**Оглавление**

[Введение 3](#_Toc47046628)

[1 Постановка цели и формулировка задач 4](#_Toc47046629)

[2 Анализ и сравнительный обзор аналогов 5](#_Toc47046630)

[3 Выбор инструментария и технологии проектирования 7](#_Toc47046631)

[4 Проектирование программного средства 9](#_Toc47046632)

[4.1 Обобщенная архитектура проекта 9](#_Toc47046633)

[4.2 Проектирование базы данных 10](#_Toc47046634)

[5 Программная реализация проекта 11](#_Toc47046635)

[5.1 Физическая структура программного средства 11](#_Toc47046636)

[5.2 Аутентификация пользователей 12](#_Toc47046637)

[Заключение 13](#_Toc47046638)

[Список использованных источников 14](#_Toc47046639)

[Приложение А 15](#_Toc47046640)

[Приложение Б 16](#_Toc47046641)

[Приложение В 20](#_Toc47046642)

# Введение

Технологическая практика студентов является необходимым и важнейшим этапом в системе подготовки высококвалифицированных специалистов. Она является составной частью учебного процесса и направлена на закрепление и углубление теоретических знаний, приобретение студентами навыков практической работы путем самостоятельного выполнения заданий.

Местом прохождения технологической практики является УО «Республиканский институт профессионального образования». Сроком прохождения технологической практики является период с 6 июля 2020 года по 31 июля 2020 года.

УО «Республиканский институт профессионального образования» осуществляет свою деятельность с 1993 года и является единственным на евразийском экономическом пространстве многофункциональным учреждением образования, которое осуществляет свою деятельность по следующим направлениям:

* научно-, учебно-, организационно-методическим обеспечением профессионально-технического, среднего специального образования (в т.ч. лиц с ОПФР), непрерывного профессионального обучения рабочих и служащих;
* повышение квалификации и переподготовкой руководящих и педагогических кадров профессионального образования;
* подготовкой научных кадров высшей квалификации для системы профессионального образования;
* подготовкой рабочих и специалистов с профессионально-техническим и средним специальным образованием;
* научно-исследовательская работа.

В структуру данного предприятия входят:

* администрация;
* центр научно-методического обеспечения профессионального образования;
* центр развития профессионального образования;
* факультет повышения квалификации и переподготовки кадров;
* центр научно-методического обеспечения воспитательной работы в учреждениях ПТО и ССО;
* аспирантура;
* центр международного сотрудничества;
* информационно-аналитический центр профессионального образования;
* центр учебной книги и средств обучения;
* центр мониторинга качества профессионального образования.

# 1 Постановка цели и формулировка задач

В качестве индивидуального задания необходимо было разработать веб-приложение «Психологические тесты». Суть задания состояла в том, чтобы автоматизировать работу психолога и создать приложение, для прохождения психологических тестов, которые ранее были только в печатном варианте.

Функционально приложение должно выполнять следующие задачи:

* регистрация новых пользователей;
* аутентификация пользователей;
* аутентификация администратора (психолога);
* просмотр пользователем доступных для прохождения тестов;
* прохождение пользователем выбранного теста;
* просмотр пользователем результатов пройденных тестов;
* просмотр администратором результатов пройденных тестов всех зарегистрированных пользователей.

Таким образом, при помощи данного программного средства, психолог может проанализировать результаты тестируемых, при этом не обрабатывая каждый тест вручную.

Для того, чтобы наглядно показать, как именно пользователи могут взаимодействовать с приложением, на рисунке 1.1 представлена UML диаграмма использования. Диаграмма UML – это графическое представление набора элементов, изображаемое в виде связанного графа с вершинами (сущностями) и ребрами (отношениями).

