ОБЩЕСТВО С ОГРАНИЧЕННОЙ ОТВЕТСТВЕННОСТЬЮ
ОНЛАЙН-ШКОЛА «ТОЧКА ЗНАНИЙ»

УТВЕРЖДЕНА
Приказом Генерального
директора
ООО «Точка знаний»
№ 20 от «25» августа 2026 г.

Дополнительная общеразвивающая программа «Курс по программированию на Python. Ступень 3»

(трудоемкость 121 ак. ч.)

Разработчик:
Гуртовцев Никита Валерьевич
Преподаватель дополнительного образования

Возраст: дети (от 12 лет)
Срок обучения: 9 месяцев

Краснодар, 2026 г.

СОДЕРЖАНИЕ

1. Общая характеристика……... 3
1.1. Пояснительная записка……... 3
1.2. Направленность…….. 3
1.3. Актуальность программы……... 3
1.4. Цели и задачи программы……... 3
1.5. Категория обучающихся…….…… 4
1.6. Форма обучения и сроки освоения…….…… 4
1.7. Форма организации образовательной деятельности…… 4
1.8. Документ, выдаваемый после завершения обучения…… 4
2. Содержание программы…… 5
2.1. Календарный учебный график……... 5
2.2. Учебный план……5
2.3. Рабочая программа…….…….. 7
2.4. Кадровое обеспечение……12
2.5. Материально-техническое обеспечение реализации программы……..…….12
2.6. Форма аттестации и оценочные материалы ……13
2.7. Список рекомендованной литературы……..…….18
2.8. Список использованной литература…….…….18
2.8.1. Электронные ресурсы…….…….19

1. ОБЩАЯ ХАРАКТЕРИСТИКА

1.1. Пояснительная записка

Дополнительная общеобразовательная общеразвивающая программа «Курс по программированию на Python. Ступень 3» (далее – Программа) разработана в соответствии с Федеральным законом от 29.12.2012 № 273-ФЗ «Об образовании в Российской Федерации», Приказом Министерства просвещения РФ от 27.07.2022 г. № 629 «Об утверждении Порядка организации и осуществления образовательной деятельности по дополнительным общеобразовательным программам», Письмом Министерства образования и науки РФ от 18.11.2015 № 09-3242 «О направлении Методических рекомендаций по проектированию дополнительных общеразвивающих программ (включая разно уровневые)».

Программа ориентирована на учащихся, уже владеющих базовыми навыками Python, и направлена на переход от учебных задач к практике, максимально приближенной к реальной разработке. В ходе обучения учащиеся осваивают инструменты Python-разработчика и полный цикл создания мини-продукта: организация проекта и зависимостей, работа с Git, разработка backend-сервиса на FastAPI, подключение базы данных через SQLAlchemy и написание автотестов на Pytest. Отдельное внимание уделяется инженерным практикам: проектированию API, обработке ошибок, валидации данных, интеграции со сторонними сервисами (Requests/JSON), парсингу данных и оптимизации работы сервиса с помощью кеширования и Redis. Итогом становится проект, подготовленный к демонстрации и запуску, с понятной структурой, документацией и базовыми мерами устойчивости и безопасности.

1.2. Направленность: техническая.

1.3. Актуальность программы:

Программа актуальна в условиях роста спроса на прикладные навыки разработки и перехода отрасли к сервисной архитектуре и API-взаимодействию. Современный Python-разработчик должен уметь не только писать код, но и создавать устойчивые сервисы: работать с зависимостями и окружением, вести репозиторий, проектировать API, подключать базы данных, писать тесты и интегрироваться с внешними источниками данных. Освоение этих навыков позволяет учащимся познакомиться с реальными практиками индустрии и подготовиться к выполнению задач уровня стажёра: создание простых веб-сервисов, мини-проектов и интеграций.

1.4. Цели и задачи Программы:

Цель программы: сформировать у учащихся практические навыки Python-разработчика, необходимые для создания и сопровождения небольших backend-сервисов и мини-продуктов, максимально приближенных к реальным задачам разработки.

