Лабораторная работа № 4

Анализ и моделирование программной архитектуры и

архитектуры данных программного средства (средств) автоматизации ИТ-процессов

во взаимосвязи с функциональной и информационной архитектурой ИС.

М3315

Hoppscotch — инструмент для тестирования и взаимодействия с API

|  |  |  |
| --- | --- | --- |
| **Процесс** | **Описание** | **Информационные объекты** |
| **Сбор данных** | Получение данных от API через HTTP-запросы (GET, POST, PUT, DELETE и др.) | Запросы (URL, метод, заголовки, тело), ответы (статус-код, заголовки, тело) |
| **Обработка** | Валидация запросов, преобразование данных (JSON, XML), подстановка переменных | Параметры запроса, переменные окружения, скрипты (JavaScript для pre-request и тестов) |
| **Хранение** | Сохранение истории запросов, коллекций API, переменных окружения | Коллекции API, проекты, история запросов, конфигурации |
| **Передача** | Отправка запросов к серверам, проксирование, WebSocket-соединения | HTTP/HTTPS-запросы, WebSocket-сообщения, GraphQL-запросы |
| **Предоставление** | Визуализация ответов, экспорт данных (JSON, cURL), генерация документации | Отформатированные ответы, cURL-команды, документация API |

**Цели и показатели качества:**

* **Скорость обработки запросов** (время от отправки до получения ответа)
* **Удобство интерфейса** (простота создания и настройки запросов)
* **Поддержка стандартов** (HTTP/HTTPS, WebSocket, GraphQL, OpenAPI)
* **Безопасность** (поддержка OAuth, JWT, защита данных пользователя)

**Функциональные требования:**

* Поддержка REST, GraphQL, WebSocket.
* Возможность сохранения и организации коллекций запросов.
* Интеграция с системами аутентификации (OAuth, JWT).

**Нефункциональные требования:**

* Производительность: минимальные задержки при отправке запросов.
* Кроссплатформенность: работа в браузере и как PWA.
* Безопасность: шифрование локальных данных, защита токенов.

**A diagram of a computer

AI-generated content may be incorrect.**

**A diagram of a software application

AI-generated content may be incorrect.**

**A diagram of a flowchart

AI-generated content may be incorrect.**

Часть 3:

1. Сопоставление функциональной и программной архитектуры

|  |  |  |
| --- | --- | --- |
| Функциональный компонент | Функции | Программные компоненты |
| Интерфейс тестирования API | Отправка HTTP-запросов, просмотр ответов | Request.vue, request.service.ts, fetch.ts |
| Управление коллекциями | Создание, редактирование, удаление коллекций и запросов | collections.service.ts, IndexedDB API |
| Авторизация и безопасность | Добавление токенов, аутентификация, шифрование | auth.service.ts, env.service.ts |
| Поддержка WebSocket, GraphQL и SSE | Интерактивные соединения, подписка на события | realtime.service.ts, graphql.service.ts, ws.service.ts |
| Настройки окружений | Установка переменных окружения | env.service.ts, EnvironmentManager.vue, storage.ts |
| Управление пользователями | Вход, регистрация, синхронизация | firebase.service.ts, auth.service.ts |

2. Сопоставление информационной архитектуры и архитектуры данных

|  |  |  |  |
| --- | --- | --- | --- |
| Информационный объект | Структура данных / формат хранения | Хранение | Особенности реализации |
| Коллекция запросов | Объект с полями: id, name, requests[] | IndexedDB / Firebase | Уникальный id, поддержка вложенности, сериализация в JSON |
| Запрос | Объект: method, url, headers[], body, auth, envVars[] | JSON | Валидация типов данных, поддержка параметров из окружения |
| Окружение (env) | Объект: envId, name, variables. | IndexedDB / Firebase | Значения переменных подставляются в шаблоны запросов |
| Пользователь | Firebase User Object | Firebase Authentication | Используется безопасная авторизация, данные синхронизируются с базой |

**Целостность и идентификаторы:**

1. Уникальность объектов обеспечивается через UUID
2. Транзакционность частично реализуется при синхронизации с Firebase.
3. Целостность данных:
   * При сохранении в IndexedDB реализована валидация схем.
   * При работе с Firebase используется аутентификация и права доступа

**У нас:**

1. Клиент написан на Vue.js, сервер — на Node.js
2. Архитектура данных — в основном JSON-ориентированная, с возможностью работы offline и online.