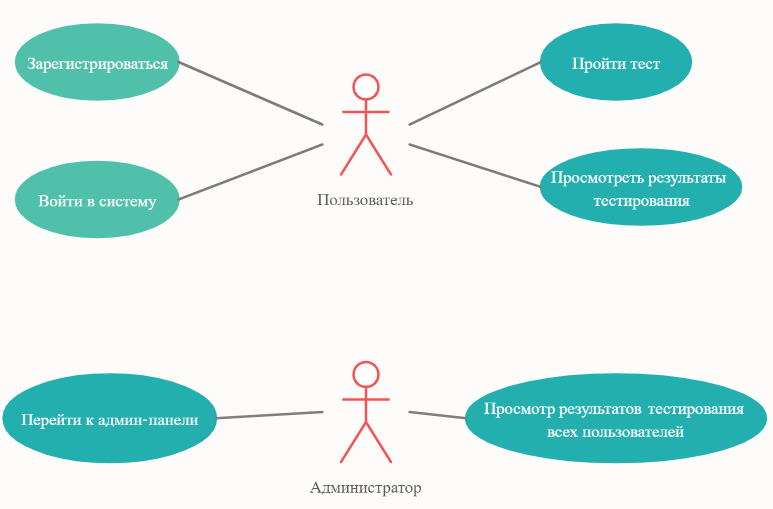


Рисунок 1.1 – UML диаграмма использования

# 2 Анализ и сравнительный обзор аналогов

Прежде чем приступить к разработке программного средства «Психологические тесты», необходимо ознакомиться и проанализировать существующие решения в этой области. Это позволит понять, каким функционалом должно обладать приложение такого типа.

Рассмотрим похожий по тематике веб-сайт Testometrika.com. Данный сервис позволяет:

* регистрировать и аутентифицировать пользователей;
* просматривать список тестов;
* проходить тесты;
* осуществлять поиск тестов;
* фильтровать тесты;
* получать результат прохождения теста.

На рисунке 2.1 представлен интерфейс данного веб-сайта.