Задачи программы:

- освоить базовые инструменты разработчика: окружение, зависимости, структура проекта, подготовка к запуску;
- изучить основы работы с Git: репозиторий, ветки, Pull Request, базовый workflow;
- познакомить с принципами backend-разработки и HTTP;
- освоить разработку API-сервисов на FastAPI (эндпоинты, методы HTTP, параметры запросов);
- изучить валидацию данных, обработку ошибок и статус-коды;
- освоить основы тестирования и написание тестов на Pytest, включая тестирование API;
- изучить основы SQL и работу с базами данных через ORM SQLAlchemy (CRUD, связи, запросы);
- освоить миграции схемы базы данных с помощью Alembic;
- научить работать с внешними API (Requests/JSON), обрабатывать сетевые ошибки и лимиты;
- познакомить с парсингом данных и сохранением результатов (в файл или базу данных) при соблюдении этичных практик;
- освоить базовые подходы к оптимизации (кеширование) и работу с Redis;
- сформировать навыки подготовки проекта к релизу (структура, документация, стабильный запуск, базовая безопасность).

1.5. Категория обучающихся: К освоению дополнительной общеобразовательной общеразвивающей программы допускаются учащиеся 7–11-х классов (рекомендуется), в возрасте 12–17 лет, имеющие базовые навыки программирования на Python (условия, циклы, функции, структуры данных) и опыт выполнения учебных проектов.

В результате изучения курса обучающиеся должны

знать:

- основы организации проекта на Python: виртуальное окружение, зависимости, структура каталогов;
- базовые принципы работы с Git (коммиты, ветки, Pull Request);
- базовые понятия backend-разработки, HTTP-методы и статус-коды;
- принципы построения API на FastAPI и правила проектирования эндпоинтов;
- основы валидации данных с помощью Pydantic и обработку ошибок;
- назначение тестирования и базовые приёмы работы с Pytest;
- основы баз данных и SQL, принципы ORM и моделирования данных;
- основы SQLAlchemy, CRUD-операции, типовые запросы и связи;
- назначение миграций и принципы работы Alembic;
- основы интеграций со сторонними сервисами через Requests/JSON;
- базовые принципы кеширования и назначение Redis;
- базовые требования к подготовке проекта к релизу (README, инструкции, проверка запуска).

уметь:

- создавать и поддерживать структуру Python-проекта, управлять зависимостями и запуском;
- работать с Git в базовом командном формате (ветки, PR, история изменений);
- разрабатывать API-сервис на FastAPI (GET/POST/PUT/DELETE), документировать его через Swagger;
- валидировать входные данные, обрабатывать ошибки и корректно возвращать статус-коды;
- писать и запускать автоматические тесты на Pytest, включая тесты API-эндпоинтов;
- проектировать и подключать базу данных к сервису, реализовывать CRUD через SQLAlchemy;
- создавать и применять миграции Alembic для развития схемы без потери данных;
- получать данные из внешних API, обрабатывать исключения и ограничения;
- реализовывать парсер данных и сохранять результаты в файл или базу;
- применять кеширование (в т.ч. с Redis) для ускорения типовых запросов;
- готовить проект к демонстрации/релизу: порядок в коде, понятная документация, стабильный запуск.

1.6. Форма обучения и сроки освоения:

Программа реализуется исключительно с применением дистанционных образовательных технологий и электронного обучения заочной форме.

Сроки освоения программы – **9 месяцев.**

Лекционные занятия онлайн (видеоурок/вебинар) – **60 ак.ч.**

Самостоятельная работа – **60 ак.ч.**

Итоговая аттестация (тестирование) - **1 ак.ч.**

Период обучения и режим занятий:

Занятия проводятся 1 раз в неделю по **1,67 ак.ч.**

1.7. Форма организации образовательной деятельности: групповая.

1.8. Документ, выдаваемый после завершения обучения: сертификат об обучении.

2. СОДЕРЖАНИЕ ПРОГРАММЫ

Программа включает: 36 уроков и итоговый тест.
Перечень, трудоемкость, последовательность и распределение уроков, формы промежуточной аттестации слушателей определяются учебным (тематическим) планом.
Содержание каждого урока включает лекционный и практический материал.

