ЦЕНТРАЛЬНЫЙ УНИВЕРСИТЕТ

**УТВЕРЖДЕНА**

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

**Рабочая программа дисциплины (модуля)**
**«Основы разработки на Go»**

**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*

ДОКУМЕНТ ПОДПИСАН
ЭЛЕКТРОННОЙ ПОДПИСЬЮ

Сертификат: 0215B0C000D1B3FEB44A37EE631E0C27D4
Владелец: Ивашкевич Евгений Васильевич
Действителен: с 13.01.2026 до 13.04.2027

Электронный документ

**Москва**
**2025**

**Содержание**

1. **Краткая характеристика дисциплины (модуля)** ..... 3
2. **Перечень планируемых результатов обучения** ..... 4
3. **Тематический план** ..... 6
4. **Содержание дисциплины (модуля)** ..... 6
5. **Учебно-методическое обеспечение** ..... 7
6. **Материально-техническое обеспечение** ..... 8
7. **Методические и оценочные материалы** ..... 9

## 1. Краткая характеристика дисциплины (модуля)

Рабочая программа дисциплины (модуля) «Основы разработки на Go» 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) "Основы разработки на Go" позволяет студентам освоить современный язык программирования, востребованный в индустрии для создания микросервисов, облачных приложений и высоконагруженных систем, что повышает их конкурентоспособность на рынке труда.

Кроме того, данная дисциплина способствует развитию понимания принципов эффективного кодирования и параллельного программирования, что является фундаментом для дальнейшего обучения в области компьютерных наук и способствует успешному применению навыков в реальных проектах.

**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) «Основы разработки на Go» подходит для обучающихся, знакомых с основами программирования или базовыми концептами разработки на Python.

**Цель изучения дисциплины (модуля):** формирование базовых навыков создания эффективных, высокопроизводительных программ на языке Go для решения практических задач в области разработки программного обеспечения.

**Задачи изучения дисциплины (модуля):**

— освоение синтаксиса языка Go;
— изучение работы с пакетами и модулями;
— практика параллельного программирования;
— разработка простых приложений и тестов.

**В результате освоения дисциплины (модуля), обучающийся должен:**

***знать:***

— основы Go: синтаксис, типы данных, структуры;
— пакеты стандартной библиотеки;
— основы тестирования;

***уметь:***

— писать, отлаживать и структурировать код;
— разрабатывать программные инструменты;
— разрабатывать и тестировать сервисы;

***владеть:***

— навыком использования языка Go как инструмента для решения прикладных задач;
— созданием эффективных и надежных сервисов.

## 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) |
|---|---|---|---|
| УК-1. | Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач | УК-1.1. | Знает методы поиска и анализа информации в области искусственного интеллекта, основные принципы критической оценки источников информации и их релевантности. |
| | | УК-1.2. | Умеет критически оценивать источники информации и синтезировать данные из различных источников для решения задач, применять системный подход к анализу и решению комплексных проблем |
| | | УК-1.3. | Имеет практический опыт работы с современными инструментами и технологиями для обработки информации, формулировании и структурировании задач на основе полученной информации |
| УК-2. | Способен определять круг задач в рамках поставленной цели и выбирать оптимальные способы их решения, исходя из действующих правовых норм, имеющихся ресурсов и ограничений | УК-2.1. | Знает действующие правовые нормы, регулирующие деятельность в области решения задач, основные методы и подходы к определению круга задач |
| | | УК-2.2. | Умеет определять круг задач в рамках поставленной цели, выбирать оптимальные способы решения задач, учитывая имеющиеся ресурсы и ограничения |
| | | УК-2.3. | Имеет практический опыт применения знаний о правовых нормах и ресурсах в реальных ситуациях, разработки и реализации решений в соответствии с установленными ограничениями |
| ОПК-1. | Способен находить, формулировать и решать актуальные и значимые проблемы прикладной и компьютерной математики | ОПК-1.1. | Знает основные методы и подходы к решению задач прикладной и компьютерной математики, включая алгоритмы, математическое моделирование и теорию оптимизации, а также современные инструменты и технологии, используемые в этой области |

