|  |  |
| --- | --- |
|  | УТВЕРЖДАЮ |
|  | Руководитель Учебной практики  \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Долженкова М. Л.  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г.  М.П. |

ТЕХНИЧЕСКАЯ ДОКУМЕНТАЦИЯ

приложения «Steam Achievment view»

|  |
| --- |
| СОГЛАСОВАНО |
| Колледж ВятГУ  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г.  М.П. |
|  |
|  |

2023

1. **Описание проекта:**

Название проекта: Steam Achievements Analyzer.

Цель проекта: Создание web-приложения, которое позволит пользователям анализировать и просматривать свои достижения в играх Steam.

1. **Основные функциональные возможности:**

* аутентификация пользователя с использованием Steam API.
* получение списка игр пользователя с использованием Steam API.
* отображение достижений пользователя для выбранной игры.
* поиск игр по названию или жанру.
* отображение детальной информации о достижении.
* возможность сортировки достижений по дате получения, по статусу (завершенные/незавершенные) или по названию игры.
* возможность просмотра профиля пользователя и его статистики в Steam.

1. **Уточнение структуры данных:**

Таблица 1 – Сущности и атрибуты данных

|  |  |
| --- | --- |
| Сущность | Атрибуты |
| Пользователь | Идентификатор пользователя (Steam ID) |
| Имя пользователя |
| Аватар пользователя |
| Игры пользователя (список игр) |
| средний процент достижений |
| Игры | Идентификатор игры (App ID) |
| Название игры |
| Статистика пользователя игры |
| Достижения пользователя в игре |
| Обложка игры |
| Достижение | Идентификатор достижения (Achievement ID) |
| Название достижения |
| Описание достижения |
| Статус достижения (завершено/незавершено) |
| Дата получения достижения |
| Процент получения среди игроков |
| Иконка завершенного достижения |
| Иконка не завершенного достижения |

1. **Формы представления данных**

Контекстная диаграмма приложения для достижений Steam представляет собой высокоуровневую иллюстрацию взаимодействия данного приложения с его внешней средой. Контестная диаграмма представлена на рисунке 1.

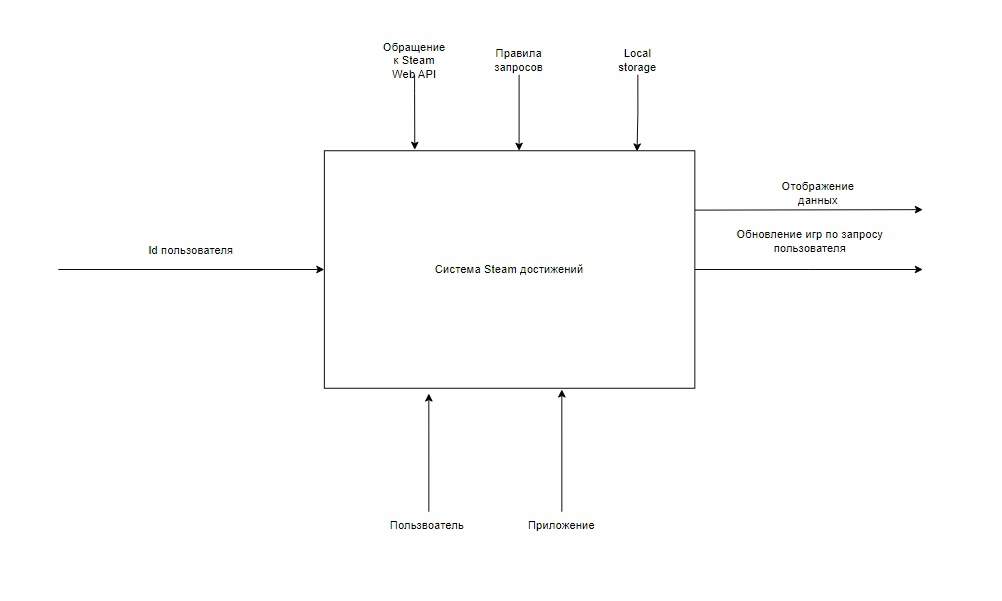


Рисунок 1 – Контекстная диаграмма

Ниже приведено описание основных элементов контекстной диаграммы:

Пользователь: Отображает всех пользователей Steam, которые могут взаимодействовать с приложением для достижений. Пользователи могут быть как обычными игроками, так и разработчиками игр.

