Документ подписан простой электронной подписью
Информация о владельце:
ФИО: Максимов Алексей Борисович
Должность: директор департамента по образовательной политике
Дата подписания: 14.07.2026 16:42:53
Уникальный программный ключ:
8db180d1a3f02ac9e60521a5672742735c18b1d6

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«Московский политехнический университет»

УТВЕРЖДАЮ
Декан факультета
«Информационных технологий»
_____ / Д.Г. Демидов /
«26» февраля 2026 г.

Рабочая программа дисциплины
«ВЕБ-РАЗРАБОТКА»

Направление подготовки:
09.03.01 Информатика и вычислительная техника

Образовательная программа (профиль):
«Разработка инженерного программного обеспечения»

Год начала обучения:
2026

Уровень образования:
бакалавриат

Квалификация (степень) выпускника:
Бакалавр

Форма обучения:
очная

Москва, 2026

Разработчик(и):

к.т.н., доцент / В.Ю. Верещагин /

Согласовано:

Заведующий кафедрой «Инфокогнитивные технологии»,
к.т.н., доцент / Е.А. Пухова /

Содержание

1 Цели, задачи и планируемые результаты обучения по дисциплине 4
2 Место дисциплины в структуре образовательной программы 4
3 Структура и содержание дисциплины 5
3.1 Виды учебной работы и трудоемкость для очной формы обучения 5
3.2 Тематический план изучения дисциплины для очной формы обучения 5
3.3 Содержание дисциплины 6
3.4 Тематика семинарских/практических и лабораторных занятий 6
4 Учебно-методическое и информационное обеспечение 7
4.1 Нормативные документы и ГОСТы 7
4.2 Основная литература 7
4.3 Дополнительная литература 7
4.4 Электронные образовательные ресурсы 7
4.5 Лицензионное и свободно распространяемое программное обеспечение 7
4.6 Современные профессиональные базы данных и информационные справочные системы 7
5 Материально-техническое обеспечение 8
6 Методические рекомендации 8
6.1 Методические рекомендации для преподавателя по организации обучения 8
6.2 Методические указания для обучающихся по освоению дисциплины 8
7 Фонд оценочных средств 9
7.1 Методы контроля и оценивания результатов обучения 9
7.2 Шкала и критерии оценивания результатов обучения 9
7.3 Оценочные средства 10

1 Цели, задачи и планируемые результаты обучения по дисциплине

Цель дисциплины: изучить основные понятия веб-технологий, освоить инструменты разработки и технологии необходимые для разработки веб-приложений с подключением к базе данных.

Задачи дисциплины: освоить на практике:

- основы языка JavaScript
- разработка клиентской части приложения на JavaScript
- разработка серверной части приложения с использованием фреймворка Express

Обучение по дисциплине «Веб-разработка» направлено на формирование у обучающихся следующих компетенций:

Код и наименование компетенций	**Индикаторы достижения компетенции**
ПК-1. Способен разрабатывать требования и проектировать программное обеспечение.	ИПК-1.1. Знает: Возможности существующей программно-технической архитектуры; Возможности современных и перспективных средств разработки программных продуктов, технических средств; Методологии разработки программного обеспечения и технологии программирования; Методологии и технологии проектирования и использования баз данных; Языки формализации функциональных спецификаций; Методы и приемы формализации задач; Методы и средства проектирования программного обеспечения; Методы и средства проектирования программных интерфейсов; Методы и средства проектирования баз данных; Принципы построения и виды архитектуры компьютерного программного обеспечения; Типовые решения, библиотеки программных модулей, шаблоны, классы объектов, используемые при разработке программного обеспечения; Нормативно-технические документы (стандарты), определяющие требования к технической документации на

