**Реферат**

ФИО

Подпись

Дата

Лист

1

БГТУ 00.00.ПЗ

Разраб.

*Лещенко А.О.*

Провер.

*Сухорукова И.Г.*

Н. контр.

*Рыжанкова А.С.*

Утверд.

Пацей Н.В.

Реферат

Лит.

Листов

1

*74417024, 2020*

У

**Содержание**

ФИО

Подпись

Дата

Лист

1

БГТУ 00.00.ПЗ

Разраб.

*Лещенко А.О.*

Провер.

*Сухорукова И.Г.*

Н. контр.

*Рыжанкова А.С.*

Утверд.

Пацей Н.В.

Содержание

Лит.

Листов

1

*74417024, 2020*

У

**Введение**

ФИО

Подпись

Дата

Лист

1

БГТУ 00.00.ПЗ

Разраб.

*Лещенко А.О.*

Провер.

*Сухорукова И.Г.*

Н. контр.

*Рыжанкова А.С.*

Утверд.

Пацей Н.В.

Введение

Лит.

Листов

1

*74417024, 2020*

У

Чтобы достичь успеха в любом своем начинании каждый человек ставит перед собой цели, к которым он может идти как на протяжении короткого участка времени, так и до конца жизни. С взрослением количество планов и стремлений только увеличивается. В различных сферах существуют специалисты, способные обучить своему умению других людей. Таких людей называют менторами. В широком смысле ментор – это опытный и старший профессионал, который помогает человеку в учебе (карьере), дает советы.

Процесс нахождения подходящего человека, который обладает определенным навыком на достаточно высоком уровне, не всегда прост из-за большого количества образовательных программ с различными возможностями.

Все вышесказанное касается не только людей как таковых, но и компаний, в которых они работают. Правильная постановка своей активности означает, как правило, повышение по карьерной лестнице.

В реалиях современного мира, чем выше уровень способностей у человека, тем выше его загруженность, так как востребованность в высококвалифицированных специалистах растет.

Чаще всего для повышения квалификации сотрудников, применяется коллективное обучение, однако при таком подходе уделяется меньше времени для каждого студента.

Целью данного проекта является создание веб-приложения, которое воссоздает функционал корпоративного продукта, способного организовывать развитие навыков сотрудников с возможностью нахождения и выбора руководителя.

Для достижения цели были поставлены следующие задачи:

* проанализировать рынок на наличие подобных корпоративных решений с целью изучения их основных возможностей;
* разработать UML-диаграммы, которые позволят более подробно и наглядно описать систему;
* разработать модель базы данных, что является одной из главных задач всей разработки, так как благодаря именно грамотной структуре хранения информации можно обеспечить долговременную целостность, простоту обслуживания данных и максимальное уменьшение материальных затрат на них;
* разработать приложение с использованием многоуровневой архитектуры, так как это позволит создать гибкий продукт;
* разработать серверную часть веб-приложения, построенного на Node.js;
* примененить аутентификацию, использующую JWT токены;
* разработать приложение-клиент, используя Java Script библиотеку React.js для асинхронного рендеринга контента.

1. **Аналитический обзор литературы и постановка задачи**

ФИО

Подпись

Дата

Лист

1

БГТУ 01.00.ПЗ

Разраб.

*Лещенко А.О.*

Провер.

*Сухорукова И.Г.*

Н. контр.

*Рыжанкова А.С.*

Утверд.

*Пацей Н.В.*

1 Аналитический обзор литературы и постановка задачи

Лит.

Листов

9

*74417024, 2020*

В современном мире веб-приложения для образования являются очень популярными, и с каждым днем их становится все больше. Над обучением студентов различным навыкам работает огромное количество профессионально обученных людей.

Сферы навыков очень разнообразны: обучение игре на музыкальных инструментах, обучение программированию, различным техникам рисования, развитию логики, тренировка памяти и многое другое.

Также стоит учесть, что обучение может быть организовано как с реальным человеком, так и с запрограммированным кодом, то есть без участия преподавателя. Обучение с реальным человеком обладает большими преимуществами, так как такой способ позволяет преподавателю в случае недопониманий со стороны студента объяснить изучаемый материал иным образом.

Одним из ключевых моментов в обучении является количество студентов, находящихся в одной группе. Чем больше людей в коллективе, тем менее эффективным может стать сам процесс обучения, так как невозможно за небольшой промежуток времени уделить внимание каждому и убедиться в хорошем усвоении пройденного материала. По этой причине индивидуальный подход является более результативным в большей части случаев.

Многие платформы по обучению не предоставляют возможность выбора себе преподавателя, который будет курировать деятельность студента, что важно, так как у всех людей разные подходы к обучению и все индивидуально.