2.1 Календарный учебный график

Общая учебная нагрузка – 36 дней								
1 месяц	2 месяц	3 месяц	4 месяц	5 месяц	6 месяц	7 месяц	8 месяц	9 месяц
ЛЗ, СР, ЕТ	ЛЗ, СР, ЕТ	ЛЗ, СР, ЕТ	ЛЗ, СР, ЕТ	ЛЗ, СР, ЕТ	ЛЗ, СР, ЕТ	ЛЗ, СР, ЕТ	ЛЗ, СР, ЕТ	ЛЗ, СР, ЕТ ИА

ЛЗ – лекционные занятия
СР – самостоятельная работа
ЕТ – ежемесячное тестирование
ИА – итоговая аттестация

2.2. Учебный план

№ п/п	**Название модулей**	**Количество часов**			
		Всего ак.ч.	**Лекционные занятия онлайн (видеоурок/ вебинар) ак.ч.**	**Самостоя-тельная работа ак.ч.**	**Форма проверки**
1	Модуль №1: Инструменты разработчика, Git и структура проекта	14	7	7	ТК, ЕТ
2	Модуль №2: Backend и FastAPI: создание API и документации	26	13	13	ТК, ЕТ

3	Модуль №3: Тестирование: Pytest и проверка API	10	5	5	ТК, ЕТ
4	Модуль №4: Базы данных и ORM: SQLAlchemy + миграции Alembic	30	15	15	ТК, ЕТ
5	Модуль №5: Внешние сервисы и парсинг данных	24	12	12	ТК, ЕТ
	Модуль №6: Оптимизация: кеширование, Redis и базовая безопасность	10	5	5	ТК, ЕТ
	Модуль №7: Подготовка к релизу и итоговая защита	6	3	3	ТК, ЕТ
	Итоговая аттестация	1	0	1	Тестирование
	ИТОГО	**121**	**60**	**61**	

ТК – текущий контроль
ЕТ – ежемесячное тестирование

2.3. Рабочая программа

Модуль №1: Инструменты разработчика, GitHub и структура проекта

Урок №1: Инструменты разработчика Python
Во время урока слушатели изучают: базовые инструменты Python-разработчика, структуру проекта и организацию рабочего процесса, включая запуск проекта и навигацию по кодовой базе. По окончании урока слушатели выполняют домашнее задание.

Урок №2: Введение в Git и GitHub
Во время урока слушатели изучают: назначение системы контроля версий Git и платформы GitHub, создание репозитория и основные команды для фиксации изменений. По окончании урока слушатели выполняют домашнее задание.

Урок №3: Ветки и Pull Request
Во время урока слушатели изучают: работу с ветками, создание Pull Request и базовый командный workflow, включая проверку и объединение изменений. По окончании урока слушатели выполняют домашнее задание.

Урок №4: Виртуальное окружение и зависимости
Во время урока слушатели изучают: создание виртуального окружения, установку и фиксацию зависимостей, а также подготовку проекта к запуску на другом компьютере. По окончании урока слушатели выполняют домашнее задание.

Модуль №2: Backend и FastAPI: создание API и документации

Урок №5: Введение в Backend и HTTP
Во время урока слушатели изучают: основы backend-разработки и принципы HTTP, включая запросы и ответы, структуру URL и базовые понятия клиент-серверного взаимодействия. По окончании урока слушатели выполняют домашнее задание.

Урок №6: Введение в FastAPI
Во время урока слушатели изучают: основы работы с FastAPI, создание первого приложения, запуск сервера и знакомство с автоматической документацией Swagger/OpenAPI. По окончании урока слушатели выполняют домашнее задание.

Урок №7: Маршруты и параметры
Во время урока слушатели изучают: маршруты (эндпоинты), path/query-параметры и передачу данных в запросах, а также структуру обработки запросов в FastAPI. По окончании урока слушатели выполняют домашнее задание.