| | | | |
|---|---|---|---|
| | | ОПК-1.2. | Умеет анализировать и формулировать математические задачи, применять соответствующие методы и алгоритмы для их решения, а также интерпретировать и представлять результаты в понятной и доступной форме |
| | | ОПК-1.3. | Имеет практический опыт работы над проектами или исследованиями в области прикладной и компьютерной математики, включая участие в конкурсах, олимпиадах или научных публикациях, где были решены актуальные и значимые задачи |
| ПК-1. | Способен формулировать задачи с математической точностью, обосновывать утверждения строго и анализировать полученные результаты в области математики и компьютерных наук | ПК-1.1. | Знает методы и подходы к формулированию задач, а также основные принципы математического доказательства и анализа результатов. |
| | | ПК-1.2. | Умеет корректно ставить и формулировать математические задачи, применять строгие методы доказательства и анализировать полученные результаты. |
| | | ПК-1.3. | Имеет опыт работы с задачами в области математики и компьютерных наук, включая применение математических методов для решения практических задач |
| ПК-2. | Способен решать типовые задачи профессиональной деятельности в области искусственного интеллекта, опираясь на информационную и библиографическую культуру, используя информационно-коммуникационные технологии и учитывая основные требования информационной безопасности | ПК-2.1. | Знает основы информационной и библиографической культуры, а также принципы информационной безопасности и применения информационно-коммуникационных технологий в профессиональной деятельности |
| | | ПК-2.2. | Умеет эффективно использовать информационно-коммуникационные технологии для решения стандартных задач профессиональной деятельности, учитывая требования информационной безопасности |
| | | ПК-2.3. | Имеет опыт работы с информационными ресурсами и технологиями в области искусственного интеллекта, включая соблюдение норм информационной безопасности |

## 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 | Основы языка программирования Go | 8 | 8 | | | 38 | Контрольная работа Домашнее задание |
| 2 | Применение Go для разработки инструментов | 10 | 10 | | 6 | 40 | Контрольная работа Домашнее задание |
| 3 | Разработка сервисов на Go | 10 | 10 | 2 | 6 | 40 | Проект Домашнее задание |
| | ***Зачет с оценкой*** | | | | | | |
| | ***Итого:*** | ***28*** | ***28*** | ***2*** | ***14*** | ***118*** | |
| ***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 | Основы языка программирования Go | Git & GitLab. Основы языка, базовые конструкции и типы данных. Массивы и слайсы. Концепция указателей, мапы. Структуры и интерфейсы |
| 2 | Применение Go для разработки инструментов | Функции стандартной библиотеки (аргументы командной строки, файлы, пакет io, strings). Пакеты, функции и дженерики. Структура проекта и использование сторонних библиотек. Юнит-тестирование |
| 3 | Разработка сервисов на Go | Горутины и каналы. Контексты и Graceful Shutdown. Протокол HTTP и реализация REST-сервера. Интеграционное тестирование сервисов. gRPC: формат ProtoBuf. gRPC: реализация взаимодействия. Анализ использования сторонних компонент |

## 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. Цукалос, М. Golang для профи: работа с сетью, многопоточность, структуры данных и машинное обучение с Go : практическое руководство / М. Цукалос. - Санкт-Петербург : Питер, 2021. - 720 с. - (Серия «Для профессионалов»). - ISBN 978-5-4461-1617-1. - Текст : электронный. - URL: https://znanium.ru/catalog/product/1733707.

2. Индрасири, К. gRPC: запуск и эксплуатация облачных приложений. Go и Java для Docker и Kubernetes : практическое руководство / К. Индрасири, Д. Курупу. - Санкт-Петербург : Питер, 2021. - 224 с. - (Серия «Бестселлеры O'Reilly»). - ISBN 978-5-4461-1737-6. - Текст : электронный. - URL: https://znanium.com/catalog/product/1733695.

3. Саммерфильд, М. Программирование на Go. Разработка приложений XXI века : практическое руководство / М. Саммерфильд ; пер. с англ. А. Н. Киселёва. — 2-е изд. - Москва : ДМК Пресс, 2023. - 581 с. - ISBN 978-5-89818-611-1. - Текст : электронный. - URL: https://znanium.com/catalog/product/2108489.

