ЦЕНТРАЛЬНЫЙ УНИВЕРСИТЕТ

УТВЕРЖДЕНА

Решением Ученого совета
АНО ВО «Центральный университет»
«24» июня 2025 г.
Протокол №2

Рабочая программа дисциплины (модуля)
«Разработка на Python. Профессиональный»

Направление подготовки: *02.03.01 Математика и компьютерные науки*

Направленность (профиль) подготовки: *Программа двух дипломов НИУ ВШЭ и ЦУ «Прикладная математика и информатика»*

Квалификация (степень) выпускника: *бакалавр*

Форма обучения: *очная*

Срок освоения программы: *4 года*

Год набора: *2025*

Москва
2025

Электронный документ

Содержание

1. Краткая характеристика дисциплины (модуля)

Рабочая программа дисциплины (модуля) «Разработка на Python. Профессиональный» составлена в соответствии с федеральным государственным образовательным стандартом высшего образования – бакалавриат по специальности 02.03.01 Математика и компьютерные науки, профиль «Программа двух дипломов НИУ ВШЭ и ЦУ «Прикладная математика и информатика», утвержденный приказом Министерства науки и высшего образования Российской Федерации № 807 от 23.08.2017 года.

Изучение дисциплины (модуля) дает умение программирования, используемым в различных областях, таких как веб-разработка, анализ данных и машинное обучение. Освоение Python позволяет эффективно решать сложные задачи в профессиональной деятельности.

Место дисциплины (модуля) в структуре образовательной программы

Настоящая дисциплина (модуль) включена в учебный план по программе подготовки бакалавриата по направлению 02.03.01 Математика и компьютерные науки, профиль «Программа двух дипломов НИУ ВШЭ и ЦУ «Прикладная математика и информатика» и входит в обязательную часть Блока 1, как дисциплина по выбору.

Дисциплина (модуль) изучается на 1 курсе в 1 семестре.

Дисциплина (модуль) «Разработка на Python» включает три уровня: основной, углубленный и профессиональный, которые соответствуют навыкам и подготовке обучающихся; уровни определяются с помощью входного тестирования, по итогам которого обучающиеся распределяются на соответствующие уровни.

«Профессиональный» уровень подходит для обучающихся, знакомых с базовыми концептами веб-разработки на Python и планирующих стать разработчиками.

Цель изучения дисциплины (модуля): формирование у студентов навыков программирования на языке Python, включая разработку, отладку и оптимизацию приложений.

Задачи изучения дисциплины (модуля):

— освоение основ языка Python и принципы структурирования кода;
— понимание базовых принципов анализа и визуализации данных;
— структурирование решений с использованием функций и классов;
— выполнение базовых обработок и анализ данных;

В результате освоения дисциплины (модуля), обучающийся должен:

знать:

— принципы работы клиент-серверной архитектуры: роль фронтенда и бэкенда;
— основные команды Git: создание репозиториев, ветвление, коммиты, слияние и разрешение конфликтов;
— основы работы с GitLab: создание проектов, управление ветками, MR;
— принципы работы HTTP: методы запросов (GET, POST, PUT, DELETE), структура запроса и ответа;
— основные возможности FastAPI: асинхронные маршруты, работа с запросами, возвращение ответов;
— основы асинхронного программирования: что такое async/await, задачи асинхронности;
— что такое middleware, их назначение и основные примеры использования в FastAPI;
— основы кеширования: зачем оно нужно, типы кеширования (в памяти, распределенное), как кеширование влияет на производительность;
— основные принципы логирования: уровни логов, структурированное логирование, назначение логов в разработке;
— что такое периодические задачи и как их использовать для автоматизации

процессов;

— основы работы с планировщиком и FastAPI BackgroundTasks для реализации периодических задач;

уметь:

— работать с Git для управления версионностью проекта: выполнять коммиты, переключаться между ветками, работать с удаленными репозиториями;

— настраивать проект на FastAPI: создавать основные маршруты, разделять файлы по назначению;

— взаимодействовать с внешними API через HTTP-запросы с использованием библиотеки httpx;

— обрабатывать данные из сторонних API, передавать их фронтенду через маршруты FastAPI;

— использовать асинхронные функции для работы с длительными операциями;

— реализовывать middleware в FastAPI;

— уастраивать кеширование с использованием Redis для повышения производительности приложений;

— использовать loguru для добавления логирования в приложение;

— настраивать формат вывода логов;

— управлять уровнями логирования;

— логировать исключения и критические события;

— создавать периодические задачи с использованием apscheduler и BackgroundTasks FastAPI;

— автоматизировать выполнение фоновых операций;

владеть:

— созданием и ведением проект в GitLab;

— разработкой проектов в команде с использованием Git: создавать MR, проводить код-ревью, разрешать конфликты при слиянии;