	компьютерное программное обеспечение. ИПК-1.2. Умеет: Проводить сбор и систематизацию требований к компьютерному программному обеспечению; Выявлять взаимосвязи и документировать требования к компьютерному программному обеспечению; Проводить анализ исполнения требований к компьютерному программному обеспечению; Вырабатывать варианты реализации требований к компьютерному программному обеспечению; Проводить оценку и обоснование рекомендуемых решений; Осуществлять коммуникации с заинтересованными; сторонами; Выбирать средства реализации требований к компьютерному программному обеспечению; Выбирать средства реализации требований к компьютерному программному обеспечению; Использовать существующие типовые решения и шаблоны проектирования компьютерного программного обеспечения; Применять методы и средства проектирования компьютерного программного обеспечения, структур данных, баз данных, программных интерфейсов; Использовать командные средства разработки компьютерного программного обеспечения; Применять существующие стандарты для разработки технической документации на компьютерное программное обеспечение. ИПК-1.3.Владеет: Инструментами и технологиями разработки требований и проектирования программного обеспечения;

	Инструментами и технологиями разработки программного кода.
ПК-2. Способен осуществлять концептуально-логическое проектирование Системы и сопровождение разработанных проектных решений	ИПК-2.1 Знает: Методы выявления, формулирования и обоснования требований; Процессы жизненного цикла систем Методы моделирования устройства и функционирования ИТ-систем/продуктов модели качества систем, программных продуктов и данных; Стадии создания автоматизированной системы; Требования к содержанию документов, разрабатываемых при создании автоматизированной системы; ИПК-2.2 Умеет: Вырабатывать предложения на основе типичных (для отрасли или организации) проектных решений; Выявлять концептуальные архитектурные (технические) решения по Системе; Разрабатывать деление на подсистемы, этапность и очередность построения Системы; Пользоваться системами управления знаниями. ИПК-2.3 Владеет: Навыками выявления исходных требований к системе; навыками выработки предложений по проектным решениям; Навыками классификации, систематизации и моделирования собранных фактов, решений и требований; Навыками разработки концепции системы; Навыками управления исследованием и анализом;

2 Место дисциплины в структуре образовательной программы

Дисциплина относится к части, формируемой участниками образовательных отношений блока Б1.2.1.2, модуля «Разработка в области информационных технологий» и междисциплинарно связана с поддерживающими дисциплинами: основы программирования, базы данных, веб-технологии, сети и телекоммуникации и последующими дисциплинами: облачные технологии.

3 Структура и содержание дисциплины

Общая трудоемкость дисциплины составляет 4 зачетных единицы (144 часов).

3.1 Виды учебной работы и трудоемкость для очной формы обучения

№ п/п	Вид учебной работы	Количество часов	Семестры	
1	**Аудиторные занятия**	**72**		
	В том числе:			
1.1	Лекции	8		
1.2	Семинарские/практические занятия			
1.3	Лабораторные занятия	64		
2	**Самостоятельная работа**	**72**		
3	**Промежуточная аттестация**		4	
	Экзамен		4	
	Итого:	**144**		

3.2 Тематический план изучения дисциплины для очной формы обучения

№ п/п	Разделы/темы дисциплины	Трудоемкость, час					
		Всего	Аудиторная работа				Самостоятельная работа
			Лекции	Семинарские/ практические занятия	Лабораторные занятия	Практическая подготовка	
1	Основы языка Javascript	5	1		4		4
2	ОПП, классы и конструкторы	5	1		4		4
3	Основы Three.js	10	2		4		4
4	Three.js и работа с геометрией	10			4		6
5	Среда исполнения Node.js	12	2		4		6
6	Фреймворк Express	11	1		4		6
7	Промежуточное ПО	11	1		4		6
8	Подключение БД	10			4		6

9	CRUD и обработка ошибок	10			4		6
10	Fetch API	10			4		6
11	Реактивность интерфейса	10			4		6
12	Разработка веб-приложений	10			4		6
13	Рубежный контроль	12			6		6
Итого		**144**	**8**		**64**		**72**

3.3 Содержание дисциплины

Основы языка Javascript

Повторение основ Javascript: операторы, функции, объекты, массивы, DOM

ОПП, классы и конструкторы

Разбор основ ООП и как они могут быть реализованы в JS.