4. Макгаврен, Д. Head First. Изучаем Go : практическое руководство / Д. Макгаврен. - Санкт-Петербург : Питер, 2020. - 544 с. - (Серия «Head First O'Reilly»). - ISBN 978-5-4461-1395-8. - Текст : электронный. - URL: https://znanium.com/catalog/product/1756123.

5. Батчер, М. Go на практике / Мэтт Батчер, Мэтт Фарина ; пер. с англ. Р.Н. Рагимова ; под науч. ред. А.Н. Киселева. - Москва : ДМК Пресс, 2017. - 374 с. - ISBN 978-5-97060-477-9. - Текст : электронный. - URL: https://znanium.com/catalog/product/1028131.

6. 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.

***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.

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.

## 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) «Основы разработки на Go» в рамках текущего контроля успеваемости используются такие виды учебной работы, как лекции, семинары, консультации, контрольные работы, соревнование и домашние задания, 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.

*Контрольная работа* – письменная работа с набором задач, которые нужно решить за ограниченное время.

Цель контрольной работы – получить специальные знания по одной или нескольким темам дисциплины и продемонстрировать навыки их практического применения.

*Соревнование* – организованное мероприятие, в рамках которого участники соперничают друг с другом для достижения определенной цели, демонстрируя свои навыки, знания или способности в заданной области.

*Домашнее задание* – набор задач по темам недели.

При работе над домашними заданиями важно внимательно ознакомиться с требованиями и сроками выполнения. Рекомендуется разбивать задания на этапы, чтобы избежать перегрузки и лучше усвоить материал, использовать различные источники информации, включая учебники и онлайн-ресурсы, для более глубокого понимания темы.

*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.

## Система оценивания результатов обучения по дисциплине (модулю)

### Критерии получения уровня и оценивания сформированности компетенций по дисциплине «Основы разработки на Go»

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 |

| Десятибалльная оценка | Пятибалльная оценка | Оценка за зачет | Общая характеристика результата обучения по дисциплине (модулю) |
|---|---|---|---|
| 4 | Удовлетворительно | Зачтено | по дисциплине, но испытывает трудности при самостоятельных ответах и использует неточные формулировки. В ходе ответов он допускает ошибки, 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) «Основы разработки на Go» оценивается следующим образом:

| Активность | Вес | Количество | Описание |
|---|---|---|---|
| Контрольная работа | 30% | 3 | Письменная работа с набором задач, которые нужно решить за ограниченное время |
| Проект | 40% | 1 | Подготовка проекта |
| Зачет с оценкой | 30% | 1 | Защита итогового проекта |

**Формула расчёта итоговой оценки по дисциплине (модулю) «Основы разработки на Go»:** «0,3 × среднее за контрольные работы + 0,4 × среднее за проект + 0,3 × среднее за зачет с оценкой».

**Текущий контроль успеваемости обучающихся по дисциплине (модулю)**

**Примерные задания для контрольной работы**

**Контрольная работа №1**

1. Выберите правильный способ объявления переменной в Go:
a) var x int = 5,
b) int x = 5,
c) x := 5,
d) x = 5 int.

2. Что такое слайс в Go и чем он отличается от массива?

3. Напишите код на Go, который создает мапу с ключами типа string и значениями типа int, добавляет в нее три пары и выводит содержимое.

4. Указатели в Go позволяют работать с памятью напрямую (Верно/Неверно).

5. Сопоставьте концепции Go с их описаниями:
1) Структура,
2) Интерфейс,
3) Указатель.

Описания:

a) Тип, определяющий набор методов без реализации,
b) Ссылка на память объекта,
c) Агрегированный тип данных с полями.

6. Объясните, почему в Go нет наследования классов, как в других языках.

7. Напишите функцию на Go, которая принимает массив int и возвращает его сумму.

8. Какой тип данных в Go используется для динамических коллекций?
a) Array,
b) Slice,
c) Map,
d) Struct.

9. Реализуйте структуру Person с полями Name и Age, создайте экземпляр и выведите его поля.

10. Почему интерфейсы в Go важны для полиморфизма? Приведите пример.