— написанием бэкенд для готового фронтенда, обеспечив корректное взаимодействие клиентской и серверной частей;

— организацией проектной структуры для комплексного FastAPI-приложения, разделяя логику API, бизнес-логику и обработку данных;

— реализацией взаимодействия с несколькими сторонними API, комбинируя их данные в одном сервисе;

— интеграцией асинхронную обработку данных в проект, обеспечивая стабильность и производительность приложения;

— разработкой middleware;

— настройкой кеширования с продвинутыми функциями: инвалидация кеша, настройка TTL;

— организацией централизованное структурированное логирование с использованием loguru;

— реализовать комплексные периодические задачи, используя комбинацию планировщика и FastAPI BackgroundTasks, обеспечивая стабильную и предсказуемую работу сервиса;

— связать результаты логирования, кеширования и периодических задач для мониторинга и улучшения производительности приложения.

2. Перечень планируемых результатов обучения

Компетенции, формируемые в результате освоения дисциплины (модуля) при проведении учебных занятий в форме контактной работы обучающихся с педагогическими работниками Университета и в форме самостоятельной работы обучающихся:

Компетенция	Содержание компетенции	Индикатор компетенции	Перечень планируемых результатов обучения по дисциплине (модулю)
ОПК-6.	Способен разрабатывать алгоритмы и компьютерные программы, пригодные для практического применения	ОПК-6.1.	Знает алгоритмы разработки, компьютерные программы, а также алгоритмы вычислительной математики в области искусственного интеллекта
		ОПК-6.2.	Умеет разрабатывать математические программные продукты и комплексы с использованием современных технологий программирования в области искусственного интеллекта
		ОПК-6.3.	Имеет практический опыт разработки интеллектуальных информационных систем для визуализации результатов исследований в области искусственного интеллекта
ПК-3.	Способен применять методы математического и алгоритмического моделирования для решения как теоретических, так и практических задач в рамках профессиональной деятельности	ПК-3.1.	Знает основные методы математического и алгоритмического моделирования, а также их применение для решения теоретических и прикладных задач
		ПК-3.2.	Умеет применять методы математического и алгоритмического моделирования для анализа и решения различных задач в области математики и компьютерных наук
		ПК-3.3.	Имеет опыт использования методов математического и алгоритмического моделирования при решении теоретических и прикладных задач в профессиональной деятельности
ПК-4.	Способен разрабатывать программное обеспечение для решения прикладных задач в области искусственного интеллекта под руководством более опытного специалиста	ПК-4.1.	Знает основные принципы разработки программного обеспечения и методы решения прикладных задач в сфере искусственного интеллекта
		ПК-4.2.	Умеет разрабатывать программное обеспечение под руководством специалистов

			более высокой категории, используя современные технологии и инструменты
		ПК-4.3.	Имеет опыт участия в разработке программного обеспечения для решения прикладных задач в команде под руководством более опытных специалистов

3. Тематический план

№п/п	Наименование раздела дисциплины (модуля)	Трудоемкость, академические часы					ТКУ (текущий контроль успеваемости)
		Очная форма					
		Контактная работа			Контроль	Самостоятельная работа	
		Лекции	Семинары (практические занятия)	Консультации			
1	Базовые инструменты и взаимодействие с сервисами	8	8	4		40	Проект
2	Проектирование веб-приложений на FastAPI	8	8	6	4	40	Коллоквиум Проект
3	Инженерные практики и промышленная разработка	8	8	4	4	40	Коллоквиум Проект
	Зачет с оценкой						
	Итого:	***24***	***24***	***14***	***8***	***120***	
Объем дисциплины (модуля) (в ак. ч.)		***190***					
Объем дисциплины (модуля) (в зач. ед.)		***5***					

4. Содержание дисциплины (модуля)

№п/п	Наименование раздела дисциплины (модуля)	Содержание дисциплины (модуля) по темам
1	Базовые инструменты и взаимодействие с сервисами	Git и GitLab. Основы HTTP и работа с API
2	Проектирование веб-приложений на FastAPI	FastAPI - организация проекта и работа с HTML-шаблонами. Асинхронные маршруты и обработка запросов. MongoDB. Взаимодействие с внешними API через асинхронные библиотеки
3	Инженерные практики и промышленная разработка	Middleware. Основы кеширования и использование Redis. Продвинутое кеширование и инвалидация кеша. Основы логирования и использование loging. Периодические задачи в FastAPI. Dependency Injection

5. Учебно-методическое обеспечение

Университет располагает полным набором лицензионного и свободно распространяемого программного обеспечения, включая продукты отечественного производства.

Каждый студент в течение всего периода обучения получает индивидуальный неограниченный доступ к электронно-библиотечной системе и электронной информационно-образовательной среде университета. Эти системы предоставляют возможность доступа к ресурсам из любой точки, где есть подключение к сети Интернет, как на территории университета, так и за его пределами.

