.1, ПК-4 ППК-Р4.2, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.2, ПК-5 ППК-Р5.3, ПК-6 ИП-2.1, ПК-6 ИП-2.2, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-2 ППК-Р2.4, ПК-15 РГ-2.1
42.	Что такое шаблон проектирования Наблюдатель?	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.2, ПК-5 ППК-Р5.3, ПК-2 ППК-Р2.4, ПК-15 РГ-2.1
43.	Какие возможности предоставляет шаблон Наблюдатель для модификации приложения?	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.2, ПК-5 ППК-Р5.3, ПК-2 ППК-Р2.4, ПК-15 РГ-2.1
44.	Какие преимущества и недостатки имеет использование шаблона Наблюдатель?	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.2, ПК-5 ППК-Р5.3, ПК-15 РГ-2.1
45.	В чем заключается шаблон проектирования Адаптер?	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.2, ПК-5 ППК-Р5.3, ПК-6 ИП-2.1, ПК-6 ИП-2.2, ПК-2 ППК-Р2.4, ПК-15 РГ-2.1
46.	Какой основной принцип лежит в основе Адаптера?	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.2, ПК-5 ППК-Р5.3, ПК-6 ИП-2.1, ПК-6 ИП-2.2, ПК-2 ППК-Р2.4, ПК-15 РГ-2.1
47.	В каких случаях необходимо применять Адаптер?	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.2, ПК-5 ППК-Р5.3, ПК-2 ППК-Р2.4, ПК-15 РГ-2.1

№	Вопрос	Код компетенции
48.	Что такое шаблон проектирования Заместитель (Proxy)?	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-4 ППК-Р4.1, ПК-4 ППК-Р4.2, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.2, ПК-5 ППК-Р5.3, ПК-6 ИП-2.1, ПК-6 ИП-2.2, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-2 ППК-Р2.4, ПК-15 РГ-2.1
49.	Какие задачи решает шаблон проектирования Заместитель (Proxy)?	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-4 ППК-Р4.1, ПК-4 ППК-Р4.2, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.2, ПК-5 ППК-Р5.3, ПК-6 ИП-2.1, ПК-6 ИП-2.2, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-2 ППК-Р2.4, ПК-15 РГ-2.1
50.	Как работает Удаленный заместитель (Remote Proxy)?	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-4 ППК-Р4.1, ПК-4 ППК-Р4.2, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.2, ПК-5 ППК-Р5.3, ПК-6 ИП-2.1, ПК-6 ИП-2.2, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-2 ППК-Р2.4, ПК-15 РГ-2.1
51.	Как работает Защищающий заместитель (Protective Proxy)?	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-4 ППК-Р4.1, ПК-4 ППК-Р4.2, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.2, ПК-5 ППК-Р5.3, ПК-6 ИП-2.1, ПК-6 ИП-2.2, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-2 ППК-Р2.4, ПК-15 РГ-2.1
52.	Какие преимущества имеет использование шаблона проектирования Заместитель (Proxy)?	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-4 ППК-Р4.1, ПК-4 ППК-Р4.2, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.2, ПК-5 ППК-Р5.3, ПК-6 ИП-2.1, ПК-6 ИП-2.2, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-2 ППК-Р2.4, ПК-15 РГ-2.1
53.	Что такое CQRS и как она используется в архитектуре приложений?	ПК-1 ППК-Р1.1, ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.3, ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.2, ПК-5 ППК-Р5.3, ПК-6 ИП-2.1, ПК-6 ИП-2.2, ПК-1 ППК-Р1.4, ПК-1 ППК-Р1.5, ПК-2 ППК-Р2.4, ПК-15 РГ-2.1
54.	Какую проблему решает CQRS в разработке программного обеспечения?	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.2, ПК-5 ППК-Р5.3, ПК-6 ИП-2.1, ПК-6 ИП-2.2, ПК-2 ППК-Р2.4, ПК-15 РГ-2.1

