Министерство образования РМ

**ГБПОУ РМ «Саранский государственный промышленно-экономический колледж»**

СОГЛАСОВАНО

Зав. отделением

По специальности

Л.А.Терентьева

« » 2020 года

**КУРСОВАЯ РАБОТА**

по дисциплине МДК.02.02 Инструментальные средства разработки программного обеспечения

Тема Разработка обучающего приложения в области сценарного искусства на базе веб-фреймворка Vue.js

Обозначение курсовой работы

СПЭК.10.02.03.20.30 КР

Автор курсовой работы В.А.Смолькин

(подпись) (дата)

Руководитель курсовой работы Н.И.Кручинкина

(подпись) (дата)

Дата защиты

Оценка

Саранск

2020

Министерство образования РМ

**ГБПОУ РМ «Саранский государственный промышленно-экономический колледж»**

Дата выдачи задания

« » 2020 года

Дата сдачи курсовой работы

« » 2020 года

Специальность 09.02.07

**ЗАДАНИЕ**

**НА КУРСОВУЮ РАБОТУ**

по дисциплине МДК.02.02 Инструментальные средства разработки программного обеспечения

Тема Разработка обучающего приложения в области сценарного искусства на базе веб-фреймворка Vue.js

Утверждена приказом по колледжу № 20 от 13.02.2020

Автор курсовой работы В.А.Смолькин

Студент группы ИСИП4А17

Содержание курсовой работы:

Введение

1 Описание предметной области

2 Описание инструментов разработки и обоснование их выбора

3 Программная реализация веб-приложения

Заключение

Приложение

Заведующий отделением Л.А.Терентьева

(подпись) (дата)

Руководитель курсовой работы Н.И.Кручинкина

(подпись) (дата)

Задание принял к исполнению В.А.Смолькин

(подпись) (дата)

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc53318133)

[1. Описание предметной области 5](#_Toc53318134)

[2. Описание инструментов разработки и обоснование их выбора 7](#_Toc53318135)

[2.1 Серверная часть (back-end) 7](#_Toc53318136)

[2.2 Клиентская часть (front-end) 9](#_Toc53318137)

[3. Программная реализация веб-приложения 12](#_Toc53318138)

[3.1 Настройка окружения 12](#_Toc53318139)

[3.2 Создание приложения 15](#_Toc53318140)

[3.3 Хостинг приложения 15](#_Toc53318141)

# **ВВЕДЕНИЕ**

Значение интернет-сектора в современном мире трудно переоценить. В том числе веб технологии активно используются в сфере образования, от вебинаров и онлайн-курсов с элементами интерактивности до внедрения методик дистанционного обучения. А события последнего полугодия доказали не только полную конкурентоспособность данного способа обучения, с чем уже было трудно спорить и ранее, но и незаменимость его в условиях самоизоляции.

Параллельно этому имеется устойчивая тенденция к переносу не требовательных к ресурсам десктопных приложений в интернет среду, благодаря чему делается возможной работа одного и того же приложения в устройствах с полностью иной архитектурой, упрощается работа над одними данными с разных машин.

Исходя из вышеприведенных тезисов, актуальность данной курсовой работы заключается в создании программного решения для обучения столь требовательному к творческому аспекту предмета, как написание сценария, с использованием современных веб-технологий.

Целью работы является, соответственно, разработка веб-приложения для обучения написанию сценария.

Для дальнейшей реализации цель была разбита на ряд задач:

1. Изучение процесса обучения сценарному мастерству;
2. Изучение актуальных веб-технологий и выбор из них наиболее подходящих для реализации цели;
3. Разработка непосредственно веб-приложения.

# **1. Описание предметной области**

Сценарий — литературно-драматическое произведение, написанное как основа для постановки кино- или телефильма, и других мероприятий в театре и иных местах.

Сценарий в кинематографе, как правило, напоминает пьесу и подробно описывает каждую сцену и диалоги персонажей с ремарками. Иногда сценарий представляет собой адаптацию отдельного литературного произведения для кинематографа, иногда в этом случае автор романа бывает и автором сценария (сценаристом).