Студентам обеспечен удаленный доступ к современным профессиональным базам данных и информационным справочным системам.

Основная литература:

1. Чернышев, С. А. Основы программирования на Python : учебник для вузов / С. А. Чернышев. — 2-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2025. — 349 с. — (Высшее образование). — ISBN 978-5-534-17139-6. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/567821.

2. Гаско, Р. Простой Python просто с нуля / Р. Гаско ; под ред. Н. Ю. Комлева. - Москва : СОЛОН-ПРЕСС, 2023. - 256 с. - (Серия «Программирование»). - ISBN 978-5-91359-334-4. - Текст : электронный. - URL: https://znanium.ru/catalog/product/2185854.

3. Ван Хорн II, Б. М. PyCharm: профессиональная работа на Python : практическое руководство / Б. М. Ван Хорн II, К. Нгуен ; пер. с англ. И. Л. Люско. – Москва : ДМК Пресс, 2024. - 620 с. – ISBN 978-5-93700-274-7. - Текст : электронный. - URL: https://znanium.ru/catalog/product/2205063.

Дополнительная литература:

1. Гниденко, И. Г. Технологии и методы программирования : учебник для вузов / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. — 2-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2025. — 241 с. — (Высшее образование). — ISBN 978-5-534-18130-2. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/581329.

2. Ричардс, Т. Streamlit для Data Science. Создаем интерактивные приложения в Python : практическое руководство / Т. Ричардс, А. Груздев ; пер. с англ. А. В. Груздева. – Москва : ДМК Пресс, 2024. - 356 с. – ISBN 978-5-93700-275-4. - Текст : электронный. - URL: https://znanium.ru/catalog/product/2205064.

6. Материально-техническое обеспечение

Университет располагает материально-технической базой, соответствующей действующим противопожарным правилам и нормам и обеспечивающей проведение всех видов дисциплинарной и междисциплинарной подготовки, практической и научно-исследовательской работ обучающихся, предусмотренных учебным планом.

Помещения, которые представляют собой учебные аудитории для проведения занятий лекционного типа, занятий семинарского (практического) типа, групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации, а также помещения для самостоятельной работы и помещения для хранения и профилактического обслуживания учебного оборудования. Помещения укомплектованы специализированной мебелью и техническими средствами обучения, служащими для представления учебной информации большой аудитории.

Изучение дисциплины (модуля) обеспечивается в учебных аудиториях, оснащенных:

— столами и стульями;

— компьютерной техникой;
— механическими калькуляторами;
— специализированным оборудованием, включая демонстрационное оборудование.

Помещения для самостоятельной работы обучающихся, в том числе приспособленные для использования инвалидами и лицами с ограниченными возможностями здоровья, оснащены компьютерной техникой с возможностью подключения к сети «Интернет» и обеспечением доступа в электронную информационно-образовательную среду Университета.

Обучающимся предоставляется доступ (в том числе удаленный) к ресурсам информационно-телекоммуникационной сети «Интернет», электронным ресурсам (в том числе электронным библиотечным системам, современным профессиональным базам данных и информационным справочным системам):

№	Наименование портала (издания, курса, документа)	Ссылка
1.	Научная электронная библиотека elibrary.ru библиотека	https://elibrary.ru/defaultx.asp
2.	База данных для IT-специалистов	https://habr.com
3.	База данных ScienceDirect	https://www.sciencedirect.com
4.	Официальный сайт Министерства науки и высшего образования Российской Федерации	https://minobrnauki.gov.ru/
5.	Федеральный портал «Российское образование"	https://www.edu.ru/
6.	Информационная система "Единое окно доступа к образовательным ресурсам"	http://window.edu.ru/
7.	Единая коллекция цифровых образовательных ресурсов	http://school-collection.edu.ru/
8.	Федеральный центр информационно - образовательных ресурсов	http://fcior.edu.ru/

Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине (модулю), в том числе комплект лицензионного программного обеспечения, современные профессиональные базы данных и информационные справочные системы:

Наименование ПО	Производство	Лицензионное / свободно распространяемое
Операционные системы:		
Microsoft Imagine (Windows Client, Server)	зарубежное	лицензионное
Браузеры:		
Яндекс.Браузер	отечественное	свободно распространяемое
Google Chrome	зарубежное	свободно распространяемое
Офисные приложения:		
Microsoft Imagine (Visio, OneNote)	зарубежное	лицензионное
TeXstudio	зарубежное	свободно распространяемое
Adobe Acrobat Reader	зарубежное	свободно распространяемое
Программное обеспечение для планирования и учета времени:		
Toggle app	зарубежное	свободно распространяемое
Системы управления проектами:		
Microsoft Imagine (Project)	зарубежное	лицензионное
Системы управления базами данных:		
Microsoft Imagine (SQL Server)	зарубежное	лицензионное
Системы резервного копирования (backup):		
Acronis Backup Advanced for HyperV	зарубежное	лицензионное