Steam API: Представляет собой интерфейс программирования приложения Steam, который позволяет приложению для достижений получать доступ к информации о пользователях, играх, достижениях и других функциях Steam.

Local Storage: Хранит всю информацию о пользователях, играх, достижениях и других данных, связанных с Steam. Приложение для достижений может получать и обновлять данные из базы данных Steam.

Приложение для достижений Steam: Основное приложение, которое предоставляет пользователям возможность просматривать и отслеживать достижения, полученные в играх Steam.

Для разбиения сложного процесса на составляющие подфункции применяется декомпозиция, данная диаграмма представлена на рисунке 2.

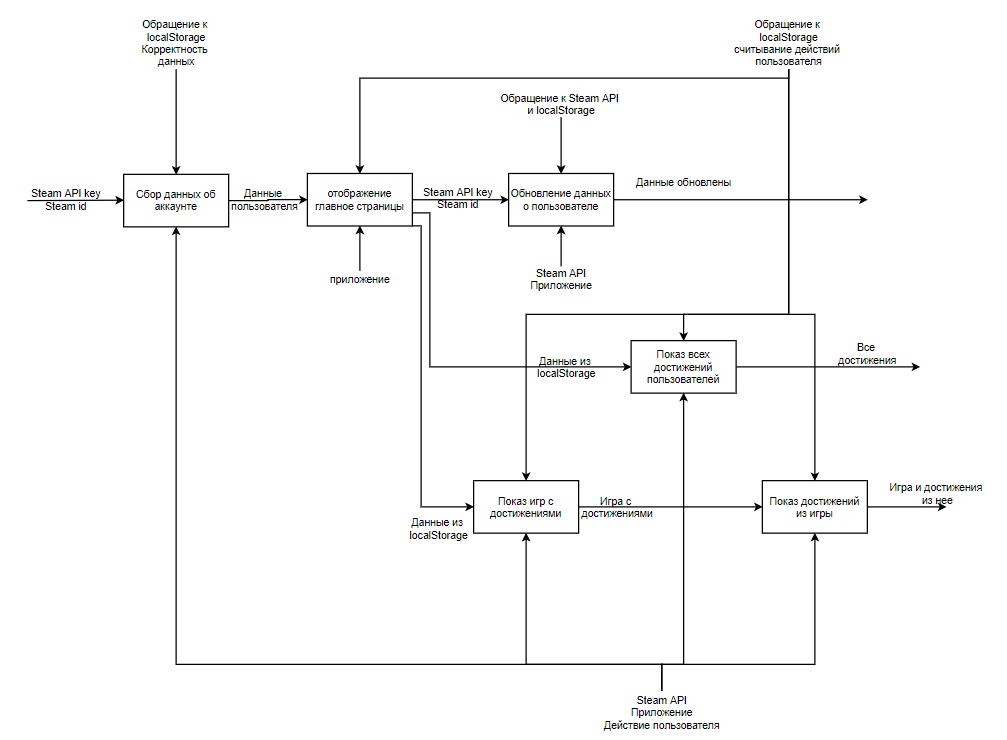


Рисунок 2 – Диаграмма IDEF декомпозиция

Поток данных, происходящих в данной программе представлен на рисунке 3.e

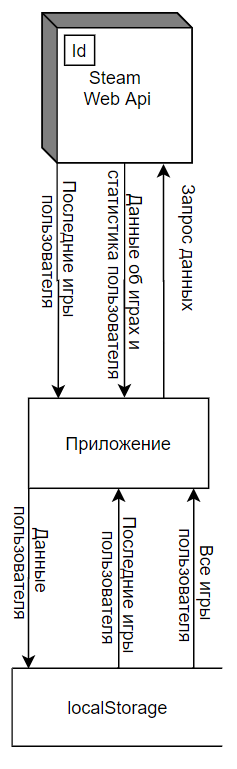


Рисунок 3 – Диаграмма DFD

Диаграммы состояний компонентов страницы отображена на рисунках 4 и 5 соотвественно.