№	Вопрос	Код компетенции
55.	Как CQRS взаимодействует с другими архитектурными паттернами и практиками, такими как DDD и Event Sourcing?	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.2, ПК-5 ППК-Р5.3, ПК-6 ИП-2.1, ПК-6 ИП-2.2, ПК-2 ППК-Р2.4, ПК-15 РГ-2.1
56.	Какие рекомендации можно дать для использования CQRS в реальных проектах?	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.2, ПК-5 ППК-Р5.3, ПК-6 ИП-2.1, ПК-6 ИП-2.2, ПК-2 ППК-Р2.4, ПК-15 РГ-2.1

3.3. Тематика курсовых работ

По данной дисциплине выполнение курсовых проектов (работ) не предусматривается.

3.4. Материалы для компьютерного тестирования обучающихся

Общие критерии оценивания

Процент правильных ответов	Оценка
91% – 100%	5 (отлично)
81% – 90%	4 (хорошо)
71% – 80%	3 (удовлетворительно)
Менее 70%	2 (неудовлетворительно)

Соответствие вопросов теста индикаторам формируемых и оцениваемых компетенций

№ вопроса в тесте	Код индикатора компетенции
1	ПК-15 РГ-2.1, ПК-15 РГ-2.2, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.3, ПК-5 ППК-Р5.4
2	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3
3	ПК-4 ППК-Р4.3, ПК-5 ППК-Р5.4
4	ПК-15 РГ-2.1, ПК-15 РГ-2.2, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.3, ПК-5 ППК-Р5.4
5	ПК-15 РГ-2.1, ПК-15 РГ-2.2, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.4
6	ПК-4 ППК-Р4.2, ПК-4 ППК-Р4.3, ПК-5 ППК-Р5.4
7	ПК-4 ППК-Р4.2, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.2
8	ПК-15 РГ-2.1, ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.2
9	ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.2, ПК-5 ППК-Р5.3, ПК-5 ППК-Р5.4
10	ПК-1 ППК-Р1.1, ПК-5 ППК-Р5.1
11	ПК-4 ППК-Р4.2, ПК-5 ППК-Р5.4
12	ПК-15 РГ-2.4
13	ПК-1 ППК-Р1.4, ПК-15 РГ-2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.4, ПК-6 ИП-2.1, ПК-6 ИП-2.2
14	ПК-4 ППК-Р4.2
15	ПК-4 ППК-Р4.2, ПК-4 ППК-Р4.3

№ вопроса в тесте	Код индикатора компетенции
16	ПК-1 ППК-Р1.4, ПК-15 РГ-2.1, ПК-5 ППК-Р5.4
17	ПК-5 ППК-Р5.1, ПК-5 ППК-Р5.3, ПК-5 ППК-Р5.4
18	ПК-15 РГ-2.1, ПК-15 РГ-2.2, ПК-15 РГ-2.3, ПК-2 ППК-Р2.4, ПК-5 ППК-Р5.1
19	ПК-2 ППК-Р2.1, ПК-2 ППК-Р2.2, ПК-2 ППК-Р2.3
20	ПК-4 ППК-Р4.2, ПК-4 ППК-Р4.3, ПК-5 ППК-Р5.2
21	ПК-1 ППК-Р1.2, ПК-1 ППК-Р1.4

Ключ ответов

№ вопроса в тесте	Номер ответа (или ответ, или соответствие)
1	d
2	b
3	c
4	a
5	d
6	c
7	b
8	b
9	c
10	a
11	d
12	b
13	a
14	a
15	a
16	c
17	c
18	b
19	b
20	d
21	b

Перечень тестовых вопросов

№ 1. Задание с единичным выбором. Выберите один правильный ответ.

Какова основная идея подхода Domain-Driven Design (DDD)?