Урок №8: Pydantic-модели
Во время урока слушатели изучают: валидацию входных данных с помощью Pydantic, создание схем запросов и ответов, а также базовые правила типизации данных. По окончании урока слушатели выполняют домашнее задание.

Урок №9: Методы GET/POST/PUT/DELETE

Во время урока слушатели изучают: назначение HTTP-методов и реализацию CRUD-логики в API, включая передачу данных и обновление ресурсов. По окончании урока слушатели выполняют домашнее задание.

Урок №10: Статус-коды и ошибки
Во время урока слушатели изучают: работу со статус-кодами, обработку ошибок и исключений в API, а также принципы формирования понятных сообщений об ошибках. По окончании урока слушатели выполняют домашнее задание.

Урок №11: Разработка API-сервиса. Часть 1
Во время урока слушатели изучают: проектирование структуры API-сервиса и реализацию базовых эндпоинтов для работы с сущностями проекта. По окончании урока слушатели выполняют домашнее задание.

Урок №12: Разработка API-сервиса. Часть 2
Во время урока слушатели изучают: доработку API-сервиса, расширение функционала, проверку корректности работы эндпоинтов и оформление документации. По окончании урока слушатели выполняют домашнее задание.

Модуль №3: Тестирование: Pytest и проверка API

Урок №13: Введение в тестирование
Во время урока слушатели изучают: зачем нужно тестирование, чем отличаются виды тестов и как тесты помогают поддерживать качество проекта при изменениях кода. По окончании урока слушатели выполняют домашнее задание.

Урок №14: Тестирование с Pytest
Во время урока слушатели изучают: основы Pytest, написание простых тестов, проверки ожидаемых результатов и организацию тестов в проекте. По окончании урока слушатели выполняют домашнее задание.

Урок №15: Тестирование FastAPI
Во время урока слушатели изучают: тестирование API-эндпоинтов с использованием тестового клиента FastAPI, проверку статусов, ответов и валидации данных. По окончании урока слушатели выполняют домашнее задание.

Модуль №4: Базы данных и ORM: SQLAlchemy + миграции Alembic

Урок №16: SQL и базы данных
Во время урока слушатели изучают: базовые понятия баз данных и основы SQL, включая таблицы, поля, ключи и простые запросы выборки данных. По окончании урока слушатели выполняют домашнее задание.

Урок №17: Введение в SQLAlchemy ORM

Во время урока слушатели изучают: принципы ORM-подхода и основы SQLAlchemy, включая описание моделей и подключение к базе данных. По окончании урока слушатели выполняют домашнее задание.

Урок №18: Создание и получение объектов в ORM
Во время урока слушатели изучают: создание записей в базе данных и получение данных через SQLAlchemy, а также базовые операции чтения данных. По окончании урока слушатели выполняют домашнее задание.

Урок №19: Обновление, удаление и валидация данных в ORM
Во время урока слушатели изучают: обновление и удаление данных (CRUD), валидацию входных данных и обработку ошибок при работе с базой. По окончании урока слушатели выполняют домашнее задание.

Урок №20: Работа со связями и запросами
Во время урока слушатели изучают: связи между таблицами, типовые запросы и выборки данных, а также принципы построения запросов для реальных задач. По окончании урока слушатели выполняют домашнее задание.

Урок №21: Миграции схемы: Alembic
Во время урока слушатели изучают: зачем нужны миграции, как создавать и применять миграции Alembic, а также как безопасно развивать схему базы данных. По окончании урока слушатели выполняют домашнее задание.

Урок №22: Интеграция FastAPI + SQLAlchemy
Во время урока слушатели изучают: подключение базы данных к FastAPI-приложению, организацию слоёв проекта и реализацию CRUD-эндпоинтов с использованием ORM. По окончании урока слушатели выполняют домашнее задание.

Урок №23: Разработка проекта API с базой данных. Часть 1
Во время урока слушатели изучают: проектирование структуры приложения с БД, создание моделей и ключевых эндпоинтов, а также настройку миграций. По окончании урока слушатели выполняют домашнее задание.