Основы Three.js

Работа с библиотекой Three.js и ее основными концепциями для работы с трехмерными объектами.

Three.js и работа с геометрией

Продолжение работы с библиотекой Three.js и изучением генерации собственных объектов

Среда исполнения Node.js

Разбор различий работы JS в браузере и Node.js, основные принципы работы Node.js, использование модуля http.

Фреймворк Express

Особенности фреймворка Express, создание собственного сервера

Промежуточное ПО

Разбор концепции промежуточного ПО и его использовании в express.

Подключение БД

Сравнение популярных СУБД и подключении одной из них к серверу приложения

CRUD и обработка ошибок

Работа с базой данных и исполнений базовых команд по работе с данными и принципами обработки ошибок

Fetch API

Разбор темы работы асинхронного кода и использование Fetch API для отправкпи запроса на сервер.

Реактивность интерфейса

Идея реактивности интерфейсов и способы ее реализации в браузере, реализация собственной простой библиотеки

Разработка веб-приложений

Реализация веб-интерфейса для разработанного веб-API на сервере и пути дальнейшего развития.

3.4 Тематика семинарских/практических и лабораторных занятий

Основы языка Javascript
ОПП, классы и конструкторы
Основы Three.js
Three.js и работа с геометрией
Среда исполнения Node.js
Фреймворк Express
Промежуточное ПО

Подключение БД
CRUD и обработка ошибок
Fetch API
Реактивность интерфейса
Разработка веб-приложений

4 Учебно-методическое и информационное обеспечение

4.1 Нормативные документы и ГОСТы

1. Приказ Минтруда России от 18.01.2017 N 44н "Об утверждении профессионального стандарта "Разработчик Web и мультимедийных приложений" (Зарегистрировано в Минюсте России 31.01.2017 N 45481)

2. п. 1 ч. 1 ст. 43 и ст. 58 Федерального закона от 29.12.2012 № 273-ФЗ «Об образовании в Российской Федерации»

4.2 Основная литература

Верещагин, В.Ю. Создание веб-страниц на стороне клиента: учебно- методическое пособие для студентов бакалавриата, обучающихся по направлению подготовки 09.03.01 «Информатика и вычислительная техника», профиль «Веб-технологии» / В.Ю. Верещагин, М.В. Даньшина. – Москва: Московский Политех, 2023. – 1 CD-R. – Загл. с титул. экрана. – Текст: электронный. ISBN 978-5-2760-2786-9. — URL: https://online.mospolytech.ru/mod/data/view.php?d=127&rid=5281&filter=1

4.3 Дополнительная литература

Полуэктова, Н. Р. Разработка веб-приложений : учебное пособие для вузов / Н. Р. Полуэктова. — Москва : Издательство Юрайт, 2023. — 204 с. — (Высшее образование). — ISBN 978-5-534-13715-6. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/519714

4.4 Электронные образовательные ресурсы

https://online.mospolytech.ru/course/view.php?id=12159

4.5 Лицензионное и свободно распространяемое программное обеспечение

1. Visual Studio Code
2. Браузеры Chrome, Edge, Firefox
3. OpenVPN с правами для запуска у студентов
4. FileZilla
5. PuTTy
6. Git
7. Node.js 18
8. Python 3.10
9. Wireshark

4.6 Современные профессиональные базы данных и информационные справочные системы

1. https://doka.guide/
2. https://developer.mozilla.org/ru/
3. https://roadmap.sh/frontend

4. https://learn.javascript.ru/

5 Материально-техническое обеспечение

Для проведения лабораторных работ и самостоятельной работы студентов подходят аудитории, оснащенные компьютерами с программным обеспечением в соответствии со списком в пункте 4.5 и подключенные к интернету.

Число рабочих мест в аудитории должно быть достаточным для обеспечения индивидуальной работы студентов.