Сценарист — человек, который пишет сценарий к фильму. Иногда в написании одного и того же сценария принимает участие несколько сценаристов, прежде чем режиссёр выберет лучший вариант. Необязательно автор книги пишет сценарий при её экранизации. Эта работа обычно отдаётся сценаристу, а автор произведения, при возможности является соавтором сценария или консультантом.

Если мы говорим о стандартах, то одна страница правильно написанного и отформатированного сценария равна приблизительно одной минуте кинофильма, из чего вытекает распространенное, но не абсолютное, ограничение на длину сценария равное 128 страницам, что приблизительно равно двум часам и восьми минутам хронометража или средней продолжительности большинства современных кинофильмов.

Над литературным сценарием работает кинодраматург, часто в этой работе участвуют продюсер и режиссёр, которые нередко становятся его соавторами.

Чтобы литературный сценарий мог быть использован, его адаптируют к условиям кино, трансформируя в киносценарий, где описательная часть сокращается, чётко прописываются диалоги, определяется соотношение изобразительного и звукового ряда. Здесь драматургическая сторона разрабатывается по сценам и эпизодам, а постановочная разработка действия ведётся по объектам съёмки. Каждая новая сцена записывается на отдельную страницу, что впоследствии облегчит работу в установлении их последовательности в развитии сюжета. Кроме того, киносценарий проходит производственное редактирование. Это необходимо для определения длины фильма, количества объектов съёмки, декораций, количества актёров, организации экспедиций и многого другого. Без этого невозможно рассчитать финансовые затраты на кинопроизводство.

Сценарии следуют основной линейной структуре, создающей форму киносценария, в пределах которой они могут варьироваться, но основа ее в том или ином виде остается стабильной и выглядит следующим образом.

1. Начало. Первый акт. Завязка. Приблизительно 1-30 минуты хронометража;
2. Первый сюжетный поворот;
3. Середина. Второй акт. Конфликт. Приблизительно 30-90 минуты хронометража;
4. Второй сюжетный поворот;
5. Конец. Третий акт. Развязка. Приблизительно 90-120 минуты хронометража.

Означенная структура — это так же то, что можно назвать парадигмой киносценария.

# **2. Описание инструментов разработки и обоснование их выбора**

## **2.1 Серверная часть (back-end)**

Node.Js и Firebase – два инструмента, служащих основой разрабатываемого приложения.

Обратимся к определению первого из них.

Node.js – открытая, кросс-платформенная, серверная среда выполнения JavaScript, исполняющая JavaScript код вне веб-браузера. Node.js позволяет разработчикам использовать JavaScript для написание инструментов командной строки и для запуска скриптов на стороне сервера, чтобы создавать контент динамических веб-страниц до того, как страница была отправлена на браузер пользователя. Как следствие, Node.js воплощает парадигму «JavaScript везде», объединяя всю разработку веб-приложения вокруг единственного языка программирования, нежели предполагая использование разных языков программирования для серверной и клиентской частей.

Именно последнее предложение является основным аргументом в пользу выбора Node.js. Приложения, созданные на его основе являются более монолитными в своей сути, упрощается процесс совместимости клиентской и серверной частей, так как они написаны на одном языке, а значит оперируют одинаковыми объектами и форматами данных.

Следующим аргументов является встроенный менеджер пакетов, позволяющий легко устанавливать дополнительные расширения, будь то библиотеки или фреймворки, что существенно упрощает процесс разработки, позволяя тратить меньше времени на настройку окружения и больше – непосредственно на создание приложения.

Третьим важным для разработчика фактором является то, что Node.js распространяется по лицензии MIT, что делает его полностью свободным, открытым и бесплатным программным обеспечением (рис. 1), исключая тем самым любые возможные юридические проблемы.

И пусть все вышеперечисленное является в большей степени преимуществами для разработчика, существует как минимум один плюс, который Node.js предоставляет для конечного пользователя. Так как в основе Node.js лежит JavaScript движок V8, разработанный компанией Google, то конечные приложения, как правило, обладают очень высокой производительностью, выражающейся в быстром отклике, меньшем времени исполнения скриптов, быстрой загрузке контента на странице.