Как правило, специалисты высокого уровня намного более заняты на проектах в связи со своим опытом, способностью решать более сложные задачи, поэтому ментору важно предоставить возможность отклонить запрос студента.

Итогом обучения служит решение, подтверждающее либо отклоняющее освоения навыка, изучаемого студентом. Такое решение принимает ментор. В случае отрицательного решения должна предполагаться возможность повторной отправки запроса ментору.

Для построения успешного и удобного с точки зрения пользователя приложения, необходимо учитывать пользовательский интерфейс. Для наглядного представления статуса обучения удобно использовать такой интерфейс, который позволит однозначно понять, на каком именно уровне находится тот или иной навык. Для примера, можно использовать несколько колонок, первая из которых будет отображать уже освоенные навыки, вторая включает в себя очередь ожидания подтверждения запросов на обучение, и третья может предоставлять список навыков, которые уже находятся в процессе тренинга.

* 1. **Анализ существующих систем**

В настоящее время на рынке представлено большое количество приложений, в которых есть возможность развивать различные навыки с отслеживанием своего прогресса. Что касается корпоративных приложений, то их не так много и в основном они доступны только в рамках компании, но все же можно рассмотреть несколько прототипов веб-приложения, разработанного в данном проекте.

* + 1. **Приложение Skills Score**

Одним из аналогов является веб-приложение «Skills Score». Данный продукт является разработкой компании «iTechArt». Веб-приложение позволяет регистрироваться сотрудникам, работающим в компании. Здесь может быть создан свой профиль, в котором описывается информация о работнике, его фотография, имя и другое.

Несомненным преимуществом данного приложения является то, что пользователю предоставляется возможность выбрать категорию навыков, которую он может изучить, после чего появляется соответствующий список навыков в правой части экрана с указанием названия и уровня, на который можно изучить выбранный пункт. Таким образом, пользователь может выбрать для себя цель. Данная функциональность приложения продемонстрирована на рисунке 1.1.

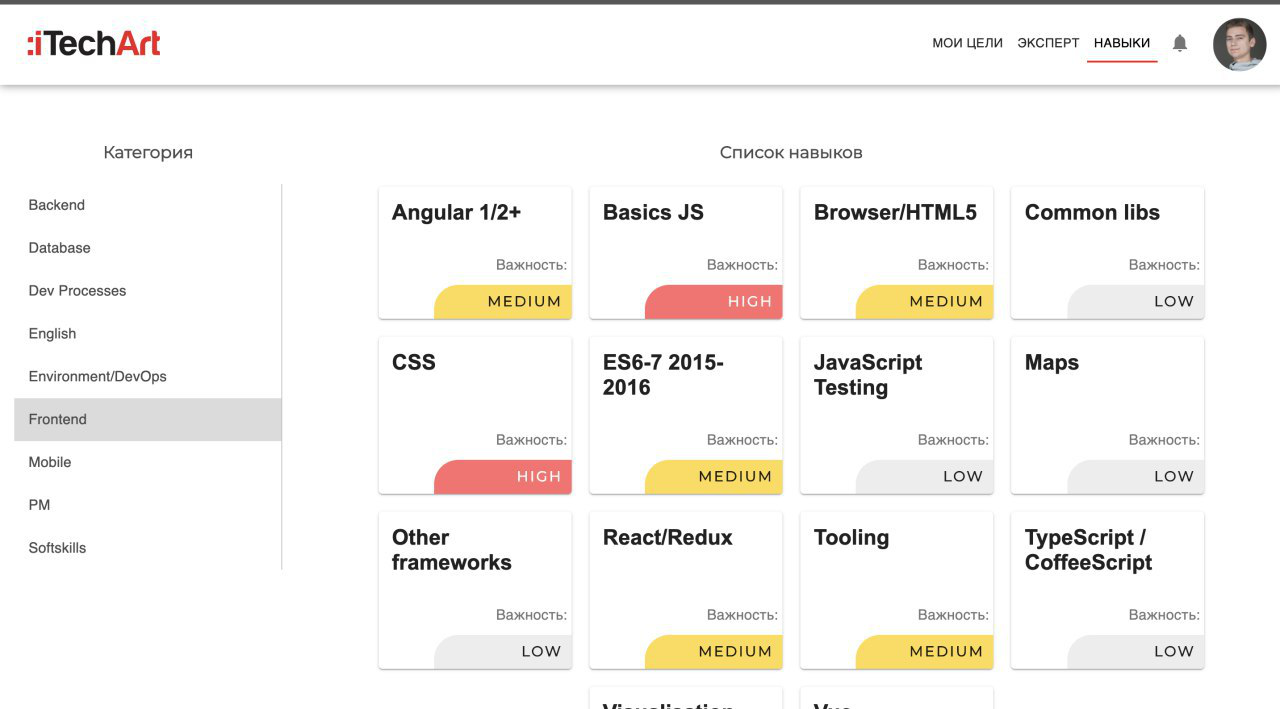


Рисунок 1.1 – Страница навыков веб-приложения «Skills Score»

Еще одним преимуществом данного приложения является наглядность отслеживания целей. На странице «Мои цели» имеется 5 колонок: «Созданные», «Начатые», «В ожидании назначения», «Назначенные» и «Завершенные», каждая из которых имеет свою смысловую нагрузку.