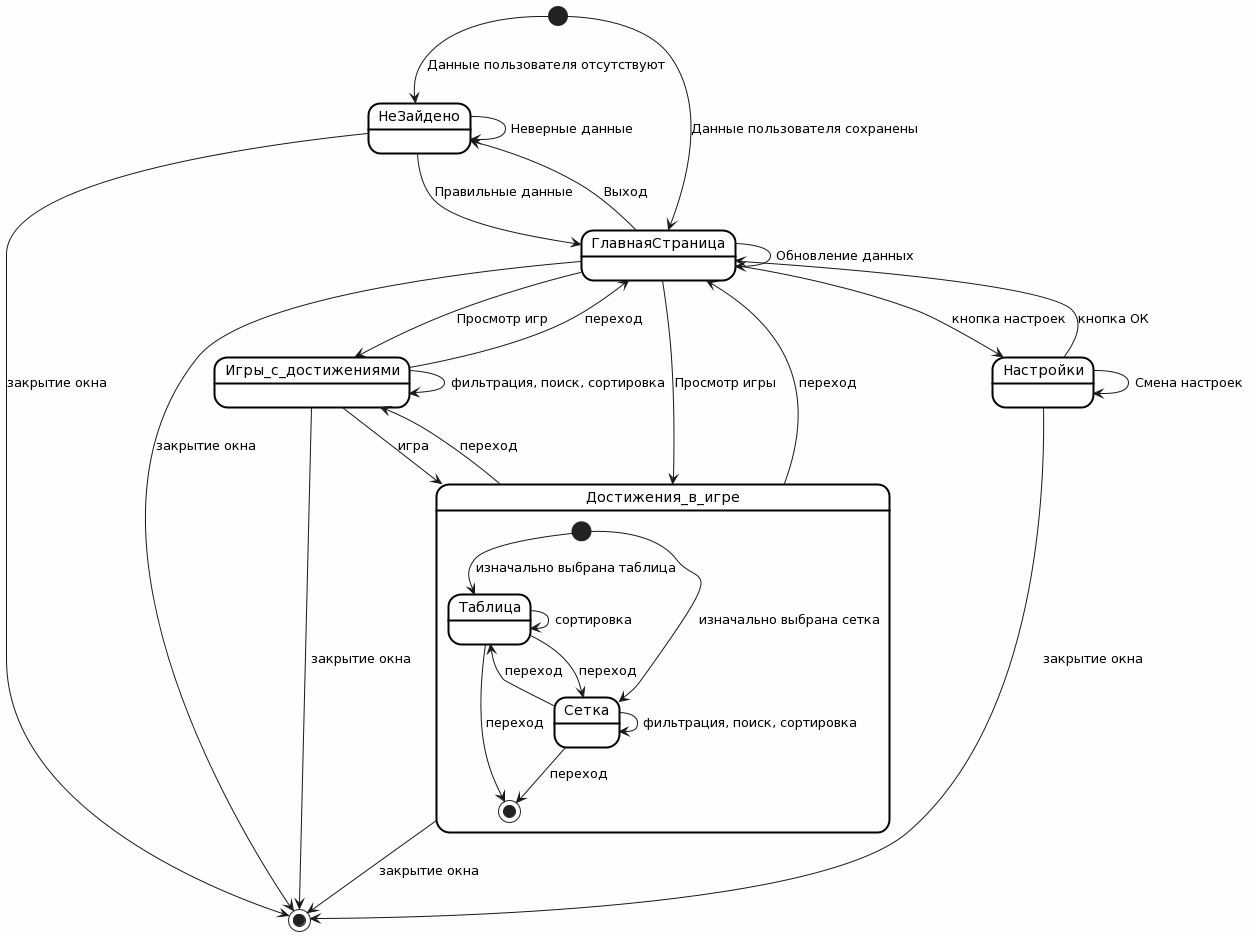


Рисунок 4 – Диаграмма состояний

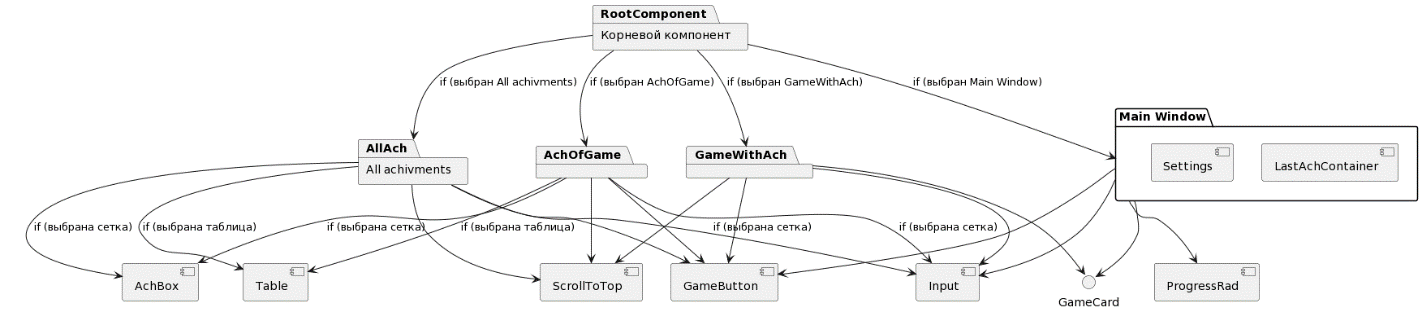


Рисунок 5 – диаграмма компонентов

Данные пользователя отображаются на главной странице в виде колонки данных:

* имя;
* аватар;
* количество игр;
* количество достижений;
* средний процент достижений;

Данные об игре представлены в виде карточке. На карточках находятся название игры, обложка игры, статистика пользователя в игре (Время игры, все достижения в игре, полученные достижения, не полученные достижения, процент достижений), и несколько последний достижений (полученных если возможно)

Последние 36 полученных достижений отображаются в квадратной контейнере в виде блока 6 на 6, при наведении на достижение отображается название игры, имя достижения, описание достижения, процент игроков, получивших достижения и время получения. Экранная форма главного окна представлена на рисунке 6.