Урок №24: Разработка проекта API с базой данных. Часть 2
Во время урока слушатели изучают: расширение функционала проекта, добавление связей и запросов, а также проверку корректности работы API с базой данных. По окончании урока слушатели выполняют домашнее задание.

Модуль №5: Внешние сервисы и парсинг данных

Урок №25: Requests и внешние API
Во время урока слушатели изучают: работу с HTTP-запросами через requests, получение данных и базовую обработку ответов веб-сервисов. По окончании урока слушатели выполняют домашнее задание.

Урок №26: Ошибки сети и лимиты
Во время урока слушатели изучают: обработку сетевых ошибок, таймаутов и ограничений API, а также принципы устойчивой работы с внешними сервисами. По окончании урока слушатели выполняют домашнее задание.

Урок №27: Интеграция внешнего API в FastAPI
Во время урока слушатели изучают: встраивание внешних данных в backend-сервис, построение эндпоинтов-агрегаторов и корректную обработку ошибок внешнего API. По окончании урока слушатели выполняют домашнее задание.

Урок №28: Парсинг данных
Во время урока слушатели изучают: основы парсинга, извлечение данных со страниц и базовые правила корректного и этичного получения информации. По окончании урока слушатели выполняют домашнее задание.

Урок №29: Сохранение результатов
Во время урока слушатели изучают: сохранение результатов обработки данных в файл или базу данных, а также организацию структуры данных для хранения. По окончании урока слушатели выполняют домашнее задание.

Урок №30: Разработка парсера. Часть 1
Во время урока слушатели изучают: проектирование парсера и реализацию базового этапа извлечения данных с обработкой типовых ошибок. По окончании урока слушатели выполняют домашнее задание.

Урок №31: Разработка парсера. Часть 2
Во время урока слушатели изучают: доработку парсера, улучшение устойчивости, сохранение результатов и подготовку парсинга к регулярному запуску. По окончании урока слушатели выполняют домашнее задание.

Модуль №6: Оптимизация: кеширование, Redis и базовая безопасность

Урок №32: Кеширование
Во время урока слушатели изучают: зачем нужно кеширование, какие задачи оно решает и как применять кеширование в API-сервисах для ускорения ответов. По окончании урока слушатели выполняют домашнее задание.

Урок №33: Redis
Во время урока слушатели изучают: основы Redis и его применение для хранения временных данных и кеша, а также подключение Redis к проекту. По окончании урока слушатели выполняют домашнее задание.

Урок №34: Оптимизация и безопасность
Во время урока слушатели изучают: базовые приёмы оптимизации сервиса и основные меры безопасности учебного проекта, включая устойчивость к ошибкам и корректную обработку данных. По окончании урока слушатели выполняют домашнее задание.

Модуль №7: Подготовка к релизу и итоговая защита

Урок №35: Подготовка к релизу
Во время урока слушатели изучают: финализацию проекта перед релизом, приведение структуры и зависимостей в порядок, обновление документации и проверку запуска. По окончании урока слушатели выполняют домашнее задание.

Урок №36: Повторение и подведение итогов
Во время урока слушатели изучают: повторение ключевых тем курса, разбор типовых ошибок, финальная проверка проекта и подготовка к защите результата. По окончании урока слушатели выполняют домашнее задание.

Итоговая аттестация
Тестирование

2.4. Кадровое обеспечение

Реализация программы обеспечивается педагогическими кадрами, имеющими высшее профессиональное образование и отвечающими квалификационным требованиям, указанным в квалификационных справочниках, и (или) профессиональным стандартам, в рамках изучаемого цикла.