Рабочее место преподавателя должно быть оснащено компьютером с подключенным к нему проектором или иным аналогичным по функциональному назначению оборудованием.

6 Методические рекомендации

6.1 Методические рекомендации для преподавателя по организации обучения

1. При подготовке к занятиям следует предварительно проработать материал занятия, предусмотрев его подачу точно в отведенное для этого время занятия. Следует подготовить необходимые материалы – теоретические сведения, задачи и др. При проведении занятия следует контролировать подачу материала и решение заданий с учетом учебного времени, отведенного для занятия.

2. При проверке работ и отчетов следует учитывать не только правильность выполнения заданий, но и оптимальность выбранных методов решения, правильность выполнения всех его шагов.

6.2 Методические указания для обучающихся по освоению дисциплины

Изучение дисциплины осуществляется в строгом соответствии с целевой установкой в тесной взаимосвязи учебным планом. Основой теоретической подготовки студентов являются лекции и самостоятельная работа.

В процессе самостоятельной работы студенты закрепляют и углубляют знания, полученные во время аудиторных занятий, готовятся к промежуточной аттестации, а также самостоятельно изучают отдельные темы учебной программы.

На занятиях студентов, в том числе предполагающих практическую деятельность, осуществляется закрепление полученных, в том числе и в процессе самостоятельной работы, знаний. Особое внимание обращается на развитие умений и навыков установления связи положений теории с профессиональной деятельностью будущего специалиста.

Самостоятельная работа осуществляется индивидуально. Контроль самостоятельной работы организуется в двух формах:

- самоконтроль и самооценка студента;
- контроль со стороны преподавателей (текущий и промежуточный).

Текущий контроль осуществляется на аудиторных занятиях.

Критериями оценки результатов самостоятельной работы студента являются:

- уровень освоения студентом учебного материала;
- умения студента использовать теоретические знания при выполнении практических задач;
- сформированность компетенций;
- оформление материала в соответствии с требованиями.

Приветствуется обсуждение самих заданий с другими студентами: можно как давать, так и получать советы по общей стратегии выполнения и изучения материала, давать и получать помощь в отладке. Однако писать код студент должен самостоятельно. Делиться кодом или писать его совместно запрещено.

7 Фонд оценочных средств

7.1 Методы контроля и оценивания результатов обучения

Приведенные ниже правила выставления оценок и опозданий могут быть изменены, если преподаватель сочтет это необходимым. Важно, чтобы студенты регулярно просматривали план курса, выложенный в СДО, на предмет его обновления или изменения.

Достижение компетенций оценивается с помощью лабораторных работ и рубежных контролей. Индикаторы ИПК-1.1, ИПК-1.2, ИПК-1.3 заложены в темах 1-12. Индикаторы ИПК-2.1, ИПК-2.2, ИПК-2.3 заложены в темах 1-12.

В соответствии с планом на дисциплины студентам выдаются задания на лабораторные работы. Помимо требований и описания функционала в работе указан крайний срок сдачи. Для сдачи лабораторной работы студенту необходимо прислать ссылку на репозиторий в GitHub Classroom и на хостинг, где размещен результат работы с реализованным функционалом, описанным в задании. Работа считается сданной если в ней реализовано 80% и более требований и функционала, описанного в задании.

Каждый студент имеет право на 6 дней опоздания, которые могут быть потрачены на любые задания в течение семестра. Опоздания предназначены для решения особых ситуаций, таких как болезнь или чрезвычайные семейные обстоятельства.

Когда использованы все дни опоздания за каждый день просрочки начисляется штраф в размере 25% от максимального результата за задание. Задания, присланные позже, чем 4 дня, не будут оцениваться. В связи с зависимостью между работами студентам может потребоваться все равно выполнить предыдущие работы, даже если они не оцениваются.