- a. Использование исключительно микросервисной архитектуры.
- b. Разделение приложения на клиентскую и серверную части.
- c. Максимальное сокращение количества классов в проекте.
- d. Проектирование программного обеспечения на основе глубокого анализа

предметной области и построения модели, отражающей бизнес-процессы.

№ 2. Задание с единичным выбором. Выберите один правильный ответ.

Для чего используется шаблон проектирования "Общие данные"?

- a. Это распространенный вариант объединения различных систем
- b. Данный шаблон является простым вариантом интеграции различных система на основе взаимодействия базы данных

№ 3. Задание с единичным выбором. Выберите один правильный ответ.

В системе автоматизации сети ресторанов «Ромашка» после изменения статуса заказа на «Готов» необходимо автоматически: уведомить курьерскую службу; обновить информацию на рабочем месте администратора; изменить статус заказа на информационном табло кухни. Какое архитектурное решение наиболее соответствует применению шаблона «Наблюдатель» (Observer)?

- a. После изменения статуса необходимо вручную обновить данные во всех подсистемах.
- b. Каждый модуль самостоятельно периодически проверяет, изменился ли статус заказа.
- c. Объект «Заказ» автоматически уведомляет все подписанные на него модули при изменении своего состояния.
- d. Все действия выполняются одним большим методом, содержащим последовательные вызовы всех подсистем.

№ 4. Задание с единичным выбором. Выберите один правильный ответ.

Какова основная цель применения шаблона проектирования Singleton (Одиночка)? а) Создание большого количества экземпляров одного класса. б) Обеспечение существования только одного экземпляра класса и предоставление глобальной точки доступа к нему. в) Автоматическое создание объектов различных типов. г) Разделение приложения на клиентскую и серверную части.

- a. Обеспечение существования только одного экземпляра класса и предоставление глобальной точки доступа к нему.
- b. Создание большого количества экземпляров одного класса.
- c. Разделение приложения на клиентскую и серверную части.
- d. Автоматическое создание объектов различных типов.

№ 5. Задание с единичным выбором. Выберите один правильный ответ.

В системе автоматизации сети ресторанов «Ромашка» необходимо реализовать создание документов различных типов: заказ поставщику, заявка на производство, маршрутный лист курьера и отчет по продажам. При этом в дальнейшем список документов может расширяться без изменения существующего кода. Какое решение наиболее соответствует применению шаблона «Фабричный метод» (Factory Method)?

- a. Создавать документы непосредственно с помощью оператора new во всех местах программы.
- b. Создать один экземпляр документа и изменять его тип во время работы программы.
- c. Создать один класс, который с помощью большого оператора if-else или switch создает все виды документов.
- d. Для каждого типа документа реализовать отдельный фабричный метод, который создает соответствующий объект, что позволит легко добавлять новые типы документов без

изменения существующей логики.

№ 6. Задание с единственным выбором. Выберите один правильный ответ.

В системе автоматизации сети ресторанов «Ромашка» официанты и кассиры постоянно создают и изменяют заказы, а руководители ресторанов одновременно просматривают отчеты по продажам, остаткам на складе и загрузенности курьеров. Для повышения производительности было принято решение разделить обработку операций изменения данных и операций их чтения. Какой архитектурный подход наиболее соответствует данному решению?

- a. Каждая подсистема самостоятельно хранит собственную копию всей информации.
- b. Все данные периодически копируются в отдельную базу данных вручную.
- c. Для операций изменения данных (команд) и операций чтения данных (запросов) используются отдельные модели и механизмы обработки.
- d. Все операции чтения и изменения данных выполняются одним сервисом и используют одну и ту же модель данных.

№ 7. Задание с единственным выбором. Выберите один правильный ответ.

В системе автоматизации сети ресторанов «Ромашка» после каждой операции поступления, списания или перемещения продукции необходимо автоматически обновлять остатки товаров на складе. Какое техническое решение наиболее корректно обеспечивает учет складских оборотов?

