МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Министерство образования и науки Российской Федерации

«Национальный исследовательский технологический университет «МИСИС» информационных технологий и компьютерных наук

Кафедра Инженерной Кибернетики

**Пояснительная записка к курсовой̆ работе  
по дисциплине "Разработка клиент-серверных приложений"**

**на тему: “Смартап Английский крокодил”**

Выполнили:

Ишмухамедов Артём Альбертович, БПМ-21-1

Цыбикжапов Даши Юрьевич, БПМ-21-1

Бабенко Данила Анатольевич, БПМ-21-1

Проверил:

Лыкошин А.С.

Москва, 2023

**АННОТАЦИЯ**

Курсовая работа состоит из: введения, постановки и анализа задачи, описания проектных решений, заключения и используемой̆ литературы.

Постановка и описание задачи представляет полное описание смартапа: категорию и целевую аудиторию приложения, иллюстрации превью приложения и примеры запуска. Краткую инструкцию по тестированию приложения.

Описание проектных решений включает в себя обзор всех используемых компонентов, их взаимодействие. Даны некоторые ссылки на источники разработки.

Ключевые слова: смартап, ассистент, SmartMarket, разработка, клиент-сервер.

Оглавление

[**ВВЕДЕНИЕ** 4](#_Toc137835305)

[**Постановка и анализ задачи** 5](#_Toc137835306)

[**Описание проектных решений** 10](#_Toc137835307)

[**Фронтенд** 10](#_Toc137835308)

[**Бэкенд** 11](#_Toc137835309)

[**Описание взаимодействия компонентов** 15](#_Toc137835310)

[**Использованные инструменты и технологии** 16](#_Toc137835311)

[**ЗАКЛЮЧЕНИЕ** 17](#_Toc137835312)

[**ИСТОЧНИКИ** 18](#_Toc137835313)

## **ВВЕДЕНИЕ**

Разработанный нами смартап называется «Английский крокодил».

Английский крокодил — это веселый и эффективный способ изучения английского языка. Благодаря инновационным функциям и интерактивному игровому процессу, игроки могут научиться новым словам, развить свою лексику и улучшить навыки общения на английском языке, и все это в увлекательной и захватывающей игровой форме.

Запустить смартап пользователь может следующими способами:

1. Кнопкой в магазине приложений SmartMarket;
2. Голосовыми командами:

* Запусти «Английский крокодил»;
* Открой «Английский крокодил»;
* Включи «Английский крокодил»

используя устройство SberBox.

## **Постановка и анализ задачи**

Наша команда проанализировала возможности устройств SberBox и пришла к выводу о том, что данное устройство, находясь все время дома, будет использоваться людьми как для личного пользования, так и в компании людей. Поэтому мы поставили следующую задачу: разработать для данной целевой аудитории смартап, который будет не только подавать развлекать людей, но и во время увлекательной игры приносить пользу. Воплощением данной задачи стало приложение «Английский крокодил», получившее возрастное ограничение «16+» и вошедшее в категории «Словесные» и «Игры для компании».

Теперь рассмотрим приложение по порядку, с точки зрения пользователя.

Как уже было сказано выше, запустить смартап возможно несколькими способами:

1. Кнопкой в магазине приложений SmartMarket;
2. Голосовыми командами:

* «Запусти Английский крокодил»;
* «Открой Английский крокодил»;
* «Включи Английский крокодил»;

После запуска загружается главное меню смартапа (рис. 1), голосовой ассистент говорит приветственное слово и кратко описывает функционал приложения.