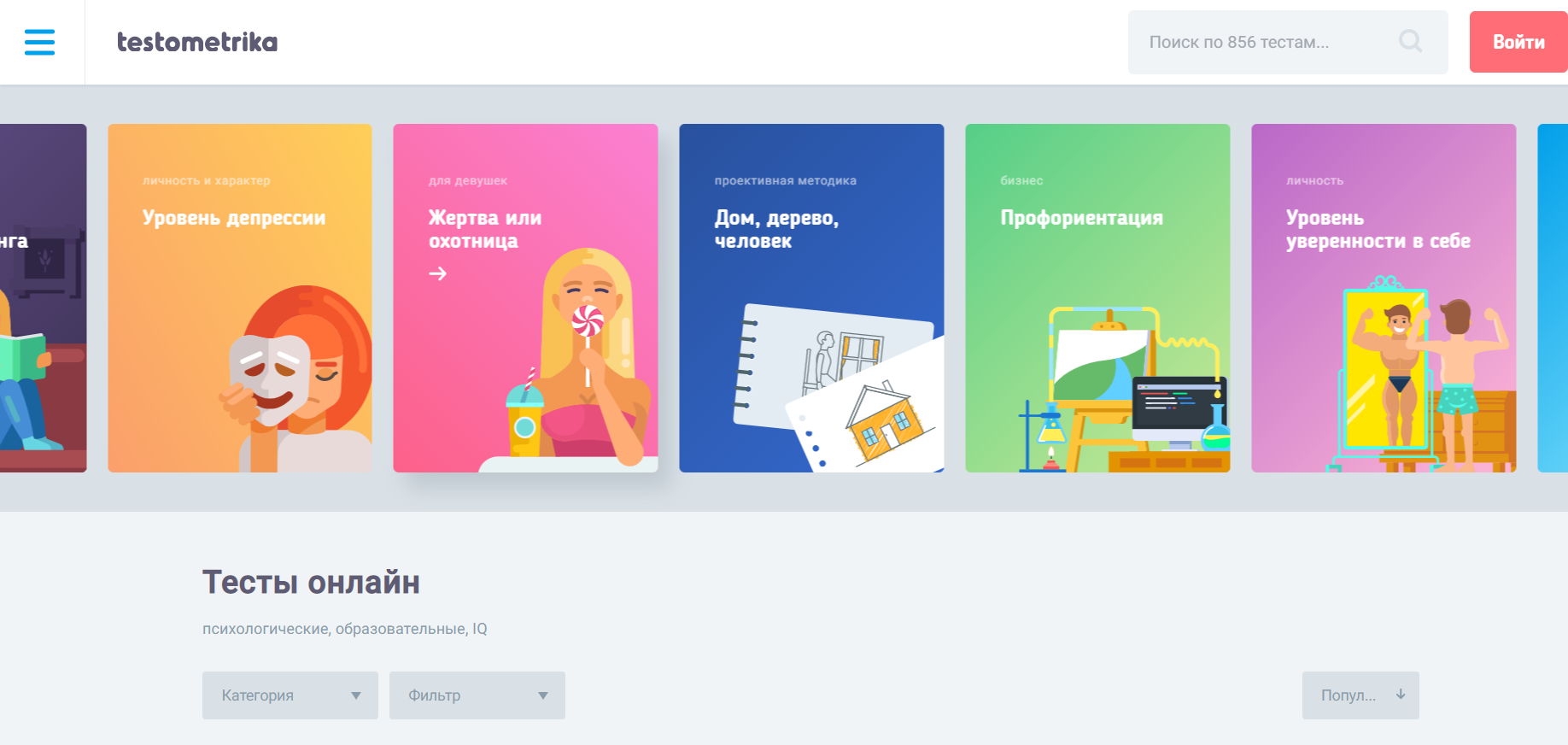


Рисунок 2.1 – Интерфейс веб-сайта Testometrika.com

Из рисунка 2.1 видно, что интерфейс данного сервиса прост и интуитивно понятен пользователю, что является большим преимуществом, для такого рода приложений.

Рассмотрим в качестве второго примера рассмотрим веб-сайт Aducar.by. Интерфейс этого сайта представлен на рисунке 2.2.

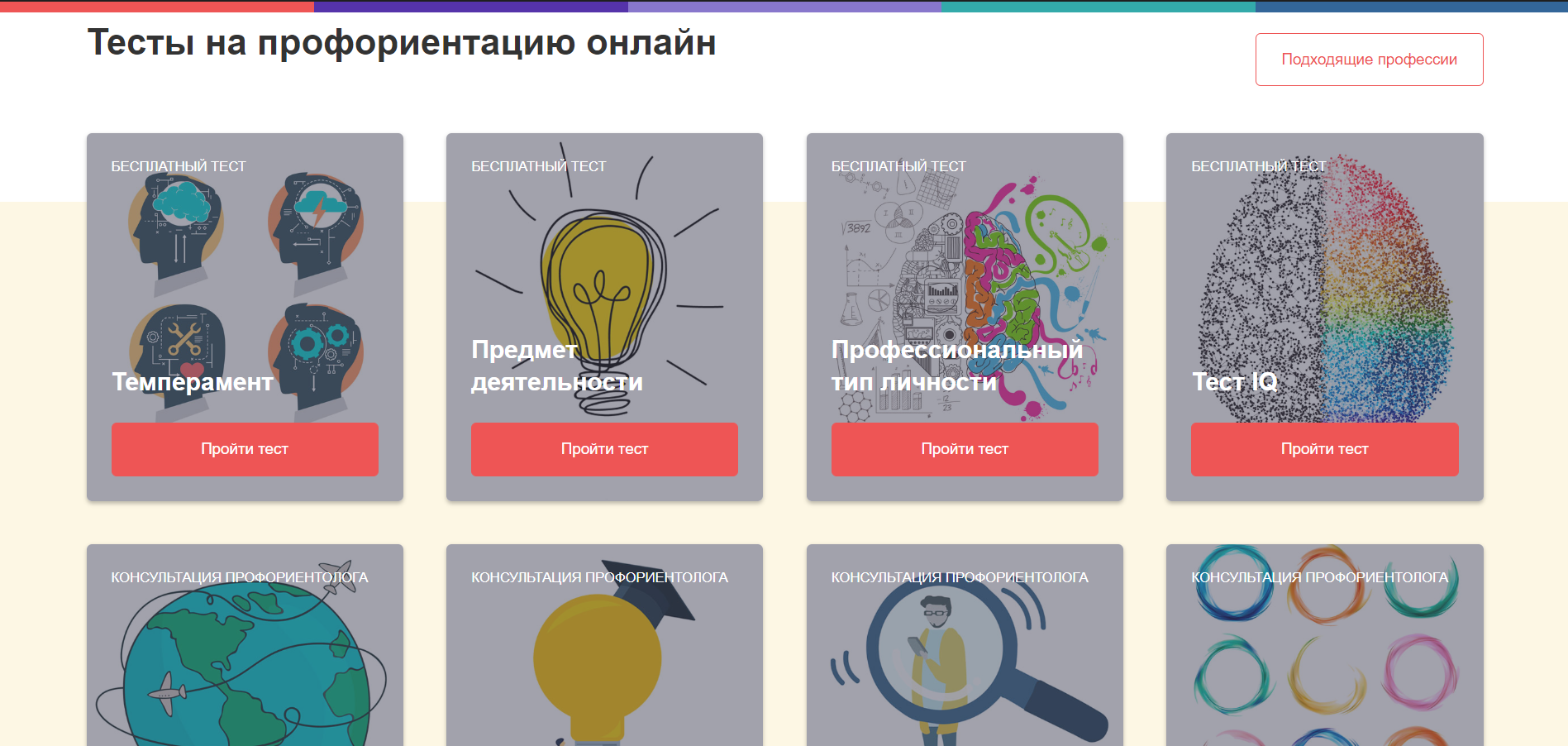


Рисунок 2.1 – Интерфейс веб-сайта Aducar.by

По функционалу данный веб-сайт очень схож с первым рассмотренным сайтом, однако здесь нету поиска и фильтров для тестов, а также, важной особенностью является то, что тесты не может проходить незарегистрированный пользователь, в отличии от первого рассмотренного сайта, где пользователь может проходить тесты будучи как зарегистрированным так и нет.