После сдачи лабораторной работы студент должен ее защитить. Во время защиты лабораторной работы преподаватель проверяет репозиторий, хостинг и выполнение критериев и требований задания, а студент отвечает на вопросы преподавателя по его коду, а также теоретических вопросов, приведенных после текста задания лабораторной работы. Если студент отказывается отвечать на вопросы, или дает полностью неверные ответы, или ответы не по теме, то работа может считаться сданной, но при этом она не оценивается.

Работа должна быть выполнена студентом самостоятельно: в репозитории в системе контроля версий студента содержатся коммиты только за его авторством, по этим коммитам можно проследить как велась работа, студент может объяснить свой код и ход выполнения работы, если эти правила не соблюдаются, то работа не считается сданной и не оценивается.

Рубежные контроли пишутся в аудитории индивидуально по варианту задания, выданному преподавателем в назначенные дни. При отсутствии студента в день написания контрольной работы ему дается еще один шанс ее написать на последнем занятии в семестре, но обязательно очно.

Студенты должны заранее сообщать о том, что у них могут возникнуть трудности со своевременной сдачей задания или проекта. При наличии реальных причин задержки студентам следует как можно скорее связаться с преподавателем и обсудить возможные условия.

7.2 Шкала и критерии оценивания результатов обучения

Лабораторная работа оценивается в процентах степени выполнения следующих критериев и для выставления оценки суммируются проценты за каждый из четырех критериев:

1. Полнота выполнения практического задания (30%): соответствует ли функциональность заданным требованиям и целям, насколько точно и без ошибок код выполняет поставленные задачи, насколько эффективно задание отвечает требованиям целевой аудитории и обеспечивает приятное восприятие.

2. Качество и структура кода (10%): качество, читаемость и организация кода, рациональность выполнения задания, последовательность именования и соблюдение лучших практик.

3. Творчество и инновации (10%): творческий подход студентов к выполнению заданий, насколько студенты вышли за рамки основных требований и реализовали дополнительные возможности или использовали уникальные решения.

4. Ответы на вопросы по коду студента и теории (50%):

Дает краткий ответ, содержащий ошибки или неточности. На наводящие вопросы отвечает неправильно (10% из 50%)

Дает развернутый ответ, содержащий ошибки или неточности. На наводящие вопросы отвечает неверно (20% из 50%)

Дает развернутый ответ, содержащий ошибки или неточности. На наводящие вопросы отвечает правильно (30% из 50%)

Дает правильные и развернутые ответы на вопросы (50% из 50%).

R лабораторные рассчитывается как среднее результатов за все лабораторные работы. За полное и безошибочное выполнение всех лабораторных работ в срок и их защиту можно получить максимум 100 баллов (R лабораторные).

Рубежный контроль оценивается по следующим критериям:

Полнота выполнения практического задания: соответствует ли функциональность заданным требованиям и целям, насколько точно и без ошибок код выполняет поставленные задачи.

Качество и структура кода: качество, читаемость и организация кода, рациональность выполнения задания, последовательность именования и соблюдение лучших практик.

Творчество и инновации: творческий подход студентов к выполнению заданий, насколько студенты вышли за рамки основных требований и реализовали дополнительные возможности или использовали уникальные решения.

Пользовательский опыт: отзывчивость, доступность, насколько эффективно задание отвечает требованиям целевой аудитории и обеспечивает приятное восприятие.

Самостоятельность решения: в репозитории студента есть коммиты только за его авторством, по коммитам в репозитории можно проследить как велась работа, студент может объяснить свой код и ход выполнения работы, если эти правила не соблюдаются, то работа не считается сданной.

Более подробное описание критериев дается в тексте задания рубежного контроля.

За полностью выполненные рубежные контроли также можно получить 100 баллов (R контроль).

Также имеется коэффициент сданных работ K сданные, который равен 1 если все работы сданы и 0 если хотя бы одна работа не сдана.

Итоговый балл рассчитывается по формуле: R сем = (0,5 x R лабораторные + 0,5 x R контроль) * K сданные.

Итоговый балл пересчитывается по шкале ниже и на основании полученной оценки фиксируется результат промежуточной аттестации.