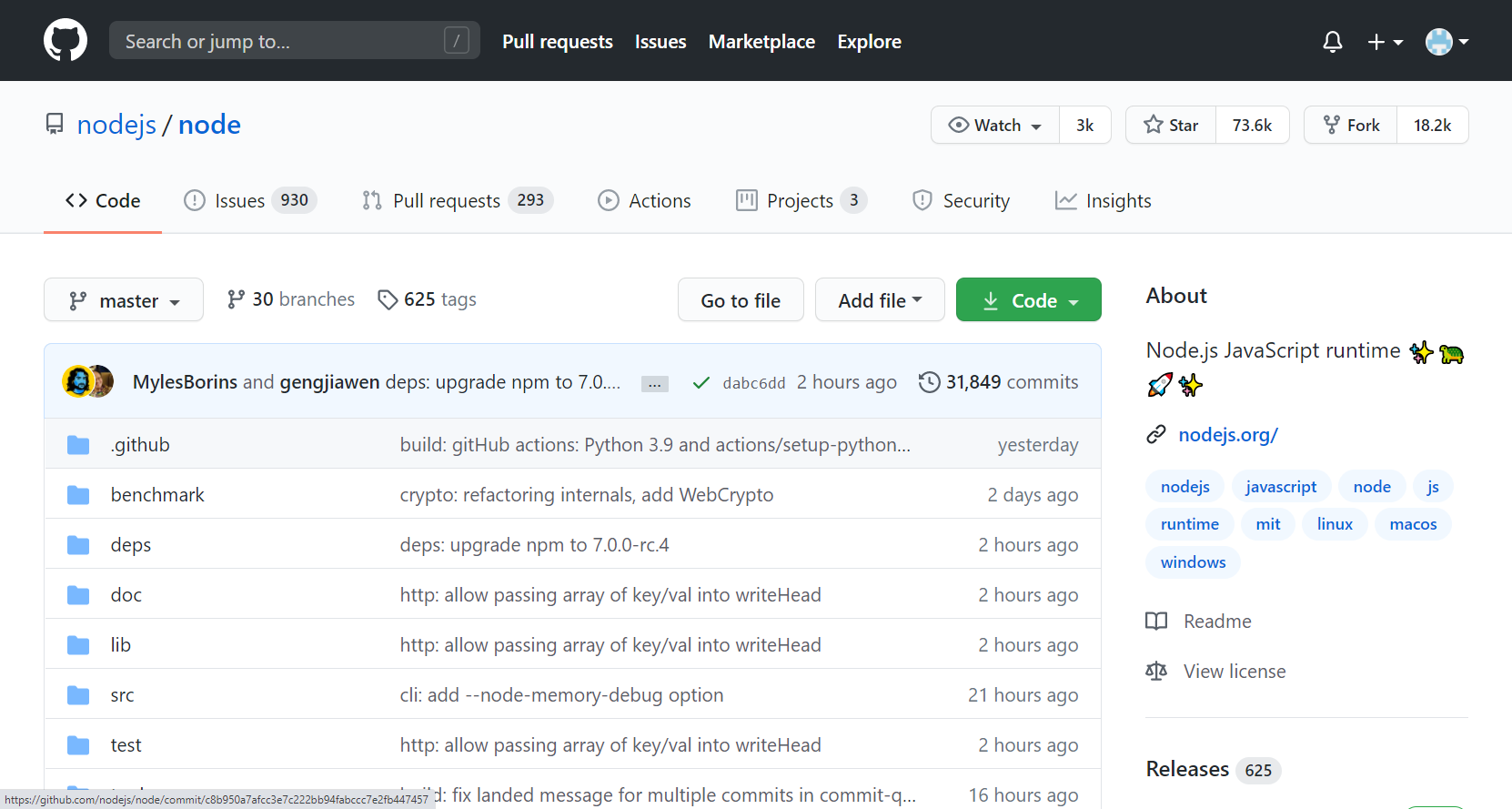


Рисунок 1 – репозиторий Node.js, хранящий весь его исходный код в открытом доступе

Вторым избранным инструментом является, как уже было сказано, Firebase представляющий собой платформу, разработанную Google для создания мобильных и веб-приложений. Firebase уже реализует ряд важнейших сервисов, таких как NoSql база данных, механизмы регистрации и авторизации, хостинг, а также удобную панель администрирования (рис. 2).