**Контрольная работа №2**

1. Какой пакет Go используется для работы с аргументами командной строки?
a) os,
b) flag,
c) io,
d) strings.
2. Напишите юнит-тест на Go для функции, которая складывает два числа.
3. Что такое дженерики в Go и для чего они используются?
4. Горутины в Go позволяют выполнять код параллельно (Верно/Неверно).
5. Сопоставьте компоненты REST-сервера в Go:
1) Контекст,
2) Graceful Shutdown,
3) Канал.
Описания:
a) Механизм для передачи данных между горутинами,
b) Управление временем жизни запросов,
c) Корректное завершение сервера.
6. Реализуйте простой HTTP-сервер на Go, который отвечает "Hello, World!" на GET-запрос к корню.
7. Что такое ProtoBuf в контексте gRPC?
8. Выберите правильный способ реализации graceful shutdown в Go:
a) Использовать context.WithCancel,
b) Просто вызвать os.Exit,
c) Игнорировать сигналы,
d) Использовать только каналы.
9. Напишите интеграционный тест для REST-сервера, проверяющий ответ на POST-запрос.
10. Почему анализ сторонних компонент важен при разработке сервисов на Go?

**Примерные задания для проекта**

**Проект 2. Часть 1: Базовый REST API для управления привычками**

**Описание задачи**

Создай REST API для управления привычками с базовым функционалом: добавление привычки, отметка выполнения и получение списка привычек.

**Цель**
Освоить навыки реализации REST API на Python, работы с HTTP-методами, обработки JSON-данных, организации бизнес-логики и хранения данных (in-memory).
**Структура проекта**

**ВАЖНО! Строго придерживайся этой структуры для прохождения автотестов!**

```
project_root/
├── habit_tracker/              # Обязательное имя пакета
│   ├── __init__.py             # Пустой файл
│   │
│   ├── api/                    # Обязательная папка для роутов
│   │   ├── __init__.py         # Пустой файл
│   │   └── habits.py           # Обязательное имя файла с роутами
│   │
│   ├── core/                   # Обязательная папка для бизнес-логики
│   │   ├── __init__.py         # Пустой файл
│   │   ├── models.py           # Модели данных (Habit, Pydantic-модели)
│   │   └── services.py         # Бизнес-логика (функции работы с привычками)
│   │
│   └── main.py                 # Точка входа приложения
│
├── .env.example                # Пример файла переменных окружения
├── requirements.txt            # Обязательный файл зависимостей
└── README.md                   # Документация проекта
```

**Требования к реализации**
**1. Файл habit_tracker/main.py**

**Задача:** Создай точку входа приложения с инициализацией FastAPI

**Требования:**

- Создай экземпляр FastAPI с названием "Habit Tracker API"
- Имя переменной должно быть `app` (для автотестов!)
- Подключи роутер из модуля `habits`

**Пример структуры:**

```python
"""Точка входа приложения Habit Tracker."""

from fastapi import FastAPI
from habit_tracker.api import habits

# TODO: Создать экземпляр FastAPI
app = ...
```

```
# TODO: Подключить роутер
```

## 2. Файл `habit_tracker/core/models.py`

**Задача:** Создай модели данных для привычек

### 2.1. Внутренняя модель привычки

**Требования:**

- Класс должен называться `Habit` (для автотестов!)
- Поля: `id` (int), `name` (str), `marks` (List[date])
- `marks` должен быть пустым списком при создании

**Пример структуры:**

```python
"""Модели данных для привычек."""

from datetime import date
from typing import List
from pydantic import BaseModel, validator


class Habit:
    """Внутренняя модель привычки для хранения в памяти."""

    def __init__(self, id: int, name: str):
        # TODO: Инициализировать поля
        pass
```

### 2.2. Pydantic модели для API

**Требования:**

- `HabitCreate` - модель для создания (поле `name`)
- `HabitResponse` - ответ после создания (поля `id`, `name`)
- `HabitMarkResponse` - ответ после отметки (поля `id`, `name`, `last_marked_at`)
- `HabitListResponse` - для списка (поля `id`, `name`, `marks`)