2.5. Материально-техническое и программное обеспечение реализации программы

ООО «ТОЧКА ЗНАНИЙ» располагает материально-технической базой, обеспечивающей проведение всех видов дисциплинарной и междисциплинарной подготовки, итоговой аттестации слушателей, предусмотренных учебным планом:

Место работы преподавателя	✔ Стол компьютерный/письменный 120х60 - 1шт; ✔ Кресло ARDOR GAMING Gun Shield 10B - 1шт; ✔ ИБП Dexp IEC Plus LCD 3000VA - 1шт; ✔ Переходник от ИБП к сетевому фильтру - 1 шт; ✔ Сетевой фильтр с USB разъемами - 1шт; ✔ Софтбокс 50х70 600ВТ - 1шт; ✔ Лампа студийная для софтбокса 135W E27 5500K, Холодный белый свет, 135 Вт, Люминесцентная (энергосберегающая) - 4шт (запасные); ✔ Предохранители в софтбокс 3- 5А, 250V - 5 штук; ✔ Монитор 21.45" DEXP DF22N2 черный - 1шт; ✔ Микрофон петличный BOYA BY-M1 - 1шт; ✔ Apple iPad 10.2" (2021) Wi-Fi 64Gb Silver - 1шт; ✔ Apple MacBook Pro 16", M1, 16ГБ, 2021г - 1шт; ✔ Стилус WiWU Pencil Pro - 1шт; ✔ Хаб Baseus (4K@60Hz 8-in-1 STARJOY 8-PORT TYPE-C) - 1шт; ✔ Патч-корд RJ 45 - 1шт; ✔ Футболки (мерч) “Точка знаний” - 2 шт.

Программное обеспечение: лицензионные системные программы, обеспечивающие взаимодействие всех других программ с оборудованием и взаимодействие пользователя персонального компьютера с программами. Универсальные офисные прикладные программы и средства ИКТ, например, программа подготовки презентаций; использование Интернета, электронной почты; использование автоматизированных поисковых систем Интернета.

Информационно-образовательная среда включает в себя образовательную LMS «Точка Знаний».

Образовательная LMS «Точка Знаний» обеспечивает через Интернет доступ к:

- электронным информационным и образовательным ресурсам ООО «ТОЧКА ЗНАНИЙ»;
- доступ к нормативным и организационно-методическим документам, регламентирующим образовательный процесс в ООО «ТОЧКА ЗНАНИЙ»;

- систему электронного учёта слушателей; (см. Положение)
- взаимодействие слушателей с преподавателями, организаторами образовательного процесса и администрацией ООО «ТОЧКА ЗНАНИЙ». (см. Положение)

2.6. Форма аттестации и оценочные материалы

Программой предусмотрен текущий контроль в виде домашнего задания и тестов, размещенных в уроках на образовательной LMS «Точка Знаний», промежуточный контроль в виде тестов, согласно учебному плану.

Итоговая аттестация проводится в виде теста. Итоговый тест включает в себя 20 вопросов. Для успешного прохождения теста необходимо правильно выполнить не менее 85 % заданий.
Обучающиеся допускаются к итоговой аттестации после изучения Модулей программы в объеме, предусмотренном учебным планом программы.

Оценка качества освоения учебной программы проводится в процессе итоговой аттестации в форме тестирования.

Оценка	Критерии
Зачтено	Оценка «Зачтено» выставляется обучающемуся, продемонстрировавшему твердое и всесторонние знания материалы, умение применять полученные в рамках занятий практические навыки и умения. Достижения за период обучения и результаты текущей аттестации демонстрировали отличный уровень знаний и умений обучающегося. Не менее 85% правильных ответов при решении теста.
Не зачтено	Оценка «Не зачтено» выставляется обучающемуся, который в недостаточной мере овладел теоретическим материалом по дисциплине, допустил ряд грубых ошибок при выполнении практических заданий, а также не выполнил требований, предъявляемых к промежуточной аттестации. Достижения за период обучения и результаты текущей аттестации демонстрировали неудовлетворительный уровень знаний и умений обучающегося. Менее 80% правильных ответов при решении теста.

Примеры оценочных материалов текущего контроля

Домашнее задание 12. Разработка API-сервиса. Часть 2
Задание
Нужно разработать мини-API на FastAPI для работы со списком задач (to-do) в памяти (без базы данных). API должен поддерживать создание задачи, получение списка задач, получение одной задачи по id, обновление задачи и удаление задачи.