В то время, как это не оказывает большого влияние на конечного пользователя, данный набор готовых к работе сервисов, которые остается только связать с конечным приложением, опять же серьезно упрощает процесс разработки, позволяя сосредоточиться на уникальных особенностях приложения вместо очередной реализации общих для подавляющего большинства веб-приложений механизмов.

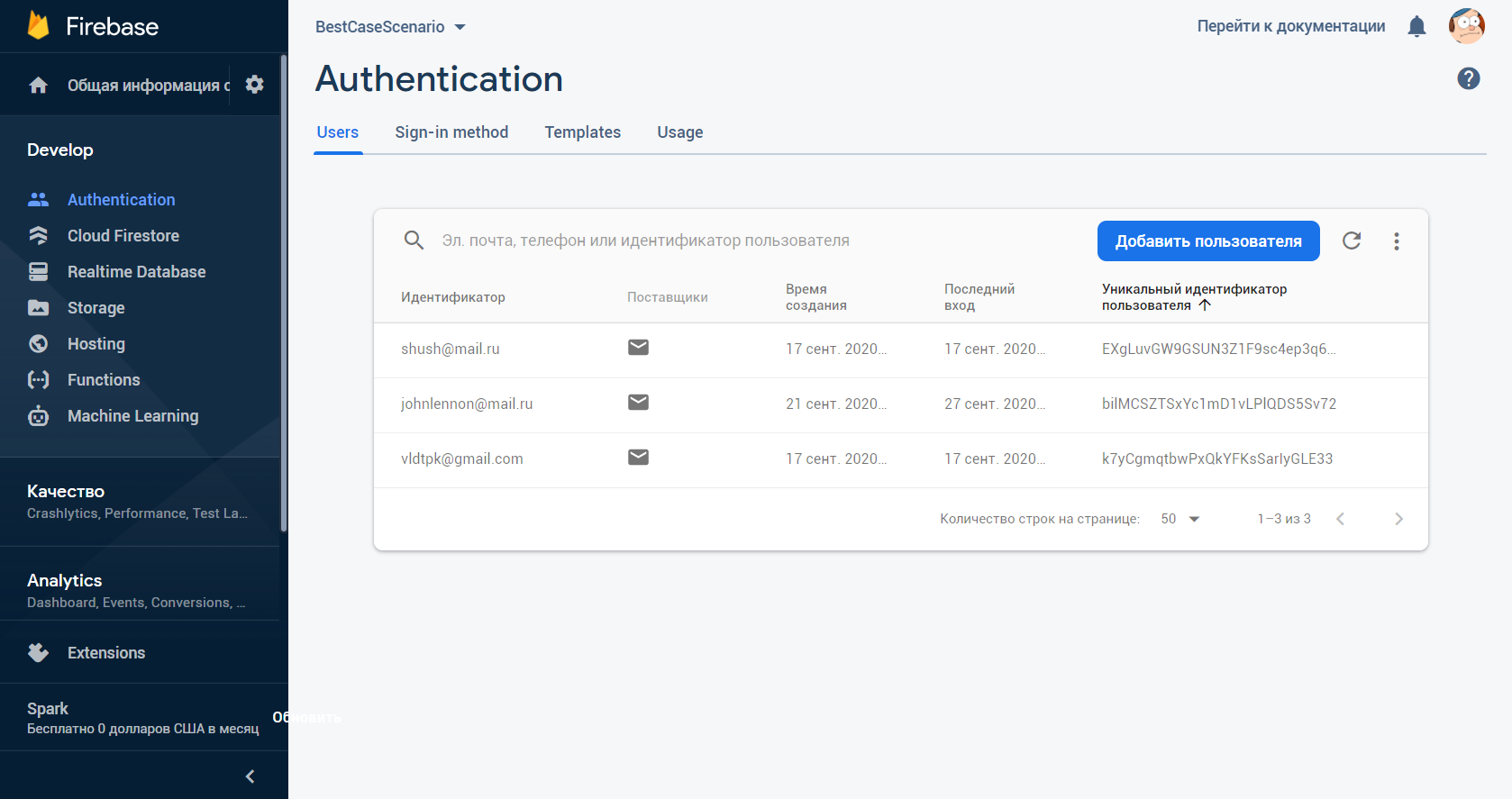


Рисунок 2 – консоль администрирования Firebase

## **2.2 Клиентская часть (front-end)**

Краеугольным камнем разработки клиентской части данного приложения является Vue.js.