**Подсказка:** Используй Pydantic validator для валидации `name` (не должно быть пустым)

**Пример структуры:**

```python
class HabitCreate(BaseModel):
    """Модель для создания привычки."""
    name: str

    @validator('name')
    def name_must_not_be_empty(cls, v):
        # TODO: Реализовать проверку на пустое имя
        pass

class HabitResponse(BaseModel):
    """Модель ответа после создания привычки."""
    # TODO: Добавить поля id и name
    pass
```

```
# TODO: Реализовать остальные модели (HabitMarkResponse, HabitListResponse)
```

### 3. Файл `habit_tracker/core/services.py`

**Задача:** Реализуй бизнес-логику работы с привычками

**Требования:**

- Используй in-memory хранилище `habits_db: Dict[int, Habit]`
- Реализуй функции: `create_habit()`, `mark_habit()`, `get_all_habits()`
- Используй HTTPException для обработки ошибок

**Обязательные имена (для автотестов):**

- Функция создания: `create_habit(name: str) -> Habit`
- Функция отметки: `mark_habit(habit_id: int) -> Habit`
- Функция получения списка: `get_all_habits() -> List[Habit]`
- Переменная хранилища: `habits_db`
- Счётчик ID: `_next_id`

**Пример структуры:**

```python
"""Бизнес-логика для работы с привычками."""

from datetime import date
from typing import Dict, List
from fastapi import HTTPException
from habit_tracker.core.models import Habit

# In-memory хранилище
habits_db: Dict[int, Habit] = {}
_next_id = 1


def create_habit(name: str) -> Habit:
    """Создать новую привычку."""
    global _next_id

    # TODO: 1. Проверить, что name не пустое
    #          Если пустое - raise HTTPException(status_code=400, detail="Habit
name cannot be empty.")

    # TODO: 2. Проверить, что привычка с таким именем не существует
    #          Если существует - raise HTTPException(status_code=400,
detail="Habit with this name already exists.")

    # TODO: 3. Создать объект Habit с текущим _next_id и name

    # TODO: 4. Сохранить в habits_db

    # TODO: 5. Увеличить _next_id

    # TODO: 6. Вернуть созданную привычку
    pass
```

```python
def mark_habit(habit_id: int) -> Habit:
    """Отметить выполнение привычки за текущий день."""

    # TODO: 1. Получить привычку из habits_db по habit_id
    #          Если не найдена - raise HTTPException(status_code=404,
detail="Habit not found.")

    # TODO: 2. Получить сегодняшнюю дату (date.today())

    # TODO: 3. Проверить, что today не в habit.marks
    #          Если уже есть - raise HTTPException(status_code=400,
detail="Habit already marked for today.")

    # TODO: 4. Добавить today в habit.marks

    # TODO: 5. Вернуть обновленную привычку
    pass


def get_all_habits() -> List[Habit]:
    """Получить список всех привычек."""
    # TODO: Вернуть список всех привычек из habits_db
    pass
```

### 4. Файл `habit_tracker/api/habits.py`

**Задача:** Создай API эндпоинты для работы с привычками

**Требования:**

- Роутер должен быть в переменной `router` (для автотестов!)
- Реализуй 3 эндпоинта:
  - `POST /habits/` - создание привычки (возвращает 201)
  - `POST /habits/{habit_id}/mark/` - отметка выполнения (возвращает 200)
  - `GET /habits/` - получение списка (возвращает 200)
- Используй функции из `services`
- Используй response_model для типизации ответов