Исходя из рассмотренных выше примеров, можно сделать вывод о том каким должно быть приложение такого типа. На таких сайтах важно наличие регистрации, так как пользователь может захотеть просмотреть результаты тестов, пройденных ранее, а для этого пользователь должен быть зарегистрированным. Также важным аспектом является наличие доступного и интуитивно понятного интерфейса. Таким образом можно выделить основные функции, которыми должно обладать приложение такого рода:

* регистрация новых пользователей;
* аутентификация пользователей;
* прохождение пользователем тестов;
* просмотр пользователем результатов пройденных тестов.

Так же, так как в первую очередь это приложение должно помогать психологу анализировать результаты, необходим администратор (психолог), который сможет осуществлять следующие действия:

* входить в приложение как администратор;
* просматривать результаты тестирования всех зарегистрированных пользователей.

1. **Выбор инструментария и технологии проектирования**

Разработка данного приложения осуществлялась на языке JavaScript. При реализации курсового проекта использовались технологии: Node.js, фреймворк Express, ORM-библиотека Sequelize[4] – для реализации back-end, для front-end части используется фреймворк Vue.js[1], Vuex[8], Vuetify[7], Axios. MS SQL Server в качестве СУБД.

JavaScript – мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. Является реализацией стандарта ECMAScript (стандарт ECMA-262). JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

Node.js – программная платформа, основанная на движке V8 (транслирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API (написанный на C++), подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода. Node.js применяется преимущественно на сервере, выполняя роль веб-сервера, но есть возможность разрабатывать на Node.js и десктопные оконные приложения и даже программировать микроконтроллеры. В основе Node.js лежит событийно-ориентированное и асинхронное (или реактивное) программирование с неблокирующим вводом/выводом.

Express – фреймворк web-приложений для Node.js, реализованный как свободное и открытое программное обеспечение под лицензией MIT. Он спроектирован для создания веб-приложений и API. Де-факто является стандартным каркасом для Node.js. Автор фреймворка, TJ Holowaychuk, описывает его как созданный на основе написанного на языке Ruby каркаса Sinatra, подразумевая, что он минималистичен и включает большое число подключаемых плагинов.

Sequelize – это ORM-библиотека для приложений на Node.js, которая осуществляет сопоставление таблиц в базе данных и отношений между ними с классами. При использовании Sequelize мы можем не писать SQL-запросы, а работать с данными как с обычными объектами.

Vue.js — JavaScript-фреймворк с открытым исходным кодом для создания пользовательских интерфейсов. Легко интегрируется в проекты с использованием других JavaScript-библиотек. Может функционировать как веб-фреймворк для разработки одностраничных приложений в реактивном стиле.

Vuex — паттерн управления состоянием + 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. На рисунке 3.1 представлена схема взаимодействия Vue.js и Vuex.