Обратимся к его определению.

Vue – это прогрессивный фреймворк для создания пользовательских интерфейсов. В отличие от фреймворков-монолитов Vue создан пригодным для постепенного внедрения. Его ядро в первую очередь решает задачи уровня представления (view), что упрощает интеграцию с другими библиотеками и существующими проектами. С другой стороны, Vue полностью подходит и для создания сложных одностраничных приложений (SPA, Single-Page Applications), если использовать его совместно с современными инструментами и дополнительными библиотеками.

В то время, как возможности постепенного внедрения интересуют нас в наименьшей степени, так как данное веб-приложения создается с нуля и сразу с использованием означенного фреймворка, другие особенности Vue имеют гораздо большее значение.

Первое и основное – Vue, как уже было постановлено в его определении, может использоваться для создания одностраничных приложений, что создает полностью иной опыт разработки и представления интернет-контента.

Для разработчика это, в сущности, означает, что он больше манипулирует не разными страницами, а отдельными компонентами (рис. 3) – будь то меню, форма регистрации, окно профиля или иной контент – добавляя, изменяя или удаляя их, что обеспечивает дополнительный уровень гибкости. По сути, мы уже не столько перескакиваем со страницы на страницу, сколько управляем тем, что отображается на одной единственной странице.

И это в свою, очередь, меняет восприятие результата для конечного пользователя, поскольку тот ощущается не как стопка связанных между собой веб-страниц, а как полноценное единое приложение сродни тому, что существуют на настольных ПК.



Рисунок 3 – пример компонентной структуры Vue. Вверху находится корневой компонент, олицетворяющий приложение целиком

Существует так же ряд иных особенностей, выделяющих Vue на фоне его прямых конкурентов – React и Angular. В то время, как их нельзя назвать абсолютными преимуществами, так как при определенных ситуациях они могут превратиться в помеху для разработчика, в случае данного приложения они несут скорее положительный окрас.

Single File Component структура – для каждого компонента создается отдельный файл, в котором хранится его шаблон (грубо говоря, то как будет сверстан компонент), его таблица стилей и его программный код.

Чистый JavaScript – код во Vue компонентах пишется на обычном JS без использования TypeScript, JSX или иных его расширений. В то время как это скорее минус в действительно крупных приложениях, отсутствие необходимости изучать особенности этих расширений вкупе с тем, что последние редакции JavaScript и сами по себе обладают достаточно широким функционалом, превращают это в плюс и дополнительно уменьшают порог вхождения. Тем более, что во Vue присутствует возможность работы с этими расширения при действительной необходимости.

Vuex – паттерн управления состоянием плюс библиотека для приложений на Vue.js. Он служит централизованным хранилищем данных для всех компонентов приложения с правилами, гарантирующими, что состояние может быть изменено только предсказуемым образом. Пожалуй, то что можно занести как однозначный плюс Vue, так как будучи разработанным той же командой, Vuex и Vue изначально совместимы и легко интегрируются.

# **3. Программная реализация веб-приложения**

## **3.1 Настройка окружения**

Так как разработка любого приложения, и веб-приложения, предполагает использование целой группы инструментов, определенная часть процесса разработки должна быть посвящена настройке рабочего окружения.

Настройка окружения данного приложения начинается с регистрации проекта и приложения в Firebase (Рис. 4).

Проходя достаточно интуитивный процесс, требующий от нас указать лишь название проекта, создаем проект в Firebase.