Справочно-правовые системы:		
КонсультантПлюс: справочно-правовая система	отечественное	лицензионное
Средства антивирусной защиты:		
Kaspersky Endpoint Security для бизнеса Стандартный Russian Edition	отечественное	лицензионное
Среды разработки:		
Visual Studio Code	зарубежное	свободно распространяемое
Bash (Unix shell)	зарубежное	свободно распространяемое
Anaconda	зарубежное	свободно распространяемое
Robotic Operating System	зарубежное	свободно распространяемое
CopelliaSim	зарубежное	свободно распространяемое
Google Colaboratory	зарубежное	свободно распространяемое
Пакеты программных средств и библиотек:		
AutoPsy	зарубежное	свободно распространяемое
Interactive Disassembler (IDA)	зарубежное	свободно распространяемое
Системы управления библиографической информацией:		
Zotero	зарубежное	свободно распространяемое
Сервисы и службы:		
Bind	зарубежное	свободно распространяемое
Docker	зарубежное	свободно распространяемое

7. Методические и оценочные материалы

Методические указания для обучающихся по освоению дисциплины (модуля)

В процессе изучения дисциплины (модуля) «Разработка на Python. Профессиональный» в рамках текущего контроля успеваемости используются такие виды учебной работы, как лекции, семинары, консультации, коллоквиумы и проекты, а также различные виды самостоятельной работы обучающихся по заданию преподавателя, направленные на развитие навыков профессиональной лексики, закрепление практических профессиональных компетенций, поощрение инициатив.

Лекция – систематическое, последовательное, монологическое изложение преподавателем учебного материала, как правило, теоретического характера.

В процессе лекций рекомендуется вести конспект лекций: кратко и схематично фиксировать основные идеи, выводы и обобщения лекции; выделять важные мысли, ключевые слова и термины. Необходимо отметить вопросы или материалы, которые вызывают затруднения, и попытаться найти ответы в рекомендованной литературе. Если разобраться в материале не удается, следует сформулировать вопрос и задать его преподавателю на консультации или во время семинарского (практического) занятия.

Участие в семинаре – активная работа студента на семинаре, его ответы на вопросы преподавателя и участие в дискуссии.

Для успешного участия в семинаре студентам рекомендуется заранее ознакомиться с темой обсуждения, прочитать необходимые материалы и подготовить вопросы. Важно активно слушать и вовлекаться в дискуссию, высказывая свои мнения и аргументируя их. При ответах на вопросы преподавателя стоит быть уверенным, четким и логичным, опираясь на изученный материал. Также полезно поддерживать диалог с однокурсниками, чтобы обогатить обсуждение и расширить свои знания.

Консультации – структурированные встречи, на которых преподаватели предоставляют индивидуальную или групповую помощь в освоении учебного материала, обсуждении вопросов и решении проблем, возникающих в процессе обучения.

Консультации могут включать разъяснение сложных тем, подготовку к экзаменам и

помощь в выполнении проектных работ, что способствует более глубокому пониманию предмета и улучшению академической успеваемости.

Коллоквиум – устные ответы на вопросы, список которых известен студенту заранее.

В процессе подготовки к коллоквиуму необходимо проанализировать учебные материалы, ознакомившись с лекциями, учебниками и дополнительными источниками, акцентируя внимание на ключевых темах. Рекомендуется создать структурированные конспекты, выделяя основные идеи, термины и формулы.

Проект – это целенаправленная деятельность, имеющая определенные цели, задачи и временные рамки, в результате которой создается уникальный продукт или услуга.

Для успешной подготовки проекта рекомендуется следует выполнять следующие рекомендации:

- четко определите цель и задачи проекта, чтобы понимать, какой результат вы хотите достичь;
- составьте план работы, разбив проект на этапы с указанием сроков выполнения каждого из них;
- используйте разнообразные источники информации и инструменты для исследования темы, чтобы обеспечить качественную основу для вашего проекта;
- регулярно проверяйте прогресс и вносите коррективы в план, если это необходимо, чтобы оставаться на правильном пути к завершению проекта.

Самостоятельная работа – работа студентов, направленная на углубленное изучение отдельных тем и вопросов учебной дисциплины (модуля).

В процессе самостоятельной работы студенты взаимодействуют с рекомендованными материалами при минимальном участии преподавателя. Задачи студента включают работу с конспектами лекций (обработка текста), повторное изучение учебных материалов планов и тезисов ответов, изучение дополнительных тем, выполнение учебно-исследовательских заданий и другое.

Бонусные баллы — это оценки, которые студенты могут получить за выполнение дополнительных заданий.