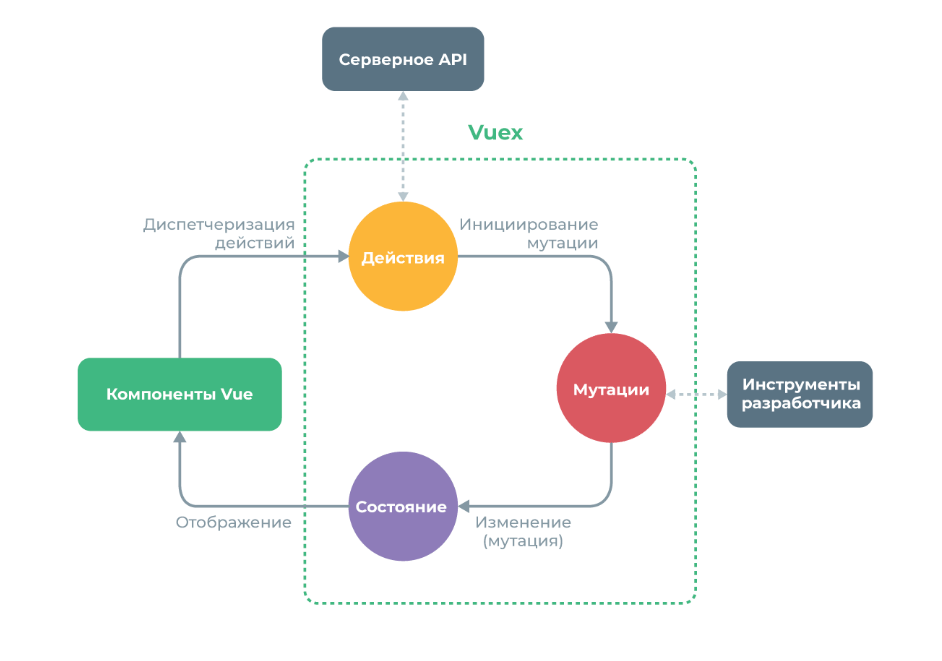


Рисунок 3.1 – Схема взаимодействия Vue.js и Vuex

И последняя технология Axios — это широко известная JavaScript-библиотека. Она представляет собой HTTP-клиент, основанный на промисах и предназначенный для браузеров и для Node.js.

# Проектирование программного средства

Проектирование программного средства – это одна из важнейших задач в процессе работы над приложением, так как в зависимости от неё определяется основные компоненты приложения и зависимости между ними.

Архитектура проекта – это его строение как оно видно (или должно быть видно) из вне его, т.е. представление программного средства как системы, состоящей из некоторой совокупности взаимодействующих подсистем. В качестве таких подсистем выступают обычно отдельные программы. Разработка архитектуры является первым этапом борьбы со сложностью программного средства, на котором реализуется принцип выделения относительно независимых компонент.

Основные задачи разработки архитектуры проекта:

– Выделение программных подсистем и отображение на них внешних функций (заданных по внешнем описании) программного средства;

– Определение способов взаимодействия между выделенными программными подсистемами.

С учетом принимаемых на этом этапе решений производится дальнейшая конкретизация и функциональных спецификаций.

### **4.1 Обобщенная архитектура проекта**

Данное программное средство построена на основе технологий MS SQL Server, Express, Vue.js, Node.js.[2] Исходя из этого архитектуру такого приложения можно представить в виде схемы, представленной на рисунке 4.1.

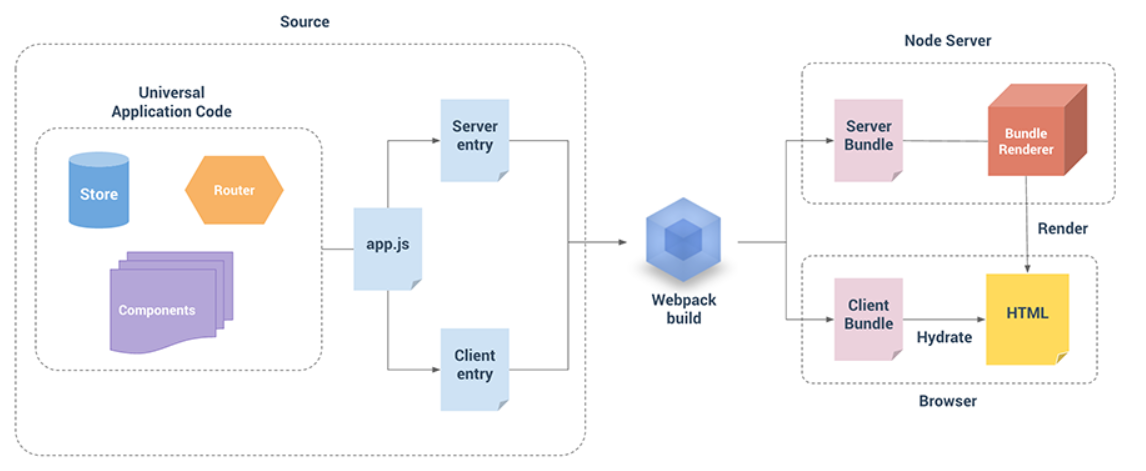


Рисунок 4.1 – Архитектура приложения

Как видно из схемы на рисунке 4.1, приложение чётко разделено на две части: клиентскую и серверную. В свою очередь клиентская часть разделяется на три основных уровня:

* Уровень пользовательского интерфейса — компоненты пользовательского интерфейса приложения;
* Бизнес-уровень — содержит бизнес-логику приложения, которую мы часто называем сервисами;
* Состояние приложения— управление состоянием приложения.