Соответствие баллов в 100 балльной рейтинговой системе оценке по 4-бальной шкале:

0-54 - неудовлетворительно

55-69 - удовлетворительно

70-84 - хорошо

85-100 – отлично

7.3 Оценочные средства

7.3.1 Текущий контроль

Примерный список вопросов:

1. Какие типы значений есть в JS?
2. Чем отличаются строки "", ” и “?
3. Какие есть циклы в JS?
4. Как создаются функции?
5. Какие методы массивов вы знаете?
6. Как можно перебирать элементы массива?
7. Что такое объект?
8. Как получить значение из свойства объекта?
9. Что такое DOM?
10. Какие есть способы получения элементов из DOM?
11. Что такое событие?
12. Как добавить обработчик события?
13. Примеры свойств в объекте event.
14. Примеры методов у элементов DOM.
15. Назовите фазы жизненного цикла события.
16. Что такое метод объекта?
17. Что такое функция-конструктор?
18. Для чего используется ключевое слово new?
19. Как работает и на что указывает ключевое слово this?
20. Что такое WebGL?
21. Что такое canvas?
22. Какие объекты требуются для полноценной работы three.js?
23. Виды камер в three.js и их основные параметры.
24. Виды источников света и их основные параметры.
25. Работа с тенями и какие свойства объектов при этом задействуются.
26. Что можно/нужно указать для создания bufferGeometry?
27. Из чего состоит mesh в three.js?
28. Что такое node.js
29. npm - назначение и основные команды
30. package.json - назначение и основные свойства
31. Польза от .gitignore
32. Способы проверки работоспособности сервера
33. Создание минимального сервера на node.js
34. Объекты запроса и ответа - свойства, методы
35. Получение данных из тела запроса.
36. Что такое express?
37. В чем заключается концепция промежуточного ПО (middleware)?
38. Для чего используется static?
39. Что дает использование Router?
40. Как работает CommonJS, для чего нужны require, module.exports, exports.
41. В чем заключается концепция промежуточного ПО (middleware)?
42. Для чего нужна функция next в middleware?
43. Как устанавливаются middleware уровня приложения и уровня маршрутизатора?
44. Как установить middleware на отдельный путь и метод?
45. Какие параметры и как работает middleware предназначенный для обработки ошибок?
46. Какие есть отрицательные стороны у использования middleware?

47. Что такое Promise?
48. Для чего нужны методы then, catch, finally?
49. Как работает async/await?
50. В чем заключается идея документо-ориентированных бд?
51. Как создать подключение к MongoDB?
52. Какие методы из драйвера для node используются для чтения и добавления документов в MongoDB?
53. Для чего используются контроллеры и сервисы?
54. Что такое CRUD
55. Какие методы используются в mondodb для CRUD.
56. Какие параметры есть у методов для работы с документами в mondodb.
57. Как реализуется и работает обработка ошибок с помощью промежуточного ПО.
58. Объект Error в JS.
59. Что такое Fetch API
60. Какие есть свойства и методы у объекта Response в Fetch API
61. Как работает функция fetch()
62. На базе чего построен Fetch API
63. Как работает делегирование событий
64. Какие методы из DOM позволяют создавать и добавлять элементы
65. Что такое data-атрибуты и как их использовать

Пример задания рубежного контроля

Главные требования

1. Работа выполнена самостоятельно.
2. Работа написана во время занятия.
3. Не менее трех коммитов за авторством студента, равномерно распределенных по времени выполнения задания.
4. Работа размещена на удаленном репозитории на Github, который создан через Github Classroom. (Без этого работа не проверяется)

