**Документация по фронтенду**

Фронтенд приложения разработан с использованием **Vue 3** и следует принципам архитектуры **Feature-Sliced Design (FSD)**. Это позволяет организовать код в логические слои, что упрощает поддержку и масштабирование приложения. Основные функции включают управление персонажами, чатами, сообщениями, а также интеграцию с бэкендом через REST API и WebSocket.  
  
 На главной странице расположен **гайд** для пользователей сайта.

**Архитектура (Feature-Sliced Design)**

Архитектура FSD разделяет приложение на следующие слои:

1. **App Layer**:
   * Входная точка приложения (App.vue).
   * Настройка маршрутизации (router/index.ts).
   * Глобальные стили и конфигурации.
2. **Features Layer**:
   * **Auth**: Регистрация, вход и выход из системы.
   * **Character**: Управление персонажами (создание, редактирование, удаление).
   * **Chat**: Управление чатами и сообщениями.
   * **Home**: Главная страница с руководством пользователя.
3. **Shared Layer**:
   * **Components**: Переиспользуемые компоненты (например, AppHeader, BackButton).
   * **Api**: Клиент для работы с API (apiClient.ts).
   * **Utils**: Вспомогательные функции (например, работа с JWT).
4. **Widgets Layer**:
   * **ChatList**: Список чатов.
   * **ChatDetails**: Детали чата с возможностью отправки сообщений.

**Технологии**

1. **Vue 3**:
   * Основной фреймворк для разработки пользовательского интерфейса.
   * Используется Composition API для управления состоянием и логикой компонентов.
2. **Vue Router**:
   * Маршрутизация между страницами.
   * Защита маршрутов с помощью навигационных хуков.
3. **Vuex**:
   * Управление глобальным состоянием (например, аутентификация).
4. **Axios**:
   * HTTP-клиент для взаимодействия с бэкендом.
5. **WebSocket**:
   * Реализация реального времени для чатов.
6. **TypeScript**:
   * Добавляет статическую типизацию для повышения надежности кода.
7. **Vite**:
   * Сборщик и инструмент для разработки.
   * Обеспечивает быструю пересборку и горячую замену модулей (HMR).

**Компоненты и фичи**

1. **Аутентификация**:
   * **LoginForm**: Форма входа.
   * **RegisterForm**: Форма регистрации.
   * **LogoutButton**: Кнопка выхода из системы.
   * **Auth API**: Взаимодействие с бэкендом для регистрации и входа.
2. **Управление персонажами**:
   * **CharacterList**: Список персонажей с фильтрацией и пагинацией.
   * **CharacterDetails**: Детали персонажа с возможностью редактирования и удаления.
   * **CharacterCreate**: Форма создания персонажа.
   * **CharacterEdit**: Форма редактирования персонажа.
   * **Character API**: Взаимодействие с бэкендом для CRUD операций.
3. **Управление чатами**:
   * **ChatList**: Список чатов пользователя.
   * **ChatDetails**: Детали чата с возможностью отправки сообщений.
   * **Chat API**: Взаимодействие с бэкендом для создания и управления чатами.
   * **WebSocket**: Реализация обмена сообщениями в реальном времени.
4. **Главная страница**:
   * **HomePage**: Руководство пользователя с инструкциями по использованию приложения.
5. **Общие компоненты**:
   * **AppHeader**: Шапка приложения с навигацией.
   * **BackButton**: Кнопка возврата на предыдущую страницу.

**Примеры компонентов**

1. **CharacterList**:
   * Отображает список персонажей с возможностью фильтрации и пагинации.
   * Использует fetchCharacters для загрузки данных.
2. **ChatDetails**:
   * Отображает историю сообщений и позволяет отправлять новые сообщения.
   * Использует WebSocket для обмена сообщениями в реальном времени.
3. **LoginForm**:
   * Форма входа с валидацией.
   * Использует login из Auth API для аутентификации.

**Примеры API**

1. **Auth API**:
   * register: Регистрация нового пользователя.
   * login: Вход в систему.
   * logout: Выход из системы.
2. **Character API**:
   * fetchCharacters: Получение списка персонажей.
   * createCharacter: Создание нового персонажа.
   * updateCharacter: Редактирование персонажа.
   * deleteCharacter: Удаление персонажа.
3. **Chat API**:
   * createChat: Создание нового чата.
   * fetchUserChats: Получение списка чатов пользователя.
   * sendSocketMessage: Отправка сообщения через WebSocket.

**Примеры маршрутов**

1. **BaseRoutes**:
   * /: Главная страница с гайдом
   * /example: Пример страницы для тестов.
2. **AccountRoutes**:
   * /login: Страница входа.
   * /register: Страница регистрации.
3. **CharacterRoutes**:
   * /characters: Список персонажей.
   * /characters/create: Создание персонажа.
   * /characters/:id: Детали персонажа.
   * /characters/edit/:id: Редактирование персонажа.
4. **ChatRoutes**:
   * /chats: Список чатов.
   * /chats/:chatId: Детали чата.