Изображение выглядит как текст, снимок экрана, зеленый

Автоматически созданное описание

Рис. 1. Интерфейс главного меню для SberBox

В основном меню пользователю даётся выбор: зайти в помощь или сразу начать играть. Зайти в помощь также можно сделать с помощью ассистента (например, фразой “Правила игры”) и начать играть также с помощью ассистента (например, фразой “Новое слово”)

Если пользователь решает сперва узнать правила игры и нажимает на кнопку “Помощь, то появляется окно помощи с правилами (рис. 2). Когда пользователь начинает играть, ему следует закрыть окно помощь кнопкой “Закрыть” и нажать на кнопку “Новое слов”, на экране появляется выбранное случайным образом слов из списка 800 английских слов (рис. 3). Далее по правилам игры один участник игры показывает слово остальные участники стараются угадать слово. Если слово отгадано, то следует нажать на кнопку “Угадано”, в таком случае счётчик “Угадано” увеличиться на 1 и счётчик “Всего” увеличиться на 1 (рис. 4), а на экране появится новое случайно выбранное слово. Если же слово не отгадано, то следует нажать на кнопку “Не угадано”, в таком случае счётчик “Угадано” останется без изменений и счётчик “Всего” увеличиться на 1 (рис. 5), а на экране появится новое случайно выбранное слово. Все эти функции также осуществляются с помощью голосового помощника следующими фразами:

* «Справка/Помоги/Помощь/Как играть/Научи/Почему/Что делать/Что умеешь/Правила»
* «Закрой помощь/Скрой помощь/Закрой справку/Скрой справку/Закрой мануал/Скрой мануал/Убери помощь/Убери справку/Убери мануал/Понял/Ясно»
* «Покажи слово/Новое слово/Покажи новое слово/Другое слово/Поменяй слово/Смени слово/Покажи другое слово/Следующее/Следующее слово»
* «Угадал/Угадано/Выиграл/Отгадал/Отгадано/Выиграно/Да/Правильно/

Получилось»

* «Не угадал/Не угадано/Не выиграл/Не отгадал/Не отгадано/Не выиграно/Нет/Неправильно»

Изображение выглядит как текст, снимок экрана, письмо, Шрифт

Автоматически созданное описание

Рис. 2. Интерфейс окна помощи

Изображение выглядит как текст, снимок экрана, зеленый, Шрифт

Автоматически созданное описание

Рис. 3. Интерфейс главного меню с новым словом

Изображение выглядит как текст, снимок экрана, зеленый, Шрифт

Автоматически созданное описание

Рис. 4. Интерфейс главного экрана при первом угаданном слове

Изображение выглядит как текст, снимок экрана, зеленый, Шрифт

Автоматически созданное описание

Рис. 5. Интерфейс главного экрана при втором не угаданном слове

## **Описание проектных решений**

### **Фронтенд**

Фронтенд нашего приложения написан на языке JavaScript и TypeScript с использованием фреймворка Svelte, языка гипертекстовой разметки HTML и стилей CSS. Работа и дизайн описаны в файлах App.svelte, Help.svelte, index.html, main.ts, global.css

* app/src/App.svelte - основная работа приложения
* app/src/Help.svelte - работа экрана помощи
* app/src/index.html – файл запускающий разметку страницы
* app/src/main.ts – конфигурационный файл запускающий App.svelte
* app/public/global.css – файл со стилями распространяющийся на все файлы приложения

Кратко рассмотрим некоторые функции и переменные приложения:

* Функция onMount вызывает callback, когда компонент монтируется в DOM. Это эквивалентно хуку useEffect из React.
* Функция call\_help - запускает определенное действие с идентификатором 'help'(Открывает помощь)
* Функция news – для определения(рандомно) слов из словаря
* Функция gcount - увеличивает значения best и score объекта game на 1.
* Функци badcount - увеличивает значение best объекта game на 1.
* Функцию typewriter - выполняет анимацию появления текста посимвольно.
* Функция changeword отправляет данные с действием с идентификатором 'changeword' через объект assistant.(поменять слово)
* Функция guessedright отправляет данные с действием с идентификатором 'guessedright' через объект assistant, только если переменная visible равна true.(угадано верно)
* Функция guessedwrong отправляет данные с действием с идентификатором 'guessedwrong' через объект assistant, только если переменная visible равна true.(угадано неверно)
* Объект game с двумя свойствами: score и best, оба изначально равны 0.