Задание

1. Принять задание по ссылке на Github Classroom
2. Клонировать репозиторий, который был создан после принятия задания.
3. Финальный вариант когда с историей работы, должен быть в этом репозитории.
4. Проект инициализирован с помощью команды npm init. Имя проекта RK, имя автора - ваше имя.
5. Для запуска сервера есть скрипт с именем start в файле package.json.
6. Проект содержит файл .gitignore с нужными исключениями для node.js.
7. Реализована серверная логика с использованием node.js и express по варианту.
8. Проект структурирован: routes, controllers, services, middlewares, configs.
9. Сервер запускается на порту 5500.
10. Все запросы логируются в консоль. Можно использовать стороннее промежуточное ПО.
11. Есть промежуточное ПО для обработки ошибок, возникающих на стороне сервера и отправляющее сообщению клиенту со статусом 500.
12. Есть промежуточное ПО, которое обрабатывает запросы на несуществующие конечные точки (end points) и отправляет ответ со статусом 400.
13. Есть промежуточное ПО проверяющее наличие заголовка Content-Type и что он содержит application/json, а также на наличие нужных данных в теле запроса согласно

варианту, в противном случае отправить ответ {status: "error", message: "Неверные данные"} с кодом 400.

14. Для хранения данных используется субд MongoDB, сервер которой работает на порту 27017.

Важно: Используйте имена свойств в объектах, такие как приведены в варианта, так как они будут использоваться для тестирования на работоспособность ваших работ.

Варианты

1. Книги

POST /authors - добавить нового автора.

Данные:

```
{
  name: string // имя автора
}
```

POST /authors/:id/books - добавить новую книгу автора.

Данные:

```
{
  title: string, // название книги
  year: number, // год издания
  series: string // серия
}
```

GET /authors/:id/books - получить все книги одного автора, сгруппированных по сериям.

Данные:

```
[
  {genre: string,[{name: string, year: number},{name: string, year: number}]},
  {genre: string,[{name: string, year: number},{name: string, year: number}]},
  …
]
```

2. Бонсай

POST /trees - добавить новое дерево.

Данные:

```
{
  type: string, // вид
  age: number, // возраст
  height: number // высота
}
```

GET /trees?key=value&key=value - получение деревьев, удовлетворяющим фильтрам из query-параметров. Если key нет среди полей документа (вид, возраст, высота) вернуть ошибку, в противном случае вернуть список удовлетворяющий критериям. Если совпадений нет, то вернуть пустой массив.

Данные:

```
[
  {type: string, age: number, height: number},
  {type: string, age: number, height: number}
  …
]
```

3. 3D-печать

POST /orders - добавить новый заказ.

Данные:

```
{
  name: string, // имя модели
  priority: [express, normal], // срочность: обычный, срочный
```

material: [pla, abs] // Вид пластика: PLA, ABS
}

GET /orders?sort=value - получить заказы, отсортированные по материалу или срочности. Если неверный query-параметр, то возвращается список отсортированный по имени.

Данные:

[
{name: string, priority: express, material: abs},
{name: string, priority: normal, material: pla},
…
]

4. Лакросс

POST /teams - добавить команду.

Данные:

{
teamName: string, // имя команды
city: string // город
}

POST /players - добавить игрока. Если команды нет в базе данных, то вернуть сообщение об ошибке, если есть, то добавить игрока к этой команде.

Данные:

{
teamName: string, // имя команды
playerName: string, // имя игрока
position: [off, def] // позиция игрока
}

GET /teams/:id - получить полную информацию о команде.

Данные:

{
teamName: string, // имя команды
city: string, // город
players: {def: number, off: number} // количество игроков по позициям
}

5. Космический туризм

POST /trips - добавить путешествие.

Данные: { destination: string, // направление spotNumber: number, // количество мест date: date // дата начала }

POST /trips/:id/passengers - добавить туриста к путешествию. Если все места уже заняты, то туриста не добавлять и вернуть сообщение об ошибке.

Данные:

{name: string // имя туриста}

GET /trips/:id - получить информацию о путешествии и список всех пассажиров.

Данные:

{
destination: string, // направление
spotsLeft: number, // количество оставшихся мест
passengers: [{name: string},{name: string}…] // список пассажиров
}