Серверная часть в свою очередь содержит следующие основные компоненты:

* Модели – определяют структуру и логику используемых данных;
* Контроллеры обрабатывают входящие http-запросы, используя для обработки модели и представления, и отправляет в ответ клиенту некоторый результат обработки;

Система маршрутизации – сопоставляет запросы с маршрутами и выбирает для обработки запросов определенный контроллер.

### **4.2 Проектирование базы данных**

Для реализации поставленной задачи была создана база данных. Для её создания использовалась система управления реляционными базами данных Microsoft SQL Server.

Всего в базе данных содержится 7 таблиц:

* в таблице USERS содержится информация о всех зарегистрированных пользователях;
* в таблице TEST\_TYPE – названия всех имеющихся тестов;
* в QUESTIONS – вопросы тестов;
* в BLOCK – наименования блоков, к которым относятся вопросы в тестах;
* в RISK – риски, для каждого из блоков;
* в ANSWERS – варианты ответов на вопросы тестов;
* в RESULTS – результаты прохождения тестов каждого пользователя.

Модель базы данных представлена в приложении А.

# Программная реализация проекта

### **5.1 Физическая структура программного средства**

Как уже было выяснено в предыдущем разделе, веб-приложения имеют четкое разделение на клиентскую и серверную части. На рисунке 5.1 представлена структура проекта.



Рисунок 5.1 – Структура проекта

Как видно из рисунка 5.1, у нас есть две основные папки, это src, в которой находится всё что относиться к клиентской части, и server, в которой соответственно находится всё, что относиться к серверной части. Рассмотрим структуру этих папок подробнее.

Для начала рассмотрим серверную часть. На рисунке 5.2 представлена структура папки server.

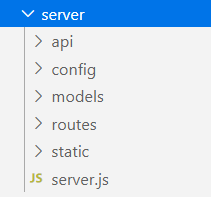


Рисунок 5.2 – Структура папки server

В папке config находится вся конфигурационная информация необходимая для работы сервера и его связи с базой данных.

В папке models располагаются так называемые модели, или по-другому, схемы документов нашей базы данных. Все модели представлены в приложении Б.

В папке routes находятся маршруты для, поступающих на сервер, запросов и так же их обработчики, которые находятся в папке api.

В папке static находятся картинки, которые используются в приложении, т.к. нецелесообразно хранить такие данные в базе данных, то они сохраняются в этой папке, а в базе данных содержаться ссылки на них.

Ну и в файле server находится непосредственно сам сервер.

Теперь рассмотрим клиентскую часть. На рисунке 5.3 представлена структура папки src.

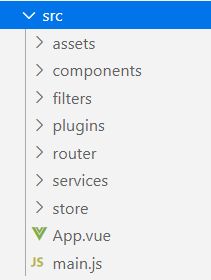


Рисунок 5.3 – Структура папки src

В папке assets располагаются картинки, которые используются для дизайна. Также там находятся некоторые переменные scss, которые используются во многих компонентах.

В папке components находятся все vue компоненты, из которых и состоит представление, которое видит пользователь.

В папке filters находятся некоторые фильтры необходимые для компонентов из папки components.

В папке plugins находится подключения плагина Vuetify.

В папке router располагаются маршруты, которые сопоставляются с компонентами из папки components. Таким образом, на определённый маршрут есть соответствующий компонент, который и видит пользователь.

В папке services находятся так называемые сервисы, которые и отправляют запросы от клиентской части на сервер с помощью Axios.

И наконец в папке store находится наш vuex, который позволяет управлять состоянием нашего клиентского приложения.

### **5.2 Аутентификация пользователей**

Аутентификация — процедура проверки подлинности, например: проверка подлинности пользователя путём сравнения введённого им пароля с паролем, сохранённым в базе данных пользовательских логинов Код функции, которая осуществляет аутентификацию админа находится в приложении В.

Для организации аутентификации использовался пакет Passport-jwt[5], на основе JSON веб-токенов. Ввиду своей невероятной гибкости и модульности, Passport может быть закинут в любое веб-приложение на основе Express.

# Заключение

За время прохождения практики были изучены организационная структура предприятия, информационные технологии, используемые на предприятии, методы разработки и ведения проектов на предприятии.