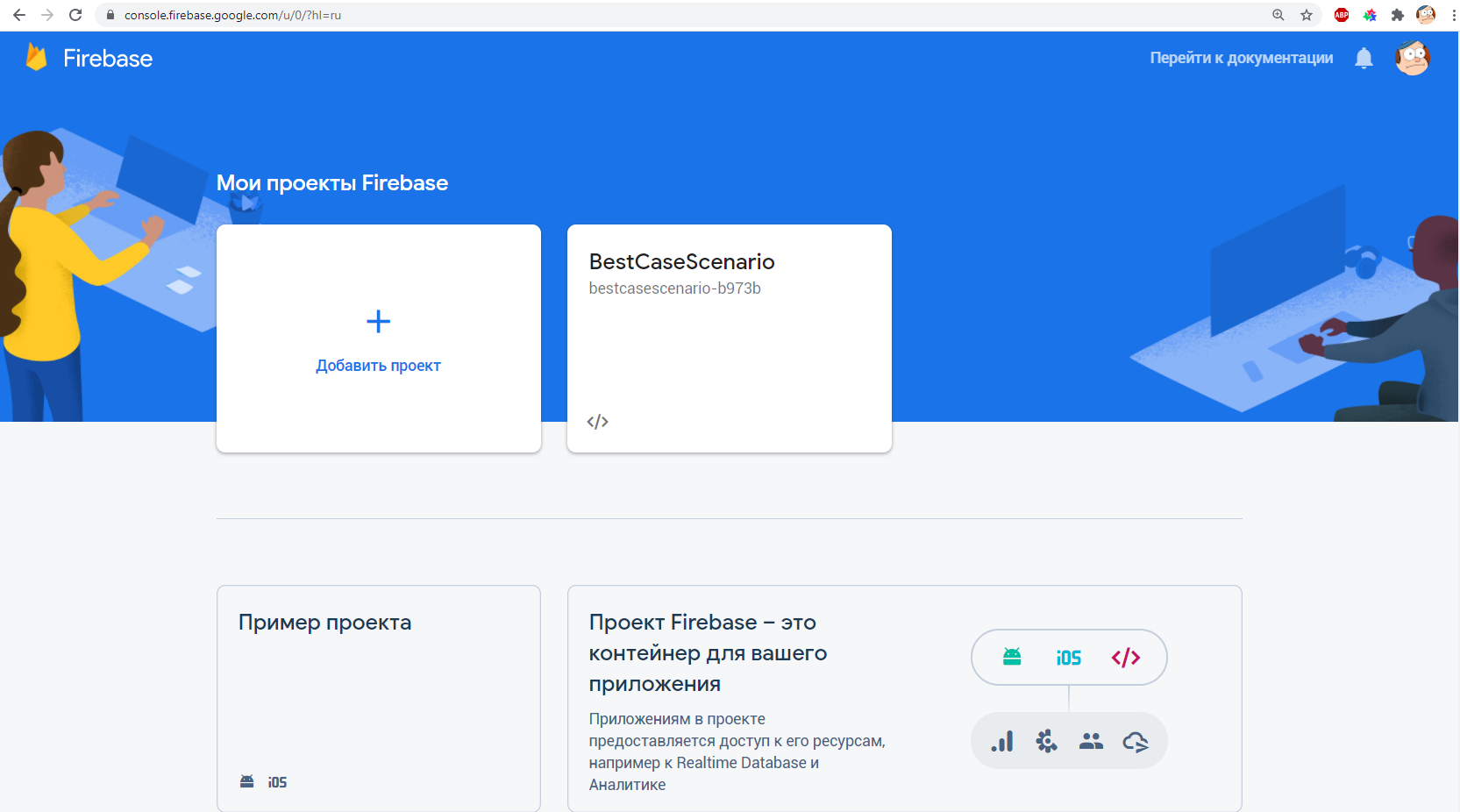


Рисунок 4 – главная страница Firebase

Попав на консоль, в разделе Authentication, Sign-In methods включаем авторизацию по адресу электронной почты и паролю. В дальнейших итерациях приложения имеет смысл реализовать авторизацию так же и через популярные сервисы, но в текущей вариации достаточно базиса.

Следующим шагом зарегистрируем веб-приложение, выбрав в консоли проекта соответствующую опцию. Здесь снова достаточно указать только имя приложения после чего, подтвердив его, нам будет продемонстрирован JSON с данными конфигурации (рис. 5). Его содержимое нам понадобиться на дальнейший этапах разработки, но так как доступ к нему можно получить в любой момент, то в данный момент он не является нашей заботой.