Пошаговая инструкция

1. Создай проект. Создай папку проекта и установи зависимости FastAPI и Uvicorn.
2. Создай файл main.py и настрой приложение FastAPI. Программа должна запускаться командой uvicorn и открывать документацию Swagger.
3. Опиши модель данных через Pydantic. Создай модель TaskCreate (для создания) и Task (для ответа). Поля: id (int), title (str), done (bool).
4. Реализуй эндпоинты:

- GET /tasks — вернуть список задач
- GET /tasks/{task_id} — вернуть одну задачу по id (если нет — вернуть ошибку 404)
- POST /tasks — создать задачу и вернуть её с id
- PUT /tasks/{task_id} — обновить задачу (title/done)
- DELETE /tasks/{task_id} — удалить задачу

1. Добавь обработку ошибок и статус-коды. Если задача не найдена — возвращай 404 с понятным сообщением.
2. Проверь работу API. Открой Swagger, создай 2–3 задачи, обнови одну, удали одну и убедись, что ответы корректные.

Важно! В каждом ответе API должен возвращать данные строго в формате, который описан в Pydantic-модели.

Примеры оценочных материалов промежуточного контроля

Вопрос 1. Какая команда создаёт виртуальное окружение в Python (Windows/Linux/macOS)?
Ответы:
python -m venv venv
pip install venv
venv create
python venv install
git init venv

Вопрос 2. Что делает команда git clone?
Ответы:
Создаёт новый коммит
Копирует удалённый репозиторий на компьютер
Удаляет ветку
Объединяет ветки
Публикует изменения на GitHub

Вопрос 3. Для чего используется Pull Request (PR) в GitHub?
Ответы:
Чтобы удалить репозиторий
Чтобы предложить изменения и попросить их проверить перед слиянием
Чтобы запустить FastAPI
Чтобы установить зависимости
Чтобы создать виртуальное окружение

Вопрос 4. Какой HTTP-метод обычно используется для получения данных?
Ответы:
GET
POST
PUT
DELETE
PATCH

Вопрос 5. Какой объект создаётся при объявлении app = FastAPI()?
Ответы:
Подключение к базе данных
Приложение FastAPI
Кэш Redis
Тестовый клиент Pytest
Модель Pydantic

Вопрос 6. Для чего используются Pydantic-модели в FastAPI?
Ответы:

Для рисования интерфейса
Для валидации и описания структуры данных запросов/ответов
Для подключения Redis
Для создания веток Git
Для запуска тестов

Вопрос 7. Какой статус-код чаще всего возвращают при успешном создании ресурса?
Ответы:
200
201
204
400
404

Вопрос 8. Какая библиотека обычно используется для тестирования в курсе?
Ответы:
unittest2
pytest
selenium
requests
pydantic

Вопрос 9. Что такое ORM (на примере SQLAlchemy)?
Ответы:
Инструмент для рисования графиков
Способ работать с БД через классы и объекты вместо “сырого” SQL
Модуль для работы с HTTP
Система контроля версий
Инструмент для кеширования

Вопрос 10. Для чего используются миграции Alembic?
Ответы:
Чтобы ускорить интернет
Чтобы менять схему базы данных контролируемо, без потери данных
Чтобы запускать тесты
Чтобы создавать изображения
Чтобы хранить пароли

Примеры оценочных материалов итоговой аттестации

Задание 1. Какой файл чаще всего используется для фиксации зависимостей проекта?
Ответы:
requirements.txt
readme.md
main.py
.gitignore
config.json

Задание 2. Какая команда отправляет изменения в удалённый репозиторий?
Ответы:
git push
git pull

git merge
git status
git branch

Задание 3. Какой HTTP-метод обычно используется для создания ресурса?
Ответы:
GET
POST
PUT
DELETE
OPTIONS

Задание 4. Какой декоратор используется в FastAPI для обработки GET-запроса?
Ответы:
@app.get("/path")
@app.route("GET")
@fastapi.get("/path")
@get("/path")
@app.http("get")

Задание 5. Какой объект чаще всего возвращает requests.get(url).json()?
Ответы:
JSON-данные, преобразованные в dict/list Python
Файл на диске
SQL-таблицу
Git-ветку
Redis-ключ

Задание 6. Что делает команда alembic upgrade head?
Ответы:
Откатывает миграции
Применяет все миграции до последней
Удаляет базу данных
Создаёт новую миграцию
Запускает FastAPI

Задание 7. Какой статус-код означает «не найдено»?
Ответы:
200
201
204
404
500

Задание 8. Для чего в API обычно используют кеширование?
Ответы:
Чтобы увеличить размер базы данных
Чтобы ускорить ответы для часто запрашиваемых данных
Чтобы заменить тестирование
Чтобы не писать документацию
Чтобы создавать новые ветки Git

Задание 9. Какой инструмент используется для хранения кеша/временных данных в курсе?