В результате прохождения практики было создано веб-приложение «Психологические тесты». В разработанном программном средстве были выполнены следующие задачи:

* сохранение рабочей информации в централизованной базе данных;
* регистрация новых пользователей;
* аутентификация пользователей;
* аутентификация администратора (психолога);
* просмотр пользователем доступных для прохождения тестов;
* прохождение пользователем выбранного теста;
* просмотр пользователем результатов пройденных тестов;
* просмотр администратором результатов пройденных тестов всех зарегистрированных пользователей.

Данное программное средство использует технологии Node.js, фреймворк Express, ORM-библиотеку Sequelize – для реализации back-end, для front-end части используется фреймворк Vue.js, Vuex, Vuetify, Axios. MS SQL в качестве СУБД.

Технологическая практика позволила закрепить знания, полученные на протяжении обучения в университете, а также получить новые, широко используемые во многих IT сферах.

# Список использованных источников

1. Vuejs [Электронный ресурс] – Режим доступа: <https://ru.vuejs.org/>
2. Dou.ua [Электронный ресурс] – Режим доступа: <https://dou.ua/lenta/articles/five-steps-to-vue-ssr/>
3. Internet-technologies.ru [Электронный ресурс] – Режим доступа: <https://www.internet-technologies.ru/articles/razrabotka-masshtabnogo-prilozheniya-vue-js.html>
4. Metanit [Электронный ресурс] – Режим доступа: https://metanit.com/web/nodejs/9.1.php
5. Passportjs [Электронный ресурс] – Режим доступа: <http://www.passportjs.org/packages/passport-jwt/>
6. Хабр [Электронный ресурс] – Режим доступа: <https://habr.com/ru/company/ruvds/blog/340750/>
7. Vuetify [Электронный ресурс] – Режим доступа: <https://vuetifyjs.com/ru/getting-started/quick-start/>
8. Vuex.vuejs [Электронный ресурс] – Режим доступа: <https://vuex.vuejs.org/ru/guide/>

# Приложение А

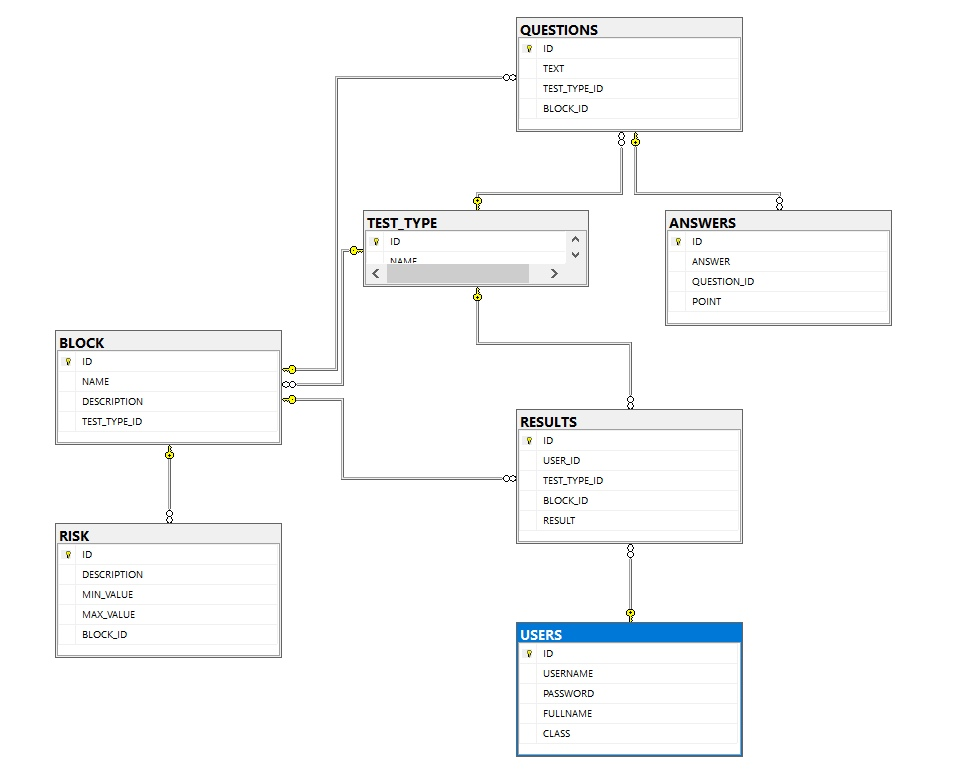


Рисунок 1 – Модель базы данных

# Приложение Б

Листинг 1 – Схема данных таблицы USERS

const Sequelize = require('sequelize');

const sequelize = require('../config/sequelize');

module.exports = sequelize.define('USERS', {

ID: {

type: Sequelize.INTEGER,

autoIncrement: true,

primaryKey: true,

allowNull: false

},

USERNAME: {

type: Sequelize.STRING,

},

PASSWORD:{

type: Sequelize.STRING,

},

FULLNAME: {

type: Sequelize.STRING,

},

CLASS: {

type: Sequelize.STRING,

}},

{

freezeTableName: true,

tableName: 'USERS',

timestamps: false

}

);