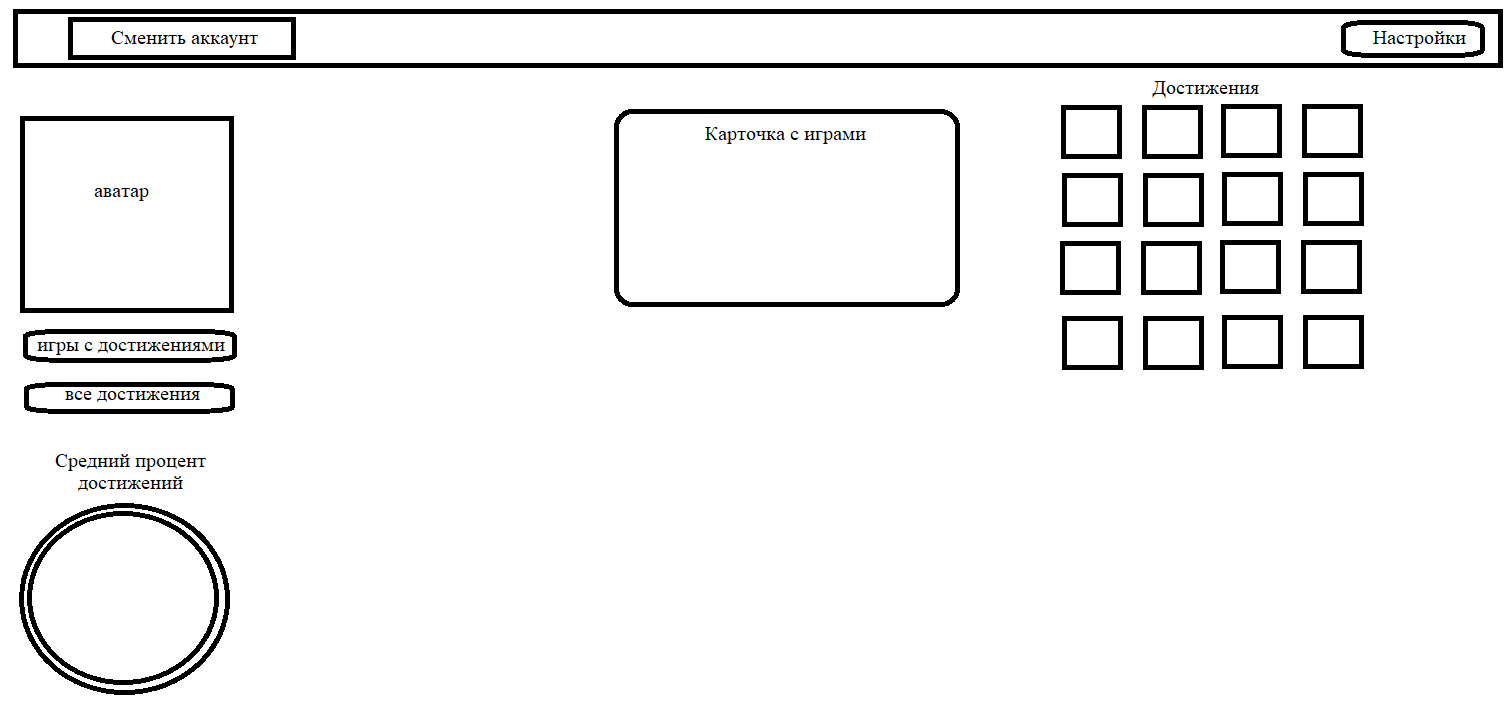


Рисунок 6 – Экранная форма главного меню

Достижения в игре или все достижения пользователя отображаются в виде таблицы, если просматриваются все достижения пользователя, то надписи открыто нет.

Для сортировки и фильтрации игр присутствует контейнер, в котором есть возможность ввести название игры (при вводе скрываются неподходящие игры в реальном времени). Для сортировки необходимо нажать на кнопку сортировать, после чего выпадет список с выбором сортировки. В контейнере есть стрелки для выбора направления сортировки. Для фильтрации есть кнопки игровое время и завершенность, которые предоставляют выпадающее списки с выбором соответствующей фильтрации.

