Министерство цифрового развития, связи и массовых коммуникаций РФ
Федеральное государственное бюджетное образовательное учреждение высшего образования
Уральский технический институт связи и информатики (филиал) ФГБОУ ВО "Сибирский государственный университет телекоммуникаций и информатики" в г. Екатеринбурге (УрТИСИ СибГУТИ)

Уральский технический институт связи и информатики

Информационные системы и технологии

КАФЕДРА
Информационных систем и технологий (ИСТ)

ОТЧЕТ

По дисциплине
«Сетевое программирование»
Практическое занятие №2
«Разработка API на node.js»

Выполнил: студент гр. ПЕ-11б
Хлебникова Е.А.

Проверил: преподаватель
Бурумбаев Д.И.

Екатеринбург, 2024

1 Цель работы:

1.1 Научиться работать с API;

1.2 Закрепить знания по теме «Знакомство с API».

3.Ход работы:

3.1 В начале необходимо создать базу данных, имя которой будет соответствовать № группы и фамилии обучающегося. В данной базе данных необходимо создать таблицу:

Имя базы данных – group_familia

Имя таблицы – users

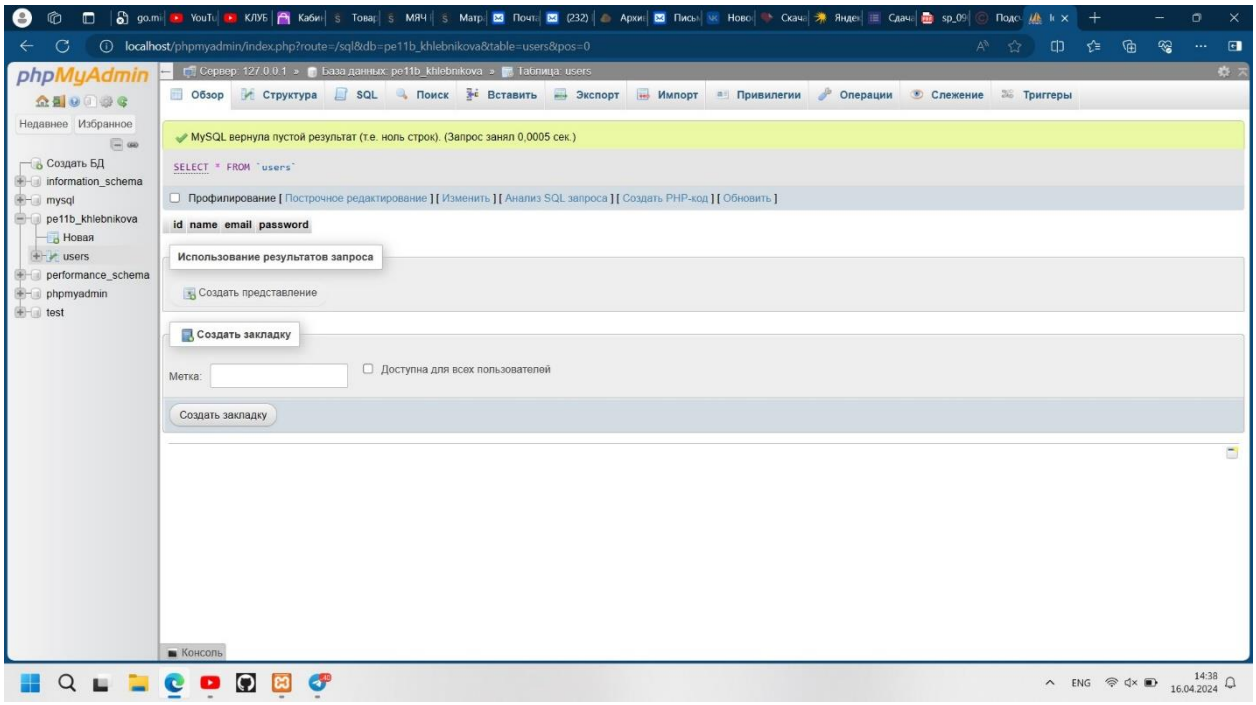

Рисунок 1 – Созданная таблица

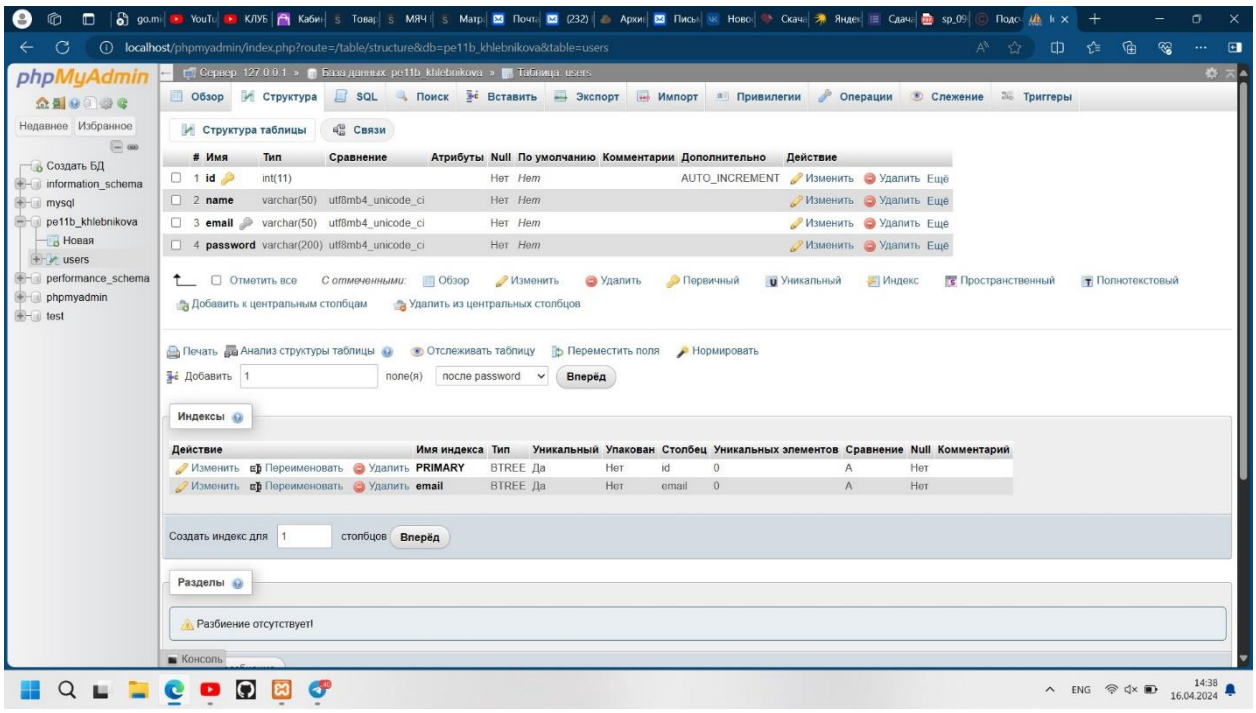

Рисунок 2 – Результат выполнения SQL-команды

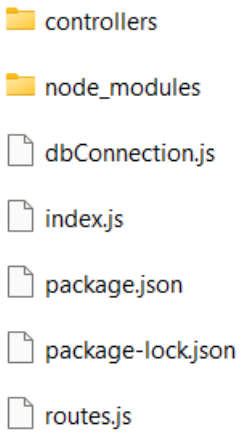

Рисунок 3 – Структура проекта

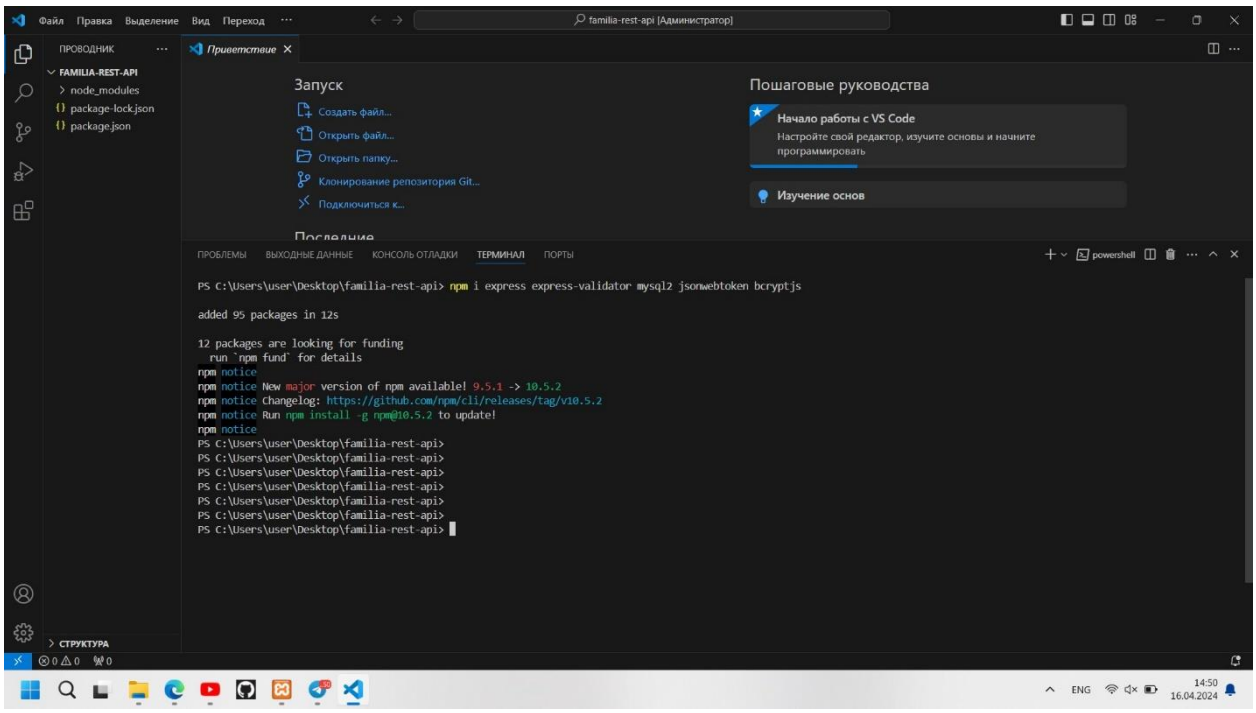

Рисунок 4 – установка пакетов

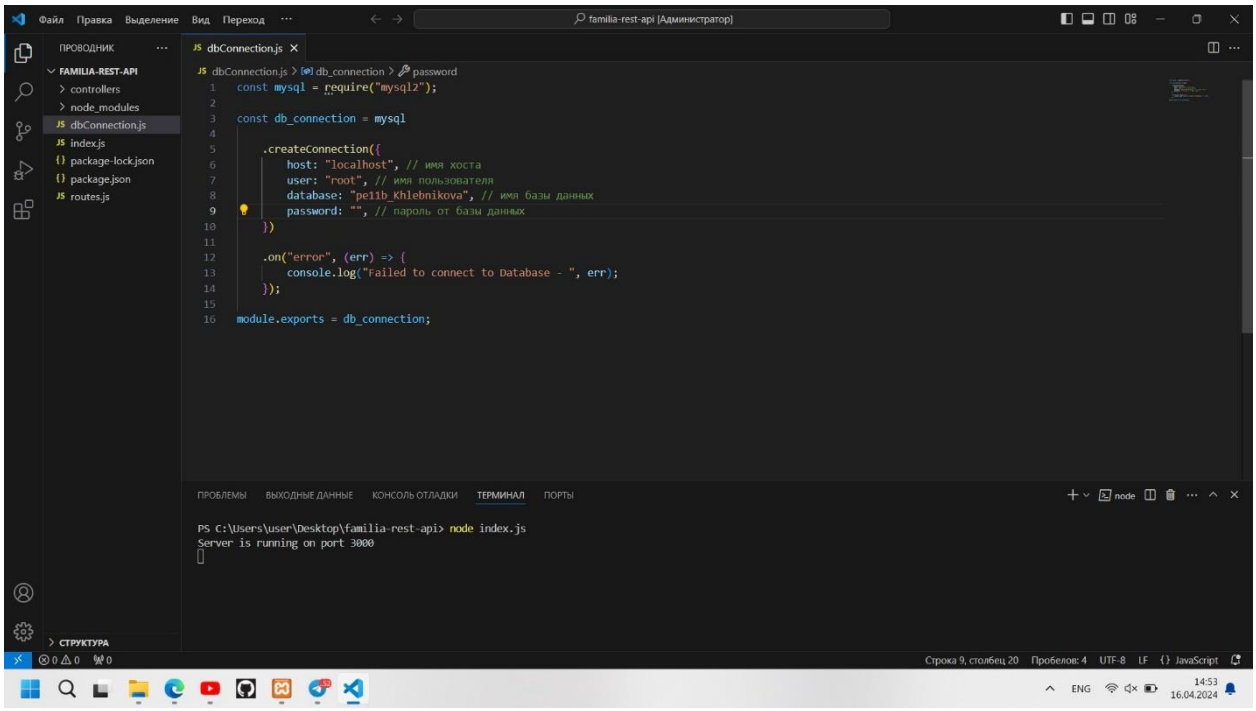

Рисунок 5 – Листинг Файл dbConnectrion.js

Индивидуальное задание:

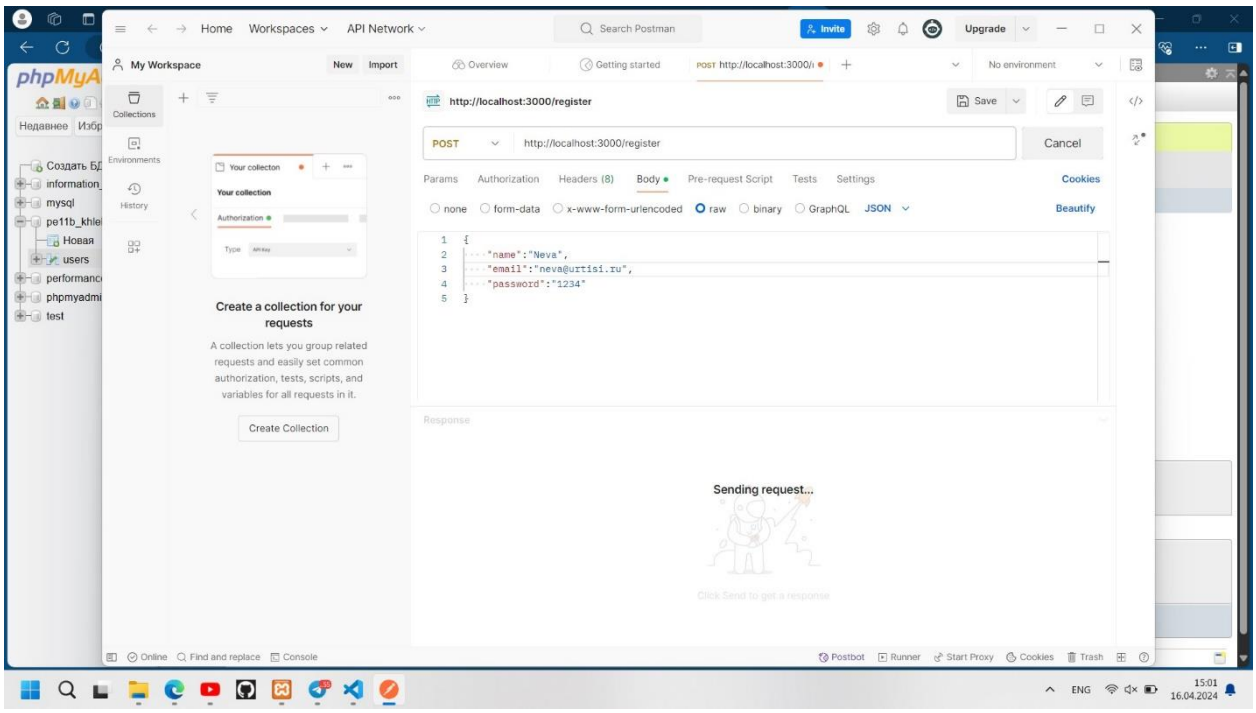

Рисунок 6 – Регистрация пользователя

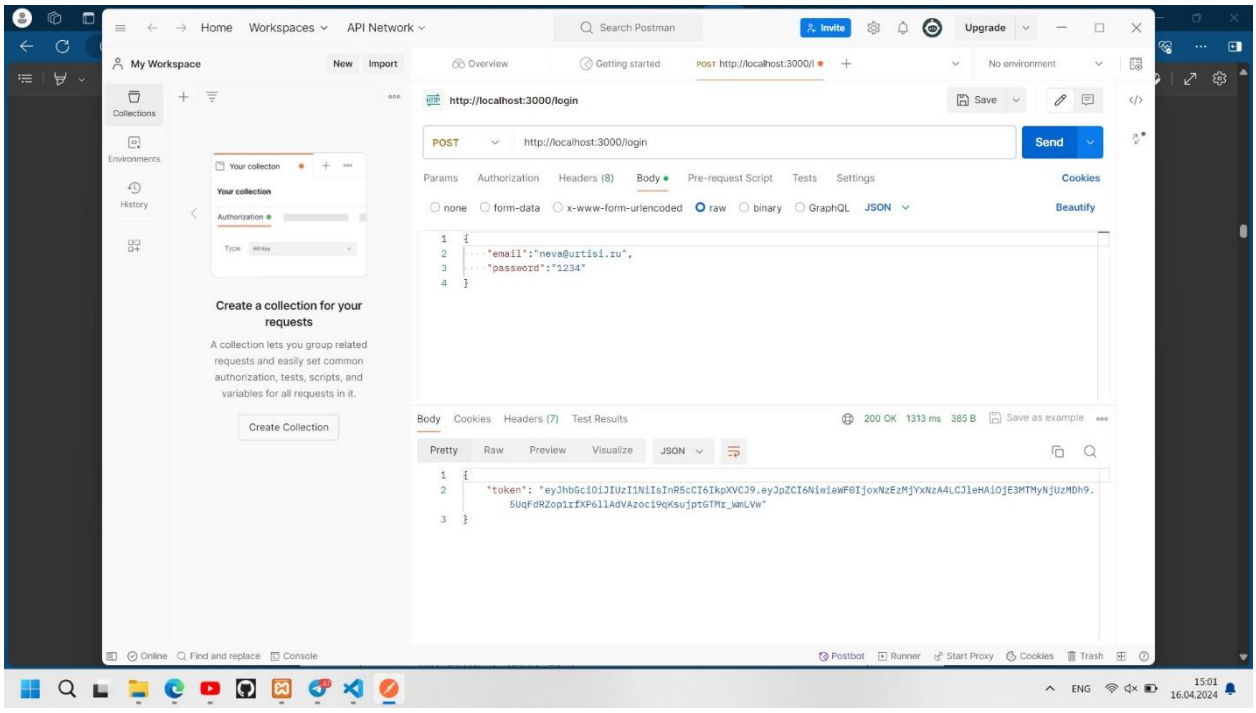

Рисунок 7 – Авторизация пользователя

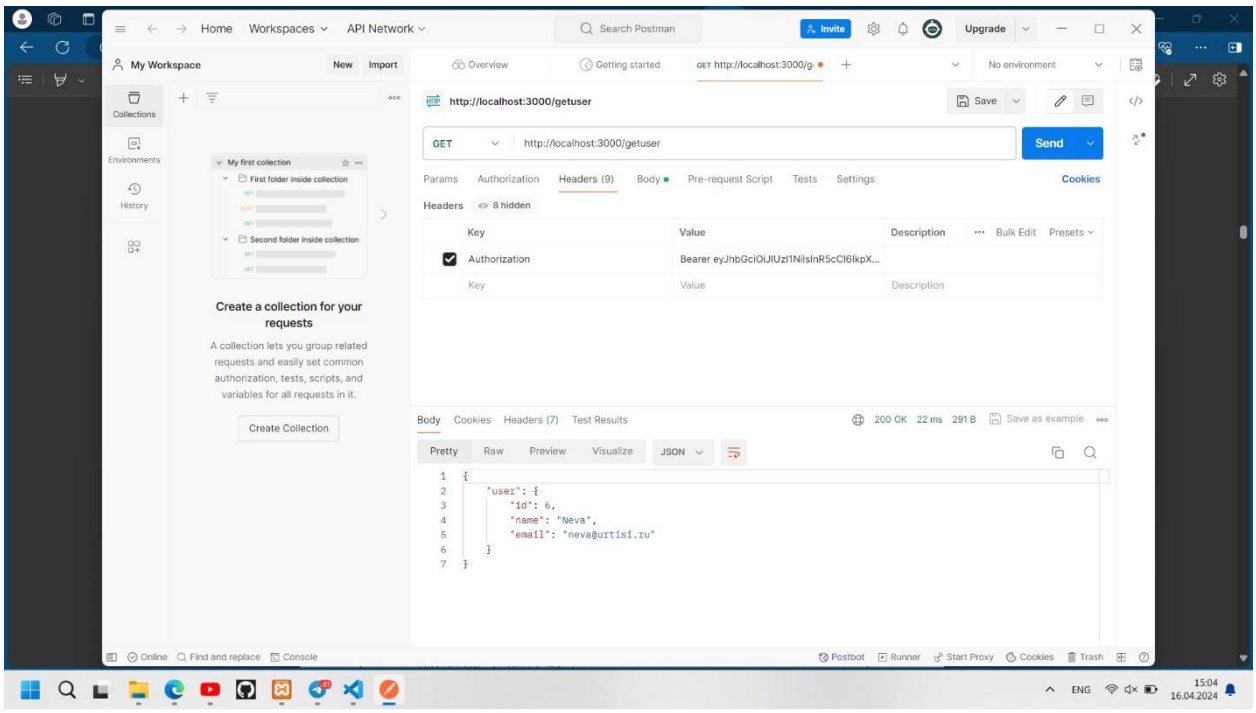

Рисунок 8 – Получение данных о пользователе

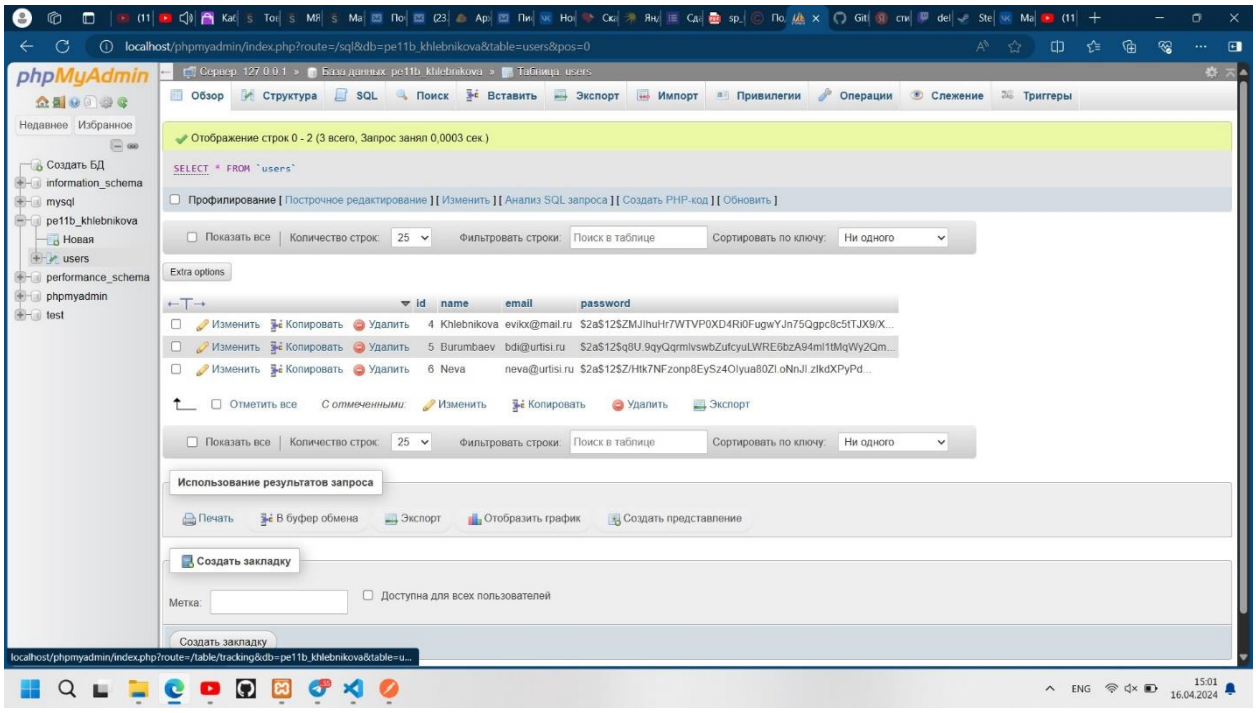

Рисунок 9 – Созданные пользователи

4. Контрольные вопросы:

4.1 Что такое API и какую роль оно играет в разработке программного обеспечения?
API (Application Programming Interface) - это набор определений, правил и протоколов, которые позволяют различным программным компонентам взаимодействовать друг с другом.
Роль в разработке программного обеспечения:
• Обеспечение взаимосвязанности: API позволяет разным системам и приложениям обмениваться данными и выполнять функции совместно.
• Повышение функциональности: Разработчики могут использовать API для интеграции функций сторонних приложений в свои собственные приложения.