### **Бэкенд**

Для реализации серверной части в приложении использован фреймворк Dialute для взаимодействия с голосовым помощником, Svelte, TypeScript, JavaScript. Вся серверная часть хранится в hook/src/index.ts

Функции, используемые в бэкенде:

Функция handleClick, которая вызывается по нажатию на кнопку. В ней задействуется объект клиента, который мы ранее инициализировали, вызвав createSmartappDebugger. Этот клиент может отправлять события в наш сценарий, каждый раз при вызове метода sendData. Нажатие на кнопку - лишь частный случай такого триггера. Чтобы обработать событие внутри сценария, прочитаем поле type, оно должно иметь значение SERVER\_ACTION. Только после этого мы можем проверить наличие поля r.act.action\_id и посмотреть является ли оно “click”. Таким образом, можно отправлять события и данные из фронтенда в бекенд. Для полноценного взаимодействия нужна двyxcтopoнняя связь, поэтому используется объект Response (const rsp = r.buildRsp();)) Чтобы отправлять расширенный ответ пользователю используем “yield rsp;”. Мы отправили во фронтенд объект state. Теперь мы хотим его прочитать и вывести в интерфейсе. Для этого используем событие “data” и добавим проверку на тип события smart\_app\_data. Благодаря подписке на событие “data”, переменная state на фронтенде будет обновляться каждый раз, когда с бекенда приходят данные. Если вы используете React, то вам необходимо использовать хук useState и внутри условия обновлять данные при помощи setState(event.smart\_app\_data)

Функция textToCommand - принимает массив строк texts и возвращает объект с типом команды на основе содержимого текстов. Внутри функции определены массивы ключевых слов и фраз, такие как smartapp, changemode, changeword, helps, greets, restarts, guessedright, guessedwrong, close, и value.

Генераторная функция script - принимает объект “r” с информацией о запросе. Внутри функции определены переменные count и rsp. Переменная rsp инициализируется с использованием метода buildRsp() объекта r, который создает новый объект ответа.

Затем определены массивы фраз для различных сценариев:

changewordPhrases: фразы для показа нового слова.

maleofficialFailPhrases, maleno\_officialFailPhrases, femaleofficialFailPhrases, femaleno\_officialFailPhrases: фразы для ситуации, когда команда не распознана или неизвестна. GuessedRightPhrases: фразы для случая, когда ответ угадан правильно.GuessedWrongPhrases: фразы для случая, когда ответ угадан неправильно.

Цикл (while (true)), который выполняет обработку запросов и генерацию ответов на основе условий.

Внутри цикла происходит следующее:

Создается новый объект ответа rsp с использованием метода buildRsp() объекта r.

Устанавливается ответ rsp.kbrd в виде массива с единственным элементом "Оценить".

Происходит деструктуризация свойств gender и appeal из объекта r.body.payload.character.

Выполняется проверка типа запроса (r.type) и значения свойства action\_id объекта r.act.

Если тип запроса SERVER\_ACTION и action\_id равно "help", то устанавливается тип данных rsp.data в "help", а rsp.msg устанавливается пустой строкой.

Если тип запроса SERVER\_ACTION и action\_id равно "restart", то rsp.msg устанавливается как "Начинаю заново", а тип данных rsp.data устанавливается как "restart".

Если тип запроса SERVER\_ACTION и action\_id равно "changeword", то устанавливается тип данных rsp.data в "changeword", а rsp.msg устанавливается случайной фразой из массива changewordPhrases.

Если тип запроса SERVER\_ACTION и action\_id равно "changemode", то устанавливается тип данных rsp.data в "changemode", а rsp.msg устанавливается случайной фразой из массива changemodePhrases.

Если тип запроса SERVER\_ACTION и action\_id равно "guessedright", то устанавливается тип данных rsp.data в "guessedright", а rsp.msg устанавливается случайной фразой из массива GuessedRightPhrases.

Если тип запроса SERVER\_ACTION и action\_id равно "guessedwrong", то устанавливается тип данных rsp.data в "guessedwrong", а rsp.msg устанавливается случайной фразой из массива GuessedWrongPhrases.

Если тип запроса не является одним из предопределенных типов (не является "MESSAGE\_TO\_SKILL", "SERVER\_ACTION", "RUN\_APP", "CLOSE\_APP"), то устанавливается тип данных rsp.data в "mark", а rsp.msg устанавливается как "Спасибо за оценку".