Ответы:
SQLite
Redis
Swagger
Pytest
Pydantic

Задание 10. Какой тест чаще всего проверяет работу эндпоинта FastAPI?
Ответы:
Тест, который делает запрос к эндпоинту через тестовый клиент и проверяет статус/ответ
Тест, который рисует диаграмму
Тест, который создаёт миграцию
Тест, который делает git push
Тест, который меняет README

Задание 11. Какой из вариантов корректно создаёт SQLAlchemy-модель (упрощённо)?
Ответы:
class User(Base): tablename = "users"
def User(Base): table = "users"
model User: table users
User = Table("users") without Base
create User(Base) with name

Задание 12. Какой из вариантов корректно описывает цель проекта на курсе?
Ответы:
Создать backend-сервис на FastAPI с базой данных и тестами
Сделать только презентацию
Собрать игру на Pygame
Нарисовать интерфейс на Tkinter
Написать один скрипт без зависимостей

Задание 13. Что такое CRUD?
Ответы:
Create, Read, Update, Delete
Copy, Run, Upload, Deploy
Code, Review, Update, Debug
Create, Remove, Use, Download
Cache, Redis, Uvicorn, Docker

Задание 14. Какая команда запускает приложение FastAPI через Uvicorn (пример)?
Ответы:
uvicorn main:app --reload
fastapi run main
python fastapi start
pytest main:app
git run server

Задание 15. Какой правильный подход при сетевой ошибке внешнего API?
Ответы:
Игнорировать ошибку и всегда возвращать 200
Обрабатывать исключение, возвращать понятную ошибку и корректный статус-код
Удалять базу данных
Делать git reset

Отключать Pydantic

Задание 16. Что обычно добавляют в README проекта перед релизом?
Ответы:
Инструкцию по установке и запуску проекта
Только картинки
Только фамилию автора
Список всех ошибок
Ничего не добавляют

Задание 17. Какой принцип лучше всего описывает назначение тестов?
Ответы:
Тесты нужны только для оценки в школе
Тесты помогают быстро проверять, что изменения не сломали проект
Тесты заменяют документацию
Тесты нужны только для работы с Redis
Тесты используются вместо базы данных

Задание 18. Какая из задач относится к подготовке к релизу?
Ответы:
Привести структуру проекта в порядок и зафиксировать зависимости
Удалить репозиторий
Запретить Swagger
Убрать обработку ошибок
Отключить тесты

Задание 19. Что нужно сделать после прохождения итогового тестирования?
Ответы:
Ничего
Подготовить проект и сдать итоговый результат (репозиторий + инструкция)
Удалить все ветки
Сбросить миграции
Отключить интернет

Задание 20. Какой элемент минимально обязателен для итогового проекта курса?
Ответы:
API-сервис на FastAPI с эндпоинтами
Только Excel-таблица
Только рисунок в Canva
Только текстовый документ
Только Telegram-бот без API

2.7. Список рекомендованной литературы

Основная:

1. Документация Python https://www.python.org/doc/
2. Руководство по началу работы с PyCharm https://www.jetbrains.com/help/pycharm/getting-started.html

Дополнительная:

1. Документация Python 3 https://docs.python.org/3/

2.8. Список использованной литературы

2.8.1. Электронные ресурсы

1. Электронная библиотека https://elibrary.ru/
2. Онлайнтестпад-онлайн конструктор тестов https://onlinetestpad.com/
3. PyCharm (интегрированная среда разработки) https://www.jetbrains.com/pycharm/