• Повторное использование кода: API позволяет разработчикам повторно использовать код и модули, созданные другими, экономя время и усилия.
• Масштабируемость и гибкость: API облегчают масштабирование и модификацию приложений, позволяя разработчикам подключать или отключать компоненты по мере необходимости.
• Уменьшение сложности: Использование API позволяет разработчикам разделить сложные задачи на более мелкие части, создавая более управляемые и удобочитаемые системы.

4.2 Какие принципы проектирования API следует учитывать при его разработке?
• Ясность и лаконичность: API должно быть простым для понимания и использования.
• Согласованность: API должно использовать единый стиль именования, синтаксиса и документирования.
• Безопасность: API должно быть защищено от несанкционированного доступа и использования.
• Гибкость: API должно быть адаптируемым к изменяющимся требованиям и совместимым с будущими версиями.
• Документация: API должно иметь четкую и всеобъемлющую документацию.
• Тестирование: API следует тщательно тестировать, чтобы обеспечить его надежность и правильность.
• Обработка ошибок: API должно иметь надежный механизм обработки ошибок, который предоставляет детальную информацию об ошибках.
• Поддержка клиентов: API следует сопровождать отличной поддержкой клиентов.
• Постепенное внедрение: API следует внедрять постепенно, тестируя и развертывая его на небольшом масштабе перед полным развертыванием.
• Мониторинг и оптимизация: API следует постоянно отслеживать и оптимизировать для обеспечения высокой производительности и отклика.

4.3 Что такое RESTful API и какие основные принципы он соблюдает?

RESTful API (Representational State Transfer API) — это тип веб-сервиса, который следует архитектурному стилю REST (Representational State Transfer).
Основные принципы RESTful API:
• Ресурсы: Данные, к которым можно обращаться по адресу, называются ресурсами (например, /users или /products).
• Состояния: Состояние системы или приложения представлено ресурсами; взаимодействие с ресурсами меняет состояние системы.
• Представления: Ресурсы могут быть представлены в различных форматах (например, JSON, XML, HTML), называемых представлениями.
• HTTP-методы: RESTful API использует HTTP-методы (GET, POST, PUT, DELETE) для выполнения действий над ресурсами.
• Без состояния: RESTful API не хранит состояние сеанса на сервере. Состояние передается через запросы и ответы.
• Единообразные интерфейсы: RESTful API используют единообразные интерфейсы для управления ресурсами.