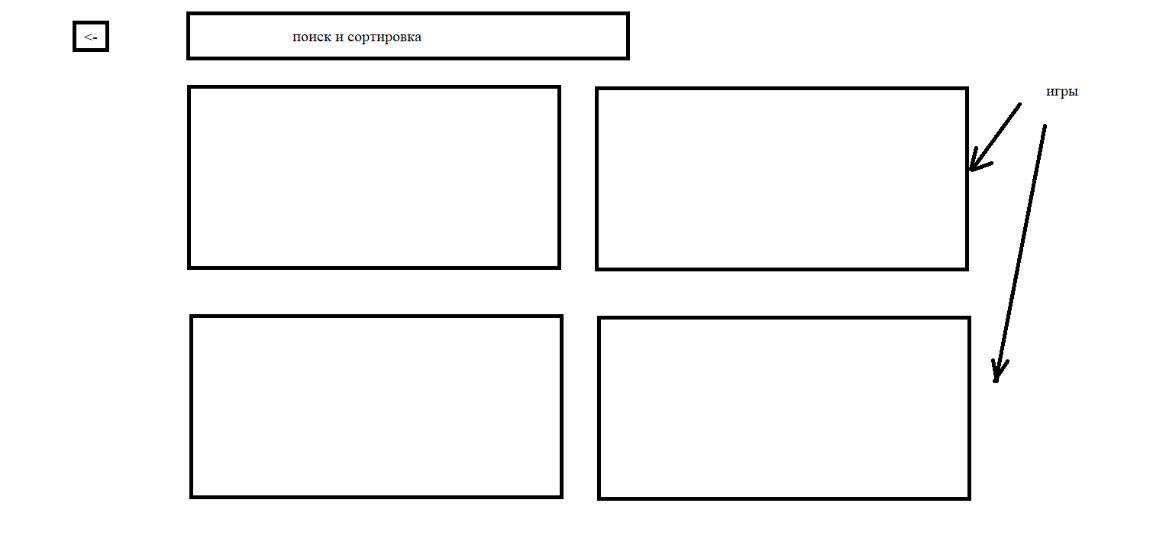


Рисунок 7 – Экранная форма окна с играми

1. **Разработка алгоритма решения задачи:**

Для выполнения аутентификации пользователя и получения данных о его играх, вам потребуется использовать Steam API и соответствующие методы.

1. Аутентификация пользователя:
   1. Пользователь должен предоставить Steam API ключ и SteamId. Для этого нужно использовать два поля ввода.
   2. После успешной аутентификации, нужно использовать полученный SteamId для запроса данных о пользователе через Steam API.
2. Получение и обработка игр пользователя:
   1. Используйте Steam API для получения списка игр пользователя, используя его SteamId.
   2. Затем, используется Steam API для получения всех данных об достижениях в играх пользователя, используя SteamId и идентификаторы игр.
   3. Обработка полученных данных о достижениях, чтобы получить процент достижений, количество достижений и количество полученных достижений для каждой игры.
   4. Соберите все эти данные в один цельный JSON объект.
3. Сохранение данных в localstorage:
   1. Используется методы localstorage для сохранения полученных данных в браузерном хранилище.
   2. Например, используется метод localStorage.setItem('data', JSON.stringify(data)) для сохранения данных в localstorage.

При реализации необходимо обратить внимание на безопасность и проверку ошибок при запросах к Steam API. Также учесть, что Steam API может иметь ограничения по количеству запросов в определенный период времени, поэтому рекомендуется использовать кэширование данных или ограничить количество запросов.

1. **Определение языка, структуры программы и требований к техническим средствам:**

Язык программирования: был выбран TypeScript из-за типизации, а также из-за использования Electron Forge.

Фреймворк: был выбран React.js из-за его реактивности и возможности обновления компонентов, без необходимости в обновлении всей страницы.

Сборка и упаковка приложения: Electron Forge, что позволяет использовать приложение на многих операционных системах.

Хранение данных: используется localstorage из-за небольшого количества хранимых данных и доступности к ним из любого места.

Взаимодействие с API Steam: используется Steam Web API из-за отсутствия альтернатив.

Использование сторонних библиотек: исльзуется i18next для локализации, react-loading-overlay-ts для анимации загрузки, react-spinners/BounceLoader для определенного вида загрузки, react-toastify для оповещения пользователя, Express для создания локального сервера (чтобы обходить ограничения CORS политики).

1. **Требования к техническим средствам:**

Клиентская часть:

* доступ к интернету для загрузки данных из API Steam;
* компьютер