Формат бонусных баллов позволяет студентам улучшить общую оценку по курсу и стимулирует углубленное изучение материала.

Система оценивания результатов обучения по дисциплине (модулю)

Критерии получения уровня и оценивания сформированности компетенций по дисциплине «Разработка на Python. Профессиональный»

Оценивание уровня учебных достижений, обучающихся по дисциплине (модулю), осуществляется в виде текущего контроля успеваемости и промежуточной аттестации.

Промежуточная аттестация по дисциплине (модулю) осуществляется в форме ***зачета с оценкой***, при этом проводится оценка компетенций, сформированных по дисциплине.

Для оценивания текущего контроля успеваемости и промежуточной аттестации используется десятибалльная шкала оценивания, которая соотносится с традиционной пятибалльной шкалой следующим образом:

Десятибалльная оценка	Пятибалльная оценка	Оценка за зачет	Общая характеристика результата обучения по дисциплине (модулю)
10	Отлично	Зачтено	Студент полностью владеет знаниями, изложенными в рабочей программе, и глубоко осмысляет дисциплину. Он самостоятельно и логически последовательно отвечает на все вопросы, акцентируя внимание на наиболее важном. Умеет анализировать, сравнивать, классифицировать, обобщать, конкретизировать и систематизировать изученный материал, выделяя ключевые моменты и устанавливая причинно-следственные связи. Четко формулирует ответы, уверенно интерпретирует результаты анализов и других исследований, а также решает сложные задачи. Студент хорошо знаком с методами исследования, необходимыми для практической деятельности, и умеет связывать теоретические аспекты дисциплины (модуля) с практическими задачами.
9	Отлично	Зачтено	
8	Отлично	Зачтено	
7	Хорошо	Зачтено	Студент обладает знаниями предмета почти в полном объеме рабочей программы и самостоятельно, логически последовательно и всесторонне отвечает на все вопросы, акцентируя внимание на наиболее значимых моментах. Он умеет анализировать, сравнивать, классифицировать, обобщать, конкретизировать и систематизировать изученный материал, выделяя его ключевые аспекты и устанавливая причинно-следственные связи. Формулирует свои ответы, уверенно интерпретирует результаты анализов и других исследований, а также решает сложные ситуационные задачи. Студент хорошо знаком с методами исследования, необходимыми для практической деятельности, и умеет связывать теоретические аспекты предмета с практическими задачами.
6	Хорошо	Зачтено	
5	Удовлетворительно	Зачтено	Студент обладает базовыми знаниями по дисциплине, но испытывает трудности при самостоятельных ответах и использует неточные формулировки. В ходе ответов он допускает ошибки,
4	Удовлетворительно	Зачтено	

Десятибалльная оценка	Пятибалльная оценка	Оценка за зачет	Общая характеристика результата обучения по дисциплине (модулю)
			касающиеся сути вопросов. Студент способен решать только самые простые задачи и владеет лишь минимальным набором методов исследования.
3	Не сдан	Не зачтено	Студент не овладел обязательным минимумом знаний по предмету и не может ответить на вопросы, даже если преподаватель задает дополнительные наводящие вопросы.
2	Не сдан	Не зачтено	
1	Не сдан	Не зачтено	

Дисциплина (модуль) «Разработка на Python. Профессиональный» оценивается следующим образом:

Активность	Вес	Количество	Описание
Коллоквиум	30%	2	Устные ответы на вопросы, список которых известен студенту заранее
Проекты	70%	6	Каждый проект разбивается на этапы и оценивается по заданным критериям

Формула расчёта итоговой оценки по дисциплине (модулю) «Разработка на Python. Профессиональный»: «0,3 × среднее за коллоквиум + 0,6 × среднее за проекты».

При изучении дисциплины (модуля) так же возможно получение бонусных баллов.

Текущий контроль успеваемости обучающихся по дисциплине (модулю)

Примерные задания для коллоквиума

Коллоквиум состоит из трёх шагов, связанных между собой:

1. **Анализ кода**
   Ты рассказываешь, что делает программа, объясняешь логику её работы, находишь потенциальные ошибки. Проверяется умение “читать код глазами”, объяснять, как он выполняется, и замечать проблемы.
2. **Дебаг (поиск и исправление ошибок)**
   Преподаватель предлагает исправить найденные ошибки — например, заменить неверный HTTP-метод, добавить обработку исключений или корректно передать данные в запрос. Проверяется внимание к деталям, знание синтаксиса и принципов HTTP/REST, понимание типичных ошибок.
3. **Дописать часть кода (мини-задача)**
   Нужно реализовать недостающий фрагмент — например, добавить новую команду бота, обработчик, функцию или часть логики. Проверяется умение применять базовые принципы написания кода: функции, структуры данных, работа с API и JSON.