4.4 Какую роль играет формат данных (например, JSON или XML) при разработке API?
Формат данных играет важную роль при разработке API:

• Взаимозаменяемость: Различные форматы данных (например, JSON, XML, YAML) обеспечивают взаимозаменяемость при передаче данных между различными системами и приложениями.
• Описание данных: Форматы данных, такие как JSON и XML, предоставляют структурированное описание данных, что упрощает их интерпретацию и обработку.
• Эффективность: Некоторые форматы данных, такие как JSON, более эффективны, чем другие (например, XML), поскольку они занимают меньше места и быстрее передаются.
• Поддержка: Различные языки программирования и среды разработки предоставляют встроенную поддержку для общих форматов данных, что облегчает их интеграцию с API.
• Удобство для разработчиков: Некоторые форматы данных, такие как JSON, удобны для разработчиков благодаря их простому и понятному синтаксису.
• Валидация данных: Форматы данных, такие как XML, поддерживают встроенную валидацию данных, что помогает гарантировать целостность данных.
Выбор формата данных для API зависит от конкретных требований, таких как производительность, совместимость и удобство использования.

4.5 Какие механизмы аутентификации и авторизации могут использоваться в API?
Для защиты API и обеспечения доступа к ресурсам можно использовать различные механизмы аутентификации и авторизации:
Аутентификация:

• Ключи API: Уникальные ключи, предоставляемые разработчикам для идентификации их запросов.
• Токены OAuth: Временные токены, которые предоставляют доступ к ресурсам в течение определенного периода времени.
• HTTP Basic Auth: Механизм аутентификации, который использует имя пользователя и пароль, передаваемые в заголовке HTTP-запроса.
• Дайджест-аутентификация HTTP: Улучшенный механизм аутентификации, который отправляет хэш пароля вместо обычного текста.
• OpenID Connect: Протокол для делегирования аутентификации поставщику удостоверений.
Авторизация:
• Роли и разрешения: Назначение ролей и разрешений пользователям или клиентам для управления доступом к ресурсам.
• Контроль доступа на основе атрибутов (ABAC): Авторизация на основе атрибутов пользователей, таких как их роль, отдел или местоположение.
• Контроль доступа на основе ролей (RBAC): Авторизация на основе ролей, назначенных пользователям.
• Контрольный список доступа (ACL): Список пользователей или групп, которым разрешен или запрещен доступ к ресурсу.

4.6 Как можно предоставить документацию для API, чтобы облегчить работу разработчикам?
Хорошая документация API имеет решающее значение для облегчения работы разработчиков:
• Справочник по API: Всестороннее руководство по API, содержащее информацию о доступных конечных точках, параметрах, форматах ответа и примерах кода.
• Интерактивная документация: Документация, которая позволяет разработчикам взаимодействовать с API, просматривать запросы и ответы и пробовать различные параметры.
• Пошаговые руководства: Подробные руководства, которые ведут разработчиков через процесс интеграции с API.
• Примеры кода: Кодовые фрагменты на разных языках программирования, которые показывают, как использовать API.
• Часто задаваемые вопросы: Раздел, отвечающий на распространенные вопросы от разработчиков.
• Форум поддержки: Платформа для разработчиков для обсуждения вопросов и получения помощи.

4.7 Какие инструменты и технологии используются для разработки и тестирования API?
Разработка API:
• Swagger/OpenAPI: Фреймворк для определения и документирования API.
• Node.js/Express: Популярный стек для создания веб-приложений и API на JavaScript.

• Python/Django/Flask: Фреймворки Python для разработки веб-приложений и API.
• Java/Spring Boot: Фреймворк на основе Java для быстрой разработки микросервисов и API.
• C#/ASP.NET Core: Фреймворк Microsoft для создания веб-приложений и API на C#.
Тестирование API:
• Postman: Инструмент для тестирования API, отправки запросов и проверки ответов.
• SoapUI: Инструмент с открытым исходным кодом для функционального тестирования, нагрузочного тестирования и тестирования безопасности API.
• Jmeter: Инструмент для нагрузочного тестирования, который можно использовать для тестирования производительности API.
• Newman: Расширение командной строки для Postman, которое позволяет запускать тесты из терминала.
• Jest: Фреймворк тестирования JavaScript для тестирования как кода API, так и клиентских приложений.