Если тип запроса равен "MESSAGE\_TO\_SKILL", то выполняется обработка текстов запроса r.nlu.texts и преобразование их в команду с помощью функции textToCommand(). Затем осуществляется проверка типа команды (command.type) и соответствующие действия:  
Если тип команды равен "changeword", то устанавливается случайная фраза из массива changewordPhrases в rsp.msg, а тип данных rsp.data устанавливается равным команде.

Если тип команды равен "guessedright", то устанавливается случайная фраза из массива GuessedRightPhrases в rsp.msg, а тип данных rsp.data устанавливается равным команде.

Если тип команды равен "guessedwrong", то устанавливается случайная фраза из массива GuessedWrongPhrases в rsp.msg, а тип данных rsp.data устанавливается равным команде.

Если тип команды равен "close", то устанавливается тип данных rsp.data равным команде, а rsp.msg устанавливается как "Закрываю".

Если тип команды равен "help", то устанавливается тип данных rsp.data равным команде.

Если тип команды равен "restart", то rsp.msg устанавливается как "Начинаю заново", а тип данных rsp.data устанавливается равным команде.

Если тип команды равен "greet", то rsp.msg устанавливается как приветственное сообщение, а тип данных rsp.data устанавливается равным команде. При этом выполняется проверка на count == 0, и если это условие выполняется, то rsp.msg устанавливается как дополнительное приветственное сообщение, и count увеличивается на 1.

Если тип команды равен "value", то rsp.msg устанавливается как "Оценивание", тип данных rsp.data устанавливается равным команде, а свойство rsp.body.messageName устанавливается равным "CALL\_RATING".

Если тип команды равен "double", то rsp.msg устанавливается пустой строкой, а тип данных rsp.data устанавливается пустым объектом.

Если тип команды равен "fail", то выполняется проверка значения gender и appeal и выбирается соответствующая фраза из соответствующего массива (maleofficialFailPhrases, maleno\_officialFailPhrases, femaleofficialFailPhrases, femaleno\_officialFailPhrases). Выбранная фраза устанавливается в rsp.msg, а тип данных rsp.data устанавливается равным команде.

Выводится в консоль значение команды (console.log(command)).

В конце каждой итерации цикла yield rsp возвращает объект ответа rsp.

В общем в цикле обрабатываются различные типы запросов, генерирутся соответствующие ответы, устанавливаются типы данных и сообщения в объекте ответа rsp и возвращает его для дальнейшей обработки.

## **Описание взаимодействия компонентов**

Общая архитектура приложения представлена на рисунке 10.

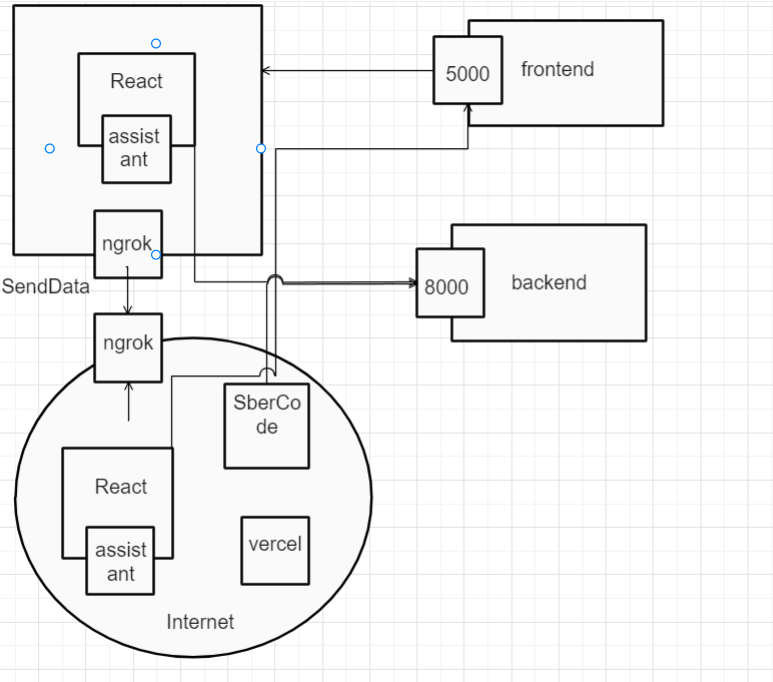


Рисунок 6. Архитектура приложения

Работа приложения начинается с загрузки необходимых частей для функционирования приложения с фронтенда.