В конце могут быть **дополнительные теоретические вопросы** по теме — например, про работу callback-запросов, обработку ошибок, структурирование кода и т.д.

Как оцениваем

Коллоквиум оценивается **по процессу работы с кодом**, а не по готовому решению. Главное — показать понимание, логику и грамотный ход мыслей.

Критерий	Что проверяется	Пример хорошего ответа
Понимание	Насколько точно студент	«Функция add_task делает POST-

логики программы	объясняет, что делает код, какие модули и принципы используются	запрос в API GitLab для создания задачи, но в коде сейчас используется GET, поэтому сервер вернёт ошибку»
Умение исправлять ошибки	Замечает ли студент проблемы (метод, токен, обработка исключений, структура данных) и может ли предложить решение	«Токен не стоит хранить в коде — лучше вынести его в .env и подгружать через переменные окружения»
Практическое мышление	Способность предложить улучшение или дополнение	«Для /list можно сделать requests.get, распарсить response.json() и пройтись по задачам циклом»
Аргументация	Умение объяснить, почему именно так нужно исправить или реализовать	«Код 201 означает, что объект успешно создан, поэтому его нужно проверять после POST-запроса»

Как подготовиться

- Повтори темы: работа с **HTTP-запросами**, **модулями requests и json**, основы **Telegram Bot API**, обработку ошибок (try/except), и принципы **REST**.
- Попробуй самостоятельно объяснить, как работает готовый код — шаг за шагом.
- Обрати внимание, **как правильно обрабатывать ошибки и проверять ответы от API**.
- Подумай, как можно улучшить читаемость и структуру кода.

Коллоквиум — это не проверка на "знание синтаксиса", а возможность показать, **что ты умеешь думать как разработчик**: видишь логику, понимаешь, где ошибка, и можешь предложить адекватное решение.

Как проходит коллоквиум

Коллоквиум — это итоговая оценка твоих навыков разработки на Python, включающая как проверку теоретической базы, так и решение практической задачи в условиях, максимально приближённых к реальной работе.

В этот раз мы уходим от устного формата и субъективной оценки — весь коллоквиум построен так, чтобы результаты каждого студента были измеримыми, прозрачными и объективными.

Коллоквиум длится **3 часа** и проходит в два этапа: теоретический тест и практическая часть с задачей в проекте.

Слоты для коллоквиума есть у вас в календарях. Сдать коллоквиум в другой слот нельзя!

Формат и тайминг

Общая длительность: 3 часа.

1. Организационный блок — 20 минут

- Брифинг и объяснение правил.
- Раздача вариантов теоретического теста.
- Ответы на вопросы по проведению.

2. Теоретическая часть — 30 минут

- Письменный тест с вопросами разных типов.

3. Перерыв — 10 минут

4. Подготовка к практике — 10 минут

- Получение варианта задания.
- Запуск прокторинга.
- Подготовка окружения.

5. Практическая часть — 100 минут

- Работа с проектом: нужно реализовать указанный в TODO функционал.
- Проверка осуществляется автоматическими тестами.

- На практической части можно использовать IDE, **без установленных ИИ агентов и копилотов** (Cursor использовать нельзя!)
- Практическую часть можно написать только при написании теоеретической части

6. Завершение — 10 минут

- Формирование архива с решением.
- Загрузка в LMS.

Как оцениваем

Главный принцип — **объективность и прозрачность**.
Теория: оценивается по чётким критериям.
Практика: оценивается исключительно автотестами — никакой субъективной интерпретации.
Итоговая оценка складывается из:

- 40% — теория
- 60% — практика

Примерные задания по проекту

Цель проекта

Разработать полнофункциональный REST API сервис для управления задачами с:

- Многопользовательской системой авторизации (JWT)
- Хранением данных в MongoDB
- Интеграцией с внешним API (Nager.Date) для импорта праздников

Основные задачи проекта

1. **Настроить FastAPI-приложение** с конфигурацией и подключением к MongoDB.
2. **Реализовать авторизацию**:
   - Регистрация пользователей с хешированием паролей
   - Аутентификация с выдачей JWT-токенов
   - Защита эндпоинтов (Bearer authentication)
3. **Создать CRUD-операции** для задач/событий, привязанных к конкретному пользователю.
4. **Ограничить доступ** к CRUD и импорт-маршрутам только авторизованным пользователям.
5. **Разработать модуль импорта** праздников из Nager.Date с нормализацией и дедупликацией данных.
6. **Обеспечить документацию** (Swagger).

Исходные данные и источники

- **Публичные API**
  - https://date.nager.at — JSON.
- **Переменные окружения (.env)**
  - `MONGODB_DSN` — строка подключения к MongoDB.
  - `JWT_SECRET` — секретный ключ для подписания токенов.
  - `JWT_EXPIRE_MINUTES` — время жизни токена.
- **Документация**
  - FastAPI.
  - Motor (async MongoDB driver).
  - Python-jose.