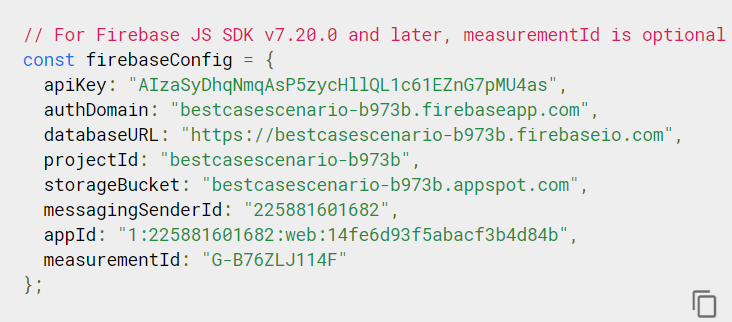


Рисунок 5 – JSON конфигурации приложения в Firebase

Следующим действием станет создание Github-репозитория (рис. 6). Так как данный проект предполагает наличие множества файлов и большого числа зависимостей, вопросы контроля версий, отслеживания изменения и безопасного хранения становятся жизненно важны. А значит создание репозитория, как раз исполняющего эти функции, еще до начала написания любого программного кода проекта, является обыденной необходимостью для любого современного разработчика.

После этого, а также установки системы контроля версий Git на рабочую машину, консольной командой git clone со ссылкой на созданный репозиторий, копируем его на ПК.

Дальнейшим шагом устанавливаем актуальную версию Node.js c официального сайта разработчика (nodejs.org). На этом шаге не имеет смысла останавливаться, так как процесс абсолютно линейный и ничем не отличается от установки любого стандартного ПО.

На данной стадии для завершения настройки окружения достаточно нескольких консольных команд.

npm install vue – установка Vue.js.

npm install -g @vue/cli – установка консольный инструментов Vue.

После чего необходимо перейти в папку хранящую копию репозитория и оттуда ввести команду vue create, чтобы создать каркас проекта.

Завершается настройка окружения установкой SDK для Firebase, что может быть сделано консольной командой npm install --save firebase.

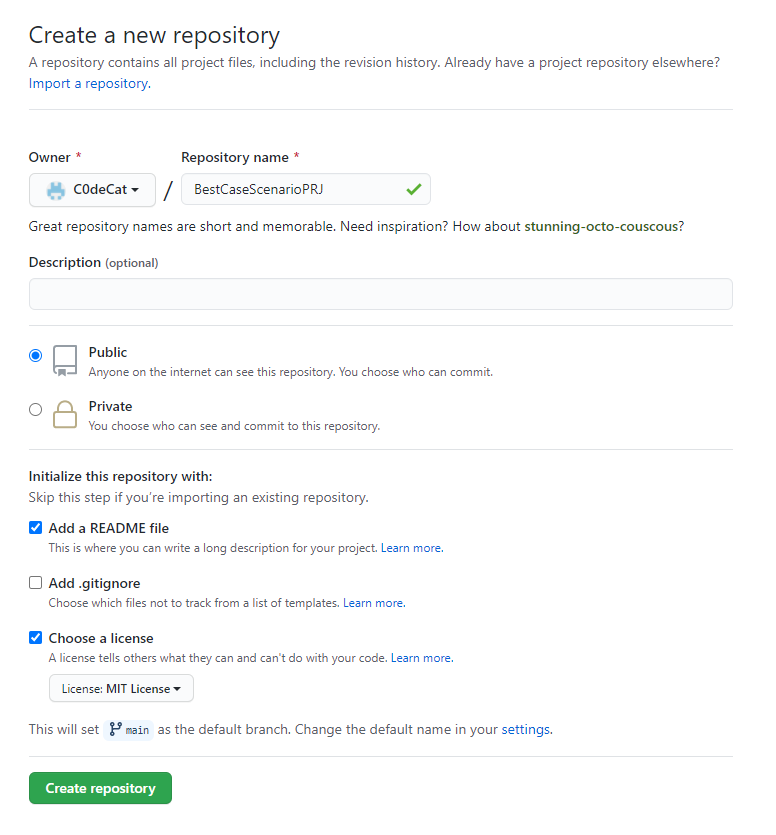


Рисунок 6 – окно создания нового репозитория.

## **3.2 Создание приложения**

## **3.3 Хостинг приложения**