- a. Хранить только текущие остатки товаров без информации об операциях.
- b. После каждой операции создавать запись о складском обороте и на ее основе автоматически пересчитывать текущие остатки продукции.
- c. Изменять остатки товаров только один раз в конце рабочего дня.
- d. Пересчитывать остатки вручную администратором ресторана.

№ 8. Задание с единственным выбором. Выберите один правильный ответ.

При разработке системы автоматизации сети ресторанов «Ромашка» необходимо спроектировать доменную модель. Какой из перечисленных вариантов наиболее корректно отражает понятие доменной модели? а) Совокупность экранных форм пользовательского интерфейса, используемых администраторами ресторанов. б) Модель, описывающая ключевые бизнес-сущности (например, Ресторан, Заказ, Склад, Поставка, Курьер, Сотрудник) и правила их взаимодействия в предметной области. в) Набор SQL-запросов для работы с базой данных. г) Структура каталогов и файлов программного проекта.

- a. Набор SQL-запросов для работы с базой данных.
- b. Модель, описывающая ключевые бизнес-сущности (например, Ресторан, Заказ, Склад, Поставка, Курьер, Сотрудник) и правила их взаимодействия в предметной области.
- c. Структура каталогов и файлов программного проекта.
- d. Совокупность экранных форм пользовательского интерфейса, используемых администраторами ресторанов.

№ 9. Задание с единственным выбором. Выберите один правильный ответ.

Какой из перечисленных объектов наиболее целесообразно реализовать с использованием шаблона Singleton?

- a. Запись о товаре на складе.
- b. Объект, представляющий отдельного пользователя системы.
- c. Менеджер конфигурации приложения, обеспечивающий доступ к настройкам

программы.

d. Экземпляр заказа интернет-магазина.

№ 10. Задание с единственным выбором. Выберите один правильный ответ.

Какой из принципов SOLID утверждает, что класс должен иметь только одну причину для изменения?

a. Single Responsibility Principle (SRP)

b. Open/Closed Principle (OCP)

c. Interface Segregation Principle (ISP)

d. Dependency Inversion Principle (DIP)

№ 11. Задание с единственным выбором. Выберите один правильный ответ.

При разработке системы автоматизации сети ресторанов «Ромашка» необходимо передать информацию о заказе из ресторана в центральную систему обработки. Для этого объект «Заказ» преобразуется в формат JSON и отправляется по сети. Какова основная цель сериализации данных в данной ситуации?

a. Оптимизация производительности процессора при обработке заказа.

b. Шифрование данных перед передачей по сети

c. Создание резервной копии базы данных

d. Преобразование объекта в формат, пригодный для хранения или передачи с возможностью последующего восстановления объекта.

№ 12. Задание с единственным выбором. Выберите один правильный ответ.

Команда завершила разработку системы автоматизации сети ресторанов «Ромашка». Перед передачей программного обеспечения заказчику необходимо обеспечить его запуск на сервере ресторана с сохранением всех настроек и возможностью дальнейшего обновления. Что включает в себя процесс развертывания программного приложения?

a. Только компиляцию исходного кода.

b. Установку и настройку программного обеспечения в целевой среде, включая конфигурацию, подключение необходимых компонентов и подготовку приложения к эксплуатации.

c. Исправление ошибок, обнаруженных в процессе разработки.

d. Создание резервной копии исходного кода.

№ 13. Задание с единственным выбором. Выберите один правильный ответ.

Почему внесение изменений в чужой программный код часто оказывается сложнее разработки новой программы?

a. Неизвестны структура программы, логика реализации и взаимосвязи между ее компонентами, поэтому любое изменение может повлиять на уже работающий функционал.

b. Все ответы верны.

c. Чужой код всегда содержит ошибки.

d. Чужой код невозможно изменить без полного переписывания.

№ 14. Задание с единственным выбором. Выберите один правильный ответ.