Входные данные

- HTTP-запросы:
  - Регистрация: `email`, `password`.

    - Логин: `email`, `password`.
    - CRUD: JSON-тела задач/событий + query-фильтры.
    - Импорт Nager: `country` (ISO-2), `year`.
- Переменные окружения:
    - `MONGO_DSN` (например, `mongodb://localhost:27017`).
    - `MONGO_DB` (например, `planner`).
    - `JWT_SECRET` (≥ 32 символов).
    - `JWT_EXPIRE_MINUTES` (например, `60`).

Требования (ТЗ)

Функциональные

- Асинхронные маршруты FastAPI.
- **Авторизация**:
    - `POST /auth/register` — создать пользователя (email + пароль), ответ 201.
    - `POST /auth/jwt/login` — выдать `access_token` (JWT, Bearer), form-data.
    - Доступ к CRUD/импорту — **только** с валидным токеном (`Authorization: Bearer <token>`).
    - Многопользовательский режим — каждый пользователь видит **только свои** задачи.
- CRUD по задачам/событиям (в пределах текущего пользователя):
    - `POST /tasks`;
    - `GET /tasks` (фильтры: `date`, `type`, `q`);
    - `GET /tasks/{id}`;
    - `PATCH /tasks/{id}`;
    - `DELETE /tasks/{id}`.
- Импорт из Nager.Date:
    - `POST /import/nager?country=XX&year=YYYY` — сохранить `type="holiday"` с дедупликацией.
- Хранилище: **MongoDB (Motor)**.
- Документация: Swagger `/docs`, OpenAPI `/openapi.json`.
- (Опционально) HTML-просмотр `/ui/tasks` (Jinja2).

Нефункциональные

- Python 3.12+, **async**-вызовы БД.
- Стиль кода (**ruff / PEP 8**).
- Пароли — **хеш** (`passlib[bcrypt]`); JWT — секрет из ENV, TTL (например, 60 мин.).
- Валидация входа (Pydantic), корректные HTTP-статусы, JSON-сообщения об ошибках.

Эндпоинты и примеры

Регистрация

`POST /auth/register`:

```
{
  "email": "alice@example.com",
  "password": "Qwerty123!"
}
```

201 Created:

```
{
  "id": "66cf1db2e9...",
  "email": "alice@example.com"
}
```

Ошибки: `409 {"detail":"Email already registered"}`.

Логин

`POST /auth/jwt/login`:

Важно: Используется `application/x-www-form-urlencoded`, а не JSON!

Формат запроса (form-data):

```
username=alice@example.com&password=Qwerty123!
```

Пример curl:

```
curl -X POST http://localhost:8000/auth/jwt/login \
  -H "Content-Type: application/x-www-form-urlencoded" \
  -d "username=alice@example.com&password=Qwerty123!"
```

200 OK:

```
{
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "token_type": "bearer"
}
```

Ошибки: `401 {"detail":"Invalid credentials"}` или `{"detail":"LOGIN_BAD_CREDENTIALS"}`.

Создать задачу

`POST /tasks` (Bearer):

```
{
  "title": "Сдать ДЗ по FastAPI",
  "date": "2025-10-02",
  "type": "task"
}
```

201 Created — объект `TaskOut`.

Список задач

`GET /tasks?date=2025-10-02&type=task&q=FastAPI` (Bearer) → **200 OK** — `[TaskOut]`.

Получить/обновить/удалить

- `GET /tasks/{id}` — **200/404**.
- `PATCH /tasks/{id}` — **200**.
- `DELETE /tasks/{id}` — **204/404**.

Импорт Nager.Date

`POST /import/nager?country=RO&year=2025` (Bearer).

200 OK:

```
{
  "imported": 12,
  "skipped": 3,
  "details": [
    {
      "id": "...",
      "title": "National Day",
      "date": "2025-12-01",
      "type": "holiday"
    }
  ]
}
```

Ошибки: `400` (валидация), `502` (внешний сервис недоступен).

План работ (чек-лист)

Этап 1: Настройка проекта и конфигурация

- [ ] 1.1. Изучить структуру проекта (см. раздел "Структура проекта")
- [ ] 1.2. Настроить конфигурацию (`config.py`) с переменными окружения
- [ ] 1.3. Инициализировать FastAPI приложение (`main.py`)
- [ ] 1.4. Подключить MongoDB (Motor)

Этап 2: Модели данных

- [ ] 2.1. Изучить модель `User` для MongoDB (email, hashed_password)
- [ ] 2.2. Изучить модель `Task` для MongoDB (user_id, title, date, type, status, source, meta)
- [ ] 2.3. Изучить Pydantic схемы для валидации (`UserCreate`, `TaskCreate`, `TaskUpdate`, `TaskOut`)
- [ ] 2.4. (Опционально) Настроить индексы в MongoDB (unique на email, композитные на tasks)