Эндпоинты выглядят следующим образом:

* handleClick, которая вызывается по нажатию на кнопку.
* В ней задействуется объект клиента, который мы ранее инициализировали, вызвав createSmartappDebugger. Этот клиент может отправлять события в наш сценарий, каждый раз при вызове метода sendData (для полноценного взаимодействия нужна двyxcтopoнняя связь, поэтому используется объект Response);
* Теперь мы хотим его прочитать объект state и вывести в интерфейсе. Благодаря подписке на событие “data”, переменная state на фронтенде будет обновляться каждый раз, когда с бекенда приходят данные.

## **Использованные инструменты и технологии**

Основные инструменты и технологии, использованные в нашем проекте:

* Svelte – TypeScript-библиотека для создание сайтов.
* Salute.js – библиотека для создания навыков семейства Виртуальных Ассистентов "Салют" на языке JavaScript
* Dialute - универсальный, декларативный фреймворк, помогающий создавать ChatApp и CanvasApp для голосовых ассистентов
* GitHub – система контроля версий. Была использована для хранения иерархии нашего проекта;
* Vercell – хостинг для сайтов, связанный напрямую с репозиторием GitHub. Был использован для хостинга веб-интерфейса нашего приложения и для хостинга бекенда;
* JavaScript – мультипарадигменный язык программирования.
* HTML - язык гипертекстовой разметки
* CSS - язык декорирования и описания внешнего вида

Ссылка на Github проекта: <https://github.com/Dashibug/1337>

## **ЗАКЛЮЧЕНИЕ**

В ходе выполнения данной курсовой работы мы приобрели навыки командной работы и познакомились с производственным процессом работы над приложением. Мы познакомились с множеством новых для нас фреймворков и инструментов для разработки, углубили свои знания языков программирования (в особенности, языка JavaScrip), прочитали много документации, в том числе и на английском языке. В общем и целом, для нашей команды это был очень важный этап становления как разработчиков, мы были очень рады воспользоваться такой возможностью.

В результате работы получилось приложение «Английский крокодил» для SmartMarket. Данная платформа пока набирает популярность, но мы уверены, что скоро она станет довольно распространенной на территории нашей страны. Мы считаем, что наш смартап станет довольно популярным, так как это приложение подходит людям любых возрастов и способно собрать вместе всю семью у экрана SberBox.

Возможно и дальнейшее развитие данного смартапа. Планируется добавление новых функций, таких как сбрасывание счёта и выбор уровня сложности.

## **ИСТОЧНИКИ**

1. SmartMarket Developers. // Документация для тех, кто создаёт смартапы. – URL: <https://developer.sberdevices.ru/> – Текст: электронный.
2. Svelte. // Документация. – URL: <https://svelte.dev/docs> – Текст: электронный.
3. Dialute. // Документация. – URL: <https://dialute.vercel.app/ru/1.introduction> – Текст: электронный.
4. GitHub: сайт. // Пример: Todo смартап, который демонстрирует взаимодействие с Assistant Client. – URL: <https://github.com/sberdevices/assistant-client> – Текст: электронный.
5. VC.ru: сайт. // SmartMarket Сбера: как работает наш маркетплейс приложений и почему мы уверены, что он понравится разработчикам. – URL: <https://vc.ru/dev/195861-smartmarket-sbera-kak-rabotaet-nash-marketpleys-prilozheniy-i-pochemu-my-uvereny-chto-on-ponravitsya-razrabotchikam> – Текст: электронный.
6. Habr.com: сайт. // Хитрости разработки смартапов для SmartMarket. – URL: <https://habr.com/ru/articles/599493/> – Текст: электронный
7. Habr.com: сайт. // Пишем смартапп для ассистента Сбера. – URL: <https://habr.com/ru/articles/650281/> – Текст: электронный