**Пример структуры:**

```python
"""API роуты для управления привычками."""

from typing import List
from fastapi import APIRouter, status
from habit_tracker.core import services
from habit_tracker.core.models import (
    HabitCreate,
    HabitResponse,
    HabitMarkResponse,
    HabitListResponse
)

router = APIRouter()


@router.post("/habits/", response_model=HabitResponse,
```

```
status_code=status.HTTP_201_CREATED)
def create_habit(habit: HabitCreate):
    """Создать новую привычку."""
    # TODO: 1. Вызвать services.create_habit() с habit.name
    # TODO: 2. Вернуть HabitResponse с id и name созданной привычки
    pass


@router.post("/habits/{habit_id}/mark/", response_model=HabitMarkResponse)
def mark_habit(habit_id: int):
    """Отметить выполнение привычки за текущий день."""
    # TODO: 1. Вызвать services.mark_habit() с habit_id
    # TODO: 2. Получить последнюю дату из habit.marks
    # TODO: 3. Отформатировать дату в строку (YYYY-MM-DD)
    # TODO: 4. Вернуть HabitMarkResponse
    pass


@router.get("/habits/", response_model=List[HabitListResponse])
def get_all_habits():
    """Получить список всех привычек."""
    # TODO: 1. Вызвать services.get_all_habits()
    # TODO: 2. Для каждой привычки создать HabitListResponse
    # TODO: 3. Преобразовать dates в список строк формата YYYY-MM-DD
    # TODO: 4. Вернуть список
    pass
```

### 5. Файл `requirements.txt`

**Задача:** Укажи необходимые зависимости проекта

**Требования:**

- FastAPI версии 0.104.1
- Uvicorn с поддержкой стандартных возможностей
- Pydantic для валидации
- Python-dotenv для работы с переменными окружения

### 6. Файл `.env.example`

**Задача:** Создай пример файла с переменными окружения

Добавь переменную `APP_ENV` со значением "development"

# API эндпоинты

## 1. POST /habits/ - Создать привычку

**Запрос:**

```
POST /habits/ HTTP/1.1
Content-Type: application/json

{
  "name": "Бег"
}
```

**Ответы:**

**Успех (201 Created):**

```
{
  "id": 1,
  "name": "Бег"
}
```

**Ошибка - пустое имя (400 Bad Request):**

```
{
  "detail": "Habit name cannot be empty."
}
```

**Ошибка - дубликат (400 Bad Request):**

```
{
  "detail": "Habit with this name already exists."
}
```

## 2. POST /habits/{habit_id}/mark/ - Отметить выполнение

**Запрос:**

```
POST /habits/1/mark/ HTTP/1.1
```

**Ответы:**

**Успех (200 OK):**

```
{
  "id": 1,
  "name": "Бег",
  "last_marked_at": "2025-07-07"
}
```

**Ошибка - не найдена (404 Not Found):**

```
{
  "detail": "Habit not found."
}
```

**Ошибка - уже отмечена (400 Bad Request):**

```
{
  "detail": "Habit already marked for today."
}
```

## 3. GET /habits/ - Получить список

**Запрос:**

```
GET /habits/ HTTP/1.1
```

**Ответ (200 OK):**

```
[
  {
    "id": 1,
    "name": "Бег",
    "marks": ["2025-07-06", "2025-07-07"]
  },
  {
    "id": 2,
    "name": "Чтение",
    "marks": ["2025-07-07"]
  }
]
```

**Если привычек нет:**

```
[]
```

# Запуск приложения

## 1. Создать виртуальное окружение

```
python -m venv venv
source venv/bin/activate  # На Windows: venv\Scripts\activate
```

## 2. Установить зависимости

```
pip install -r requirements.txt
```

## 3. Запустить приложение

```
uvicorn habit_tracker.main:app --reload
```

# Чек-лист перед сдачей

## Структура проекта

- [ ] Создана папка `habit_tracker/`
- [ ] Создан файл `habit_tracker/__init__.py`
- [ ] Создана папка `habit_tracker/api/`
- [ ] Создан файл `habit_tracker/api/__init__.py`
- [ ] Создан файл `habit_tracker/api/habits.py`
- [ ] Создана папка `habit_tracker/core/`
- [ ] Создан файл `habit_tracker/core/__init__.py`
- [ ] Создан файл `habit_tracker/core/models.py`
- [ ] Создан файл `habit_tracker/core/services.py`
- [ ] Создан файл `habit_tracker/main.py`
- [ ] Создан файл `requirements.txt`
- [ ] Создан файл `README.md`