Этап 3: Авторизация

Обязательно изучи спецификацию авторизации, чтобы точно понять, как она должна работать в проекте в AUTH_SPECIFICATION.md

- [ ] 3.1. Выбрать подход: ручная JWT или FastAPI Users
- [ ] 3.2. Настроить хеширование паролей (`passlib[bcrypt]`)
- [ ] 3.3. Реализовать генерацию и валидацию JWT токенов
- [ ] 3.4. Создать эндпоинт `POST /auth/register` (201, JSON)
- [ ] 3.5. Создать эндпоинт `POST /auth/jwt/login` (200, form-data → JWT)

- [ ] 3.6. Создать зависимость `get_current_user` для извлечения пользователя из токена
- [ ] 3.7. Протестировать регистрацию и логин вручную или через Swagger

Этап 4: CRUD операции для задач

- [ ] 4.1. `POST /tasks` — создание задачи (требует авторизации)
- [ ] 4.2. `GET /tasks` — список задач с фильтрами (только свои задачи!)
- [ ] 4.3. `GET /tasks/{id}` — получение задачи (проверка владельца)
- [ ] 4.4. `PATCH /tasks/{id}` — обновление задачи (проверка владельца)
- [ ] 4.5. `DELETE /tasks/{id}` — удаление задачи (проверка владельца)
- [ ] 4.6. Убедиться, что все эндпоинты возвращают **только** задачи текущего пользователя

Этап 5: Импорт из Nager.Date

- [ ] 5.1. Создать HTTP-клиент для Nager.Date API
- [ ] 5.2. Реализовать функцию `fetch_nager_public_holidays(country, year)`
- [ ] 5.2. Создать эндпоинт `POST /import/nager?country=XX&year=YYYY`
- [ ] 5.3. Реализовать дедупликацию по `(user_id, meta.source_id)`
- [ ] 5.4. Обработать ошибки внешнего сервиса

Этап 6: Документация

- [ ] 6.1. Обновить README.md с инструкциями по запуску
- [ ] 6.2. Проверить Swagger документацию на `/docs`
- [ ] 6.3. (Опционально) Добавить HTML интерфейс `/ui/tasks`
- `README.md`**:** подготовь краткое описание, системные требования, команды локального запуска и ссылку на Swagger-документацию (`/docs`).

Должно быть достаточно трёх команд, чтобы поднять проект локально: создание виртуального окружения, установка зависимостей, запуск приложения.

Рекомендации по Git

- Ветка: `part-2-fastapi-mongo-auth`.
- MR в `main`:
  - со списком эндпоинтов;
  - инструкцией запуска;
  - ссылкой на Swagger.

Задания для промежуточной аттестации по дисциплине (модулю)

№ п/п	Задание	Ответ	Компетенция
1	Назовите основные методы HTTP.	GET, POST, PUT, DELETE	ОПК-6
2	Какой командой в Git создают новую ветку?	команда branch	ОПК-6
3	Перечислите статус-коды HTTP в диапазоне 200.	200, 201, 202, 203, 204	ОПК-6
4	Какой инструмент в GitLab используется для совместного ревью кода?	merge request	ОПК-6
5	Назовите базовые операции CRUD в MongoDB.	Create, Read, Update, Delete	ОПК-6
6	Какой библиотекой в Python взаимодействуют с внешними API асинхронно?	httpx	ОПК-6
7	Назовите уровни логирования в стандартном модуле logging.	DEBUG, INFO, WARNING, ERROR,	ПК-3

		CRITICAL	
8	Какой тип зависимости используется в FastAPI для инъекции?	Depends	ПК-3
9	Перечислите основные стратегии инвалидации кеша в Redis.	TTL, manual deletion, LRU	ПК-3
10	Какой инструмент в FastAPI запускает задачи по расписанию?	APScheduler	ПК-3
11	Назовите базовые компоненты middleware в FastAPI.	Request, Response, Exception handlers	ПК-3
12	Какой командой в Git отправляют изменения на сервер?	команда push	ПК-3
13	Перечислите типы запросов в асинхронных маршрутах FastAPI.	GET, POST	ПК-4
14	Какой формат данных используется в MongoDB для хранения?	JSON	ПК-4
15	Назовите основные функции кеширования в Redis.	set, get, expire, delete	ПК-4
16	Какой модуль в Python отвечает за логирование?	logging	ПК-4
17	Перечислите шаги организации проекта на FastAPI.	настройка, маршруты, модели, зависимости, запуск	ПК-4
18	Какой инструмент для работы с HTML-шаблонами в FastAPI?	Jinja2	ПК-4
19	Назовите типы периодических задач в FastAPI.	cron, interval, date	ОПК-6