В системе автоматизации сети ресторанов «Ромашка» администратор ежедневно создает меню для каждого ресторана. В большинстве случаев новое меню отличается от меню предыдущего дня лишь несколькими позициями и ценами. Какое решение наиболее соответствует применению шаблона «Прототип» (Prototype)? а) Каждый раз создавать новое

меню вручную, заполняя все блюда заново. б) Создать копию существующего меню и изменить только те позиции, которые отличаются от предыдущего дня. в) Хранить только одно меню для всех ресторанов. г) Удалять старое меню перед созданием нового.

а. Создать копию существующего меню и изменить только те позиции, которые отличаются от предыдущего дня.

б. Каждый раз создавать новое меню вручную, заполняя все блюда заново.

с. Удалять старое меню перед созданием нового.

д. Хранить только одно меню для всех ресторанов.

№ 15. Задание с единственным выбором. Выберите один правильный ответ.

Можно ли использовать несколько Декораторов для одного объекта?

а. Да

б. Нет

№ 16. Задание с единственным выбором. Выберите один правильный ответ.

Каково основное назначение абстрактного класса в объектно-ориентированном программировании?

а. Создание объекта, экземпляры которого используются непосредственно в программе.

б. Хранение только статических методов и констант.

с. Определение общего интерфейса и базовой реализации для группы связанных классов с последующим наследованием.

д. Обеспечение существования единственного экземпляра класса.

№ 17. Задание с единственным выбором. Выберите один правильный ответ.

При разработке системы автоматизации сети ресторанов «Ромашка» необходимо создавать различные типы отчетов: финансовый отчет, отчет по складу, отчет по продажам и отчет по работе курьерской службы. В дальнейшем планируется добавление новых видов отчетов без изменения существующего кода. Какое решение наиболее соответствует использованию шаблона «Фабрика»?

а. Создавать каждый отчет непосредственно с помощью оператора new в разных частях программы.

б. Использовать один объект отчета и изменять его тип во время выполнения программы.

с. Реализовать класс-фабрику, который отвечает за создание объектов отчетов, а клиентский код работает только с их общим интерфейсом.

д. Создавать все отчеты в одном большом классе с использованием множества условных операторов.

№ 18. Задание с единственным выбором. Выберите один правильный ответ.

Какова основная цель анализа существующего программного кода перед внесением изменений?

а. Удалить неиспользуемые файлы проекта

б. Определить структуру программы, понять взаимосвязи между компонентами и оценить влияние изменений на существующий функционал.

с. Полностью переписать программу с использованием нового языка программирования.

d. Увеличить количество строк программного кода.

№ 19. Задание с единичным выбором. Выберите один правильный ответ.

Что такое CQRS?

a. Это высокоскоростная система передачи данных

b. Это архитектурный шаблон проектирования с разделением ответственности за команды и запросы

№ 20. Задание с единичным выбором. Выберите один правильный ответ.

В системе автоматизации сети ресторанов «Ромашка» центральный сервер получает от ресторана данные о новом заказе в формате JSON. Для дальнейшей обработки необходимо восстановить объект «Заказ» в программе. Какой процесс выполняется в данном случае?

a. Архивация данных.

b. Репликация данных.

c. Компиляция данных.

d. Десериализация данных.

№ 21. Задание с единичным выбором. Выберите один правильный ответ.

В системе автоматизации сети ресторанов «Ромашка» используется собственный модуль логирования, однако было принято решение перейти на стороннюю библиотеку, интерфейс которой отличается от существующего. При этом изменение кода всех модулей системы нежелательно. Какое решение наиболее соответствует применению шаблона «Адаптер» (Adapter)?

a. Отказаться от использования сторонней библиотеки и оставить старую реализацию.

b. Создать класс-адаптер, который преобразует вызовы существующего интерфейса логирования в интерфейс новой библиотеки.

c. Одновременно использовать две независимые системы логирования без их взаимодействия.

d. Полностью переписать все модули системы для работы с новой библиотекой логирования.