## Функциональность

- [ ] `POST /habits/` создает привычку и возвращает 201
- [ ] `POST /habits/` проверяет пустое имя (400)
- [ ] `POST /habits/` проверяет дубликат имени (400)
- [ ] `POST /habits/{id}/mark/` отмечает привычку (200)
- [ ] `POST /habits/{id}/mark/` возвращает 404 если не найдена
- [ ] `POST /habits/{id}/mark/` возвращает 400 если уже отмечена сегодня
- [ ] `GET /habits/` возвращает список всех привычек (200)
- [ ] `GET /habits/` возвращает пустой массив если привычек нет

## Качество кода

- [ ] Соблюдён PEP 8
- [ ] Все функции имеют type hints
- [ ] Все функции и классы имеют docstrings
- [ ] Бизнес-логика находится в `services.py`, а не в роутах
- [ ] Нет дублирования кода

- [ ] Нет циклических импортов

### Документация

- [ ] Swagger доступен по `/docs`
- [ ] Все эндпоинты корректно отображаются в Swagger
- [ ] README.md содержит инструкции по запуску
- [ ] README.md содержит примеры использования API

### Тестирование

- [ ] Приложение запускается без ошибок
- [ ] Можно создать привычку через Swagger
- [ ] Можно отметить привычку
- [ ] Можно получить список привычек
- [ ] Обработка ошибок работает корректно

## Проект 2. Часть 2

Дедлайн прошел

Это вторая часть второго проекта! Переходи по ссылке ниже в свой форк репозитоиря.

**Работа ведется в том же репозитории, что и проект 2 часть 1. Чтобы начать работу создай новую рабочую ветку от финальной версии первой части проекта.**

Ссылка на self-service

**Чтобы работать с self-service необходимо хотя бы один раз авторизоваться в GitLab, а после переходить к созданию репозитория**

Данная работа оценивается автоматически системой авторевью. После решения отправлять ссылку на репозиторий не требуется. Оценка будет выставлена в LMS после решения задач.
Для запуска авторевью укажи в MR Label:

`python_basic_web-project_02`

**Задания для промежуточной аттестации по дисциплине (модулю)**

| № п/п | Задание | Ответ | Компетенция |
|---|---|---|---|
| 1 | Назовите базовый тип данных в Go для хранения целых чисел. | int | УК-1 |
| 2 | Укажите ключевое слово для объявления функции в Go. | func | УК-1 |
| 3 | Как называется структура данных в Go, которая хранит упорядоченный набор элементов одного типа? | слайс/срез/slice | УК-2 |
| 4 | Как в Go называется структура, которая хранит пару ключ-значение? | мап/map | УК-2 |
| 5 | Какой пакет используется для чтения аргументов командной строки? | flag | УК-2 |
| 6 | Назовите функцию из пакета io, которая читает весь | ReadFile | УК-2 |

Электронный документ

| | | | |
|---|---|---|---|
| | файл в память. | | |
| 7 | Укажите ключевое слово для объявления указателя в Go. | var/ * (в контексте объявления указателя) | ОПК-1 |
| 8 | Как называется механизм в Go для параллельного выполнения функций? | горутина/gorouti ne | ОПК-1 |
| 9 | Какой тип данных используется для хранения строки в Go? | string | ОПК-1 |
| 10 | Как называется структура, описывающая набор методов для реализации интерфейса? | интерфейс/interf ace | ОПК-1 |
| 11 | Какой пакет позволяет создавать HTTP-серверы в Go? | net/http | ПК-1 |
| 12 | Назовите формат сериализации данных, используемый в gRPC. | ProtoBuf/Protocol Buffers | ПК-1 |
| 13 | Как называется функция, которая запускает тесты в Go? | Test | ПК-1 |
| 14 | Какой файл описывает зависимости проекта Go? | go.mod | ПК-2 |
| 15 | Назовите функцию для объединения слайса строк в одну строку с разделителем. | strings.Join | ПК-2 |
| 16 | Какой пакет в Go используется для логирования? | log | ПК-2 |
| 17 | Как называется структура, которая используется для передачи контекста и отмены операций? | context.Context | ПК-2 |
| 18 | Какой метод канала используется для отправки значения? | <- | УК-1 |
| 19 | Какой тип тестирования проверяет взаимодействие нескольких компонентов сервиса? | интеграционное/ интеграционное тестирование | УК-1 |