ЦЕНТРАЛЬНЫЙ УНИВЕРСИТЕТ

УТВЕРЖДЕНА

Решением Ученого совета
АНО ВО «Центральный университет»
«25» мая 2026 г.
Протокол № 2

Рабочая программа дисциплины (модуля)
«Язык программирования Go. Часть 2»

Направление подготовки: *02.03.01 Математика и компьютерные науки*

Направленность (профиль) подготовки: *Математика и компьютерные науки*

Квалификация (степень) выпускника: *бакалавр*

Форма обучения: *очная*

Срок освоения программы: *4 года*

Год набора: *2026*

ДОКУМЕНТ ПОДПИСАН
ЭЛЕКТРОННОЙ ПОДПИСЬЮ

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

Электронный документ

Москва
2026

Содержание

1. Краткая характеристика дисциплины (модуля)

Рабочая программа дисциплины (модуля) «Язык программирования Go. Часть 2» составлена в соответствии с федеральным государственным образовательным стандартом высшего образования – бакалавриат по направлению 02.03.01 «Математика и компьютерные науки», профиль «Математика и компьютерные науки», утвержденный приказом Министерства науки и высшего образования Российской Федерации № 807 от 23.08.2017 года.

Изучение дисциплины (модуля) " Язык программирования Go. Часть 2" 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 «Математика и компьютерные науки» и входит в обязательную часть Блока 1, как дисциплина по выбору.

Дисциплина (модуль) изучается на 1 курсе в 1 или 2 семестре на выбор.

Дисциплина (модуль) «Язык программирования Go. Часть 2» 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 синтаксиса и функцирования языка Golang, его идиоматику;
— базовую архитектуру сервисов на Go.

уметь:

— писать прикладные программы на Golang;
— писать конкурентный код средствами языка;
— писать тестируемый код.

владеть:

— Golang на базовом уровне для решения прикладных задач;
— знаниями о разработке конкурентного кода;
— основами тестирования кода.

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
		ОПК-6.2.	Умеет разрабатывать математические программные продукты и комплексы с использованием современных технологий программирования
		ОПК-6.3.	Имеет практический опыт разработки интеллектуальных информационных систем для визуализации результатов исследований
ПК-3.	Способен использовать методы математического и алгоритмического моделирования при решении теоретических и прикладных задач	ПК-3.1.	Знает основные методы математического и алгоритмического моделирования, а также их применение для решения теоретических и прикладных задач
		ПК-3.2.	Умеет разрабатывать и применять математические модели и алгоритмы для решения различных задач, анализируя полученные результаты
		ПК-3.3.	Имеет практический опыт использования методов математического и алгоритмического моделирования в реальных проектах или исследованиях
ПК-4.	Способен под руководством специалиста более высокой категории разрабатывать программное обеспечение для решения прикладных задач в сфере	ПК-4.1.	Знает основные принципы разработки программного обеспечения и методы решения прикладных задач
		ПК-4.2.	Умеет применять языки программирования и инструменты разработки для создания программного обеспечения под руководством более опытного специалиста

		ПК-4.3.	Имеет практический опыт участия в проектах по разработке программного обеспечения, работая в команде под руководством специалиста более высокой категории

3. Тематический план

№п/п	Наименование раздела дисциплины (модуля)	Трудоемкость, академические часы				ТКУ (текущий контроль успеваемости)
		Очная форма				
		Аудиторная работа		Контроль	Самостоятельная работа	
		Лекции	Семинары			
1	Основы языка Go	16	26	6	56	Домашнее задание Контрольная работа Контест
2	Прикладное применение языка Go	12	20	8	42	Домашнее задание Контрольная работа Контест
	Зачет с оценкой			4		
	Итого:	***28***	***46***	***18***	***98***	
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 Go	Введение в Go и первая программа Простые типы, переменные, константы и управляющие конструкции Функции, декомпозиция кода Составные типы данных Указатели, передача по значению и ссылочная семантика Структуры и методы Ошибки, паники и defer Интерфейсы
2	Прикладное применение языка Go	Работа с файлами, консольные утилиты Пакеты, модули, зависимости, структура проекта Тестирование и отладка кода Конкурентность

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. 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.

3. Стил, Т. Black Hat Go: Программирование для хакеров и пентестеров : практическое руководство / Т. Стил, К. Паттен, Д. Коттманн. - Санкт-Петербург : Питер, 2022. - 384 с. - (Серия «Библиотека программиста»). - ISBN 978-5-4461-1795-6. - Текст : электронный. - URL: https://znanium.com/catalog/product/2122878.

Дополнительная литература:

1. 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.

2. 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. 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) «Язык программирования Go. Часть 2» в рамках текущего контроля успеваемости используются такие виды учебной работы, как лекции, семинары, контрольные работы, контесты 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.

Семинар – это форма учебной деятельности, проводимая в учебном заведении под руководством преподавателя, где студенты активно участвуют в обсуждениях, практических заданиях и других формах взаимодействия.

Для успешной подготовки к семинару рекомендуется заранее ознакомиться с темой занятия и основными материалами, чтобы иметь возможность активно участвовать в обсуждении. Также полезно подготовить вопросы и идеи для обсуждения, что поможет глубже понять материал и продемонстрировать заинтересованность.

Контест – интерактивная платформа с заданиями разного уровня сложности и автоматической проверкой результатов.