Листинг 2 – Схема данных таблицы TEST\_TYPE

module.exports = sequelize.define('TEST\_TYPE', {

ID: {

type: Sequelize.INTEGER,

autoIncrement: true,

primaryKey: true,

allowNull: false

},

NAME: {

type: Sequelize.STRING,

}},

{

freezeTableName: true,

tableName: 'TEST\_TYPE',

timestamps: false

}

);

Листинг 3 – Схема данных таблицы RISK

module.exports = sequelize.define('RISK', {

ID: {

type: Sequelize.INTEGER,

autoIncrement: true,

primaryKey: true,

allowNull: false

},

DESCRIPTION:{

type: Sequelize.STRING,

},

MIN\_VALUE: {

type: Sequelize.INTEGER,

},

MAX\_VALUE: {

type: Sequelize.INTEGER,

},

BLOCK\_ID: {

type: Sequelize.INTEGER,

}},

{

freezeTableName: true,

tableName: 'RISK',

timestamps: false

}

);

Листинг 4 – Схема данных таблицы RESULTS

module.exports = sequelize.define('RESULTS', {

ID: {

type: Sequelize.INTEGER,

autoIncrement: true,

primaryKey: true,

allowNull: false

},

USER\_ID: {

type: Sequelize.INTEGER,

},

TEST\_TYPE\_ID:{

type: Sequelize.INTEGER,

},

BLOCK\_ID: {

type: Sequelize.INTEGER,

},

RESULT: {

type: Sequelize.INTEGER,

}},

{

freezeTableName: true,

tableName: 'RESULTS',

timestamps: false

}

);

Листинг 5 – Схема данных таблицы QUESTIONS

module.exports = sequelize.define('QUESTIONS', {

ID: {

type: Sequelize.INTEGER,

autoIncrement: true,

primaryKey: true,

allowNull: false

},

TEXT: {

type: Sequelize.STRING,

},

TEST\_TYPE\_ID:{

type: Sequelize.INTEGER,

},

BLOCK\_ID: {

type: Sequelize.INTEGER,

}},

{

freezeTableName: true,

tableName: 'QUESTIONS',

timestamps: false

}

);

Листинг 6 – Схема данных таблицы BLOCK

module.exports = sequelize.define('BLOCK', {

ID: {

type: Sequelize.INTEGER,

autoIncrement: true,

primaryKey: true,

allowNull: false

},

NAME: {

type: Sequelize.STRING,a

},

DESCRIPTION:{

type: Sequelize.STRING,

},

TEST\_TYPE\_ID: {

type: Sequelize.INTEGER,

}},

{

freezeTableName: true,

tableName: 'BLOCK',

timestamps: false

}

);

Листинг 7 – Схема данных таблицы ANSWERS

module.exports = sequelize.define('ANSWERS', {

ID: {

type: Sequelize.INTEGER,

autoIncrement: true,

primaryKey: true,

allowNull: false

},

ANSWER: {

type: Sequelize.STRING,

},

QUESTION\_ID:{

type: Sequelize.INTEGER,

},

POINT: {

type: Sequelize.INTEGER,

}},

{

freezeTableName: true,

tableName: 'ANSWERS',

timestamps: false

}

);

# Приложение В

Листинг 8 – Функция, осуществляющая аутентификацию админа

api.login = (User) => (req, res) => {

User.findOne({ login: req.body.login }, (error, user) => {

if (error) throw error;

if (!user) res.status(401).send({ success: false, message: 'Authentication failed. User not found.' });

else {

user.comparePassword(req.body.password, (error, matches) => {

if (matches && !error) {

const token = jwt.sign({ user }, config.secret);

res.json({ success: true, message: 'Token granted', token, user: user });

} else {

res.status(401).send({ success: false, message: 'Authentication failed. Wrong password.' });

}

});

}

});

}

Листинг 9 – Функция сравнения паролей

User.methods.comparePassword = function (password, callback) {

bcrypt.compare(password, this.password, (error, matches) => {

if (error) return callback(error);

callback(null, matches);

});

};