Веб-приложение «Skills Score» позволяет просмотреть текущие цели пользователя с отслеживание статуса цели. Данная функция продемонстрирована на рисунке 1.2.

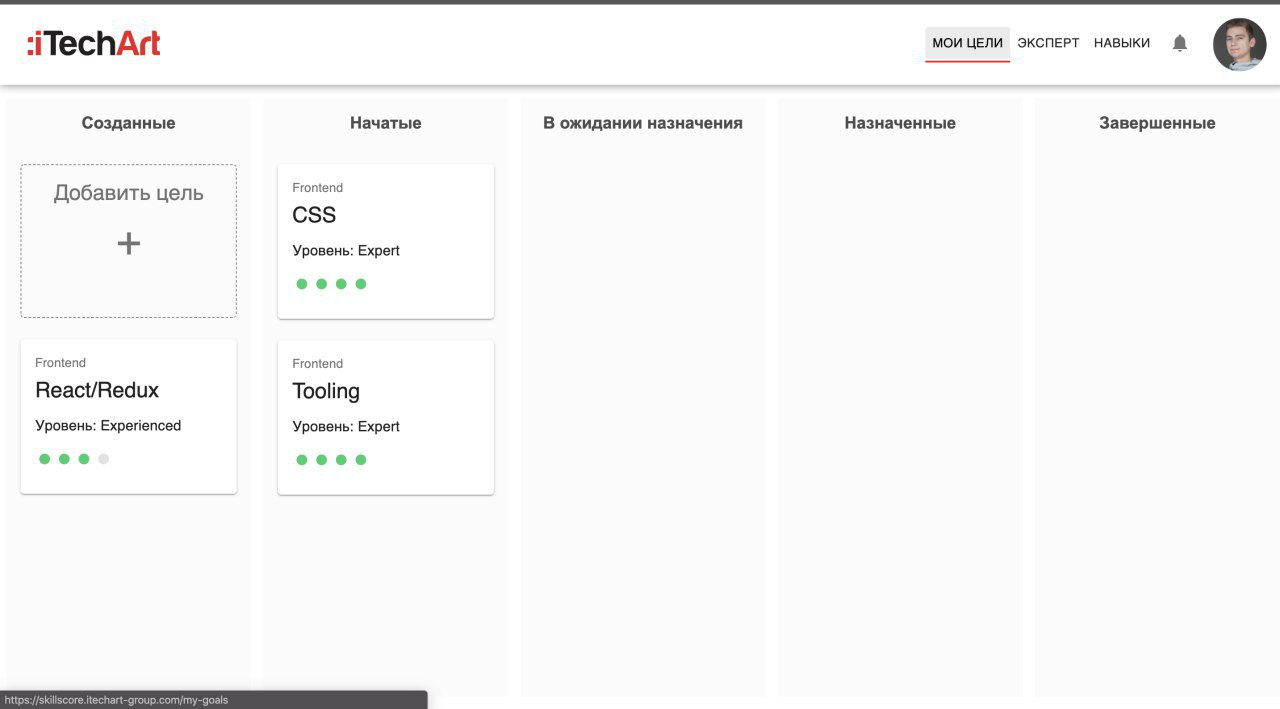


Рисунок 1.2 – Страница целей веб-приложения «Skills Score»

Таким образом, веб-приложение «Skills Score» является очень удобным для пользователя за счет правильно организованного пользовательского интерфейса и возможности фильтрации поиска навыка в соответствии с выбранной категорией.

* + 1. **Приложение Grow**

Вторым прототипом является веб-приложение «Grow», разработанное компанией «EPAM».

Удобной является функция аутентификации с помощью существующего аккаунта почты, что значительно упрощает вход в систему. Также регистрация доступна с помощью учетных записей таких приложений, как «Вконтакте», «Facebook», «Github», «Twitter» и некоторых других.

План изучения цели можно скачать, и полученный файл будет иметь формат «xlsx». В документе отображается название, дата начала, окончания, статус, условие освоения, дата создания, обновления этапа изучения всех этапов выбранного навыка.

Для проверки своих работ можно назначить ментора, который будет курировать и контролировать процесс достижения студентом выбранной цели. Для этого достаточно ввести существующий адрес электронной почты своего руководителя.

Также, веб-приложение «Grow» имеет приятный графический интерфейс в спокойных тонах.

Данное приложение позволяет добавлять цели с возможностью установки статуса: «Planned», «In Progress», «Done» и «Canceled». Причем в данном веб-приложении предоставлена возможность планирования изучения отдельных тем, связанных с целью. Страница с данной функцией продемонстрирована на рисунке 1.3.

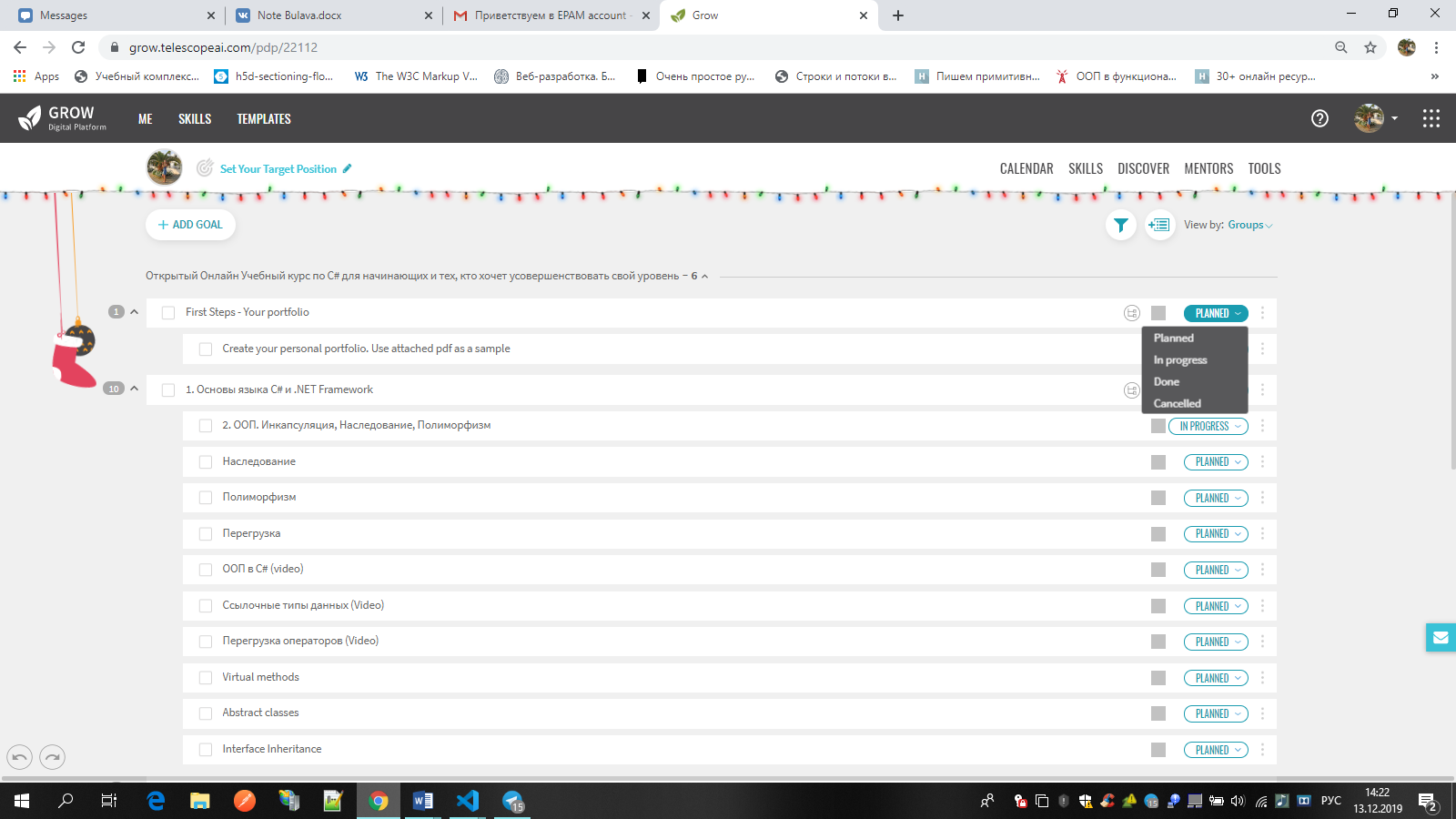


Рисунок 1.3 – Страница целей веб-приложение «Grow»

Рассматриваемое приложение позволяет выбрать пользователю свою текущую позицию. Данная функция продемонстрирована на рисунке 1.4.