Контест позволяет оперативно оценивать усвоение материала и выявлять пробелы в знаниях через тесты и практические задачи. Такой формат способствует регулярной самопроверке и повышает мотивацию к изучению дисциплины.

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.

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 «Язык программирования Go. Часть 2»

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, если иное не указано в силлабусе дисциплины:

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
9	Отлично	Зачтено	
8	Отлично	Зачтено	

Десятибалльная оценка	Пятибалльная оценка	Оценка за зачет	Общая характеристика результата обучения по дисциплине (модулю)
			методами исследования, необходимыми для практической деятельности, и умеет связывать теоретические аспекты дисциплины (модуля) с практическими задачами.
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, касающиеся сути вопросов. Студент способен решать только самые простые задачи и владеет лишь минимальным набором методов исследования.
4	Удовлетворительно	Зачтено	
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) «Язык программирования Go. Часть 2» оценивается следующим образом:

Активность	Вес	Количество	Описание
Домашнее задание	15%	6	Набор задач по темам недели
Контест	25%	5	Интерактивные задания разного уровня сложности с

Активность	Вес	Количество	Описание
			автоматической проверкой результатов
Контрольная работа	30%	2	Письменная работа с набором задач, которые нужно решить за ограниченное время
Зачет с оценкой	30%	1	Письменная или устная работа над заданием, направленным на проверку полученных знаний и навыков по дисциплине (модулю)

Формула расчёта итоговой оценки по дисциплине (модулю) «Язык программирования Go. Часть 2»: «0,15 × среднее за домашние задания + 0,25 × среднее за контест + 0,3 × среднее за контрольные работы + 0,3 × 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 домашние задания

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
```

```python
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


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:

```
"""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

Примерные задания для контеста

1. Напишите функцию **Hello(name string) string**, которая возвращает строку вида **"Hello, {name}!"**. Если имя пустое, возвращайте **"Hello, World!"**.

2. Напишите функцию **Max(a, b int) int**, возвращающую максимальное из двух чисел.

3. Напишите функцию **Sum(numbers []int) int**, вычисляющую сумму всех элементов в срезе целых чисел.

4. Напишите функцию **Reverse(s string) string**, которая возвращает перевёрнутую строку. Учтите, что строка может содержать Unicode-символы (например, русские буквы).

5. Напишите функцию **IsPalindrome(s string) bool**, проверяющую, является ли строка палиндромом (без учёта регистра и пробелов).

6. Напишите функцию **Fibonacci(n int) int**, возвращающую n-е число Фибоначчи. Считайте, что **F(0) = 0**, **F(1) = 1**.

7. Напишите функцию **FilterEven(numbers []int) []int**, которая возвращает новый срез, содержащий только чётные числа из исходного.

8. Напишите функцию **WordCount(s string) map[string]int**, которая возвращает карту, где ключ — слово, а значение — количество его вхождений в строку (слова разделены пробелами).

9. Создайте структуру **Rectangle** с полями **Width** и **Height** (типа **float64**). Реализуйте для неё метод **Area() float64**, вычисляющий площадь.

10. Создайте структуру **Person** с полями **Name** и **Age**. Реализуйте метод **String() string**, чтобы при выводе через **fmt.Println** объекта **Person** отображалось: **"Имя: {Name}, Возраст: {Age}"**.

Задания для промежуточной аттестации по дисциплине (модулю)

№ п/п	Задание	Ответ	Компетенция
1	Назовите базовый тип данных в Go для хранения целых чисел.	int	ОПК-6
2	Укажите ключевое слово для объявления функции в Go.	func	ОПК-6
3	Как называется структура данных в Go, которая хранит упорядоченный набор элементов одного типа?	слайс / срез / slice	ОПК-6
4	Какой тип данных используется для хранения строки в Go?	string	ОПК-6
5	Укажите ключевое слово для объявления переменной в Go.	var	ОПК-6
6	Какой оператор используется для короткого объявления переменной (например, **x := 5**)?	:=	ОПК-6
7	Какой цикл используется в Go для итерации по элементам слайса или мапы?	for range	ОПК-6
8	Как называется структура, описывающая набор методов для реализации интерфейса?	интерфейс / interface	ПК-4
9	Какой пакет позволяет создавать HTTP-серверы в Go?	net/http	ПК-4
10	Как называется механизм в Go для параллельного выполнения функций?	горутина / goroutine	ПК-4
11	Какой метод канала используется для отправки значения?	<-	ПК-4
12	Как называется структура, которая используется для передачи контекста и отмены операций?	context.Context	ПК-4
13	Какой пакет используется для чтения аргументов командной строки?	flag	ПК-4
14	Назовите формат сериализации данных,	ProtoBuf /	ПК-4

	используемый в gRPC.	Protocol Buffers	
15	Назовите функцию из пакета io, которая читает весь файл в память.	ReadFile	ПК-3
16	Назовите функцию для объединения слайса строк в одну строку с разделителем.	strings.Join	ПК-3
17	Какой пакет в Go используется для логирования?	log	ПК-3
18	Какой тип тестирования проверяет взаимодействие нескольких компонентов сервиса?	интеграционное / интеграционное тестирование	ПК-3
19	Как называется функция, которая запускает тесты в Go?	Test	ПК-3
20	Какой файл описывает зависимости проекта Go?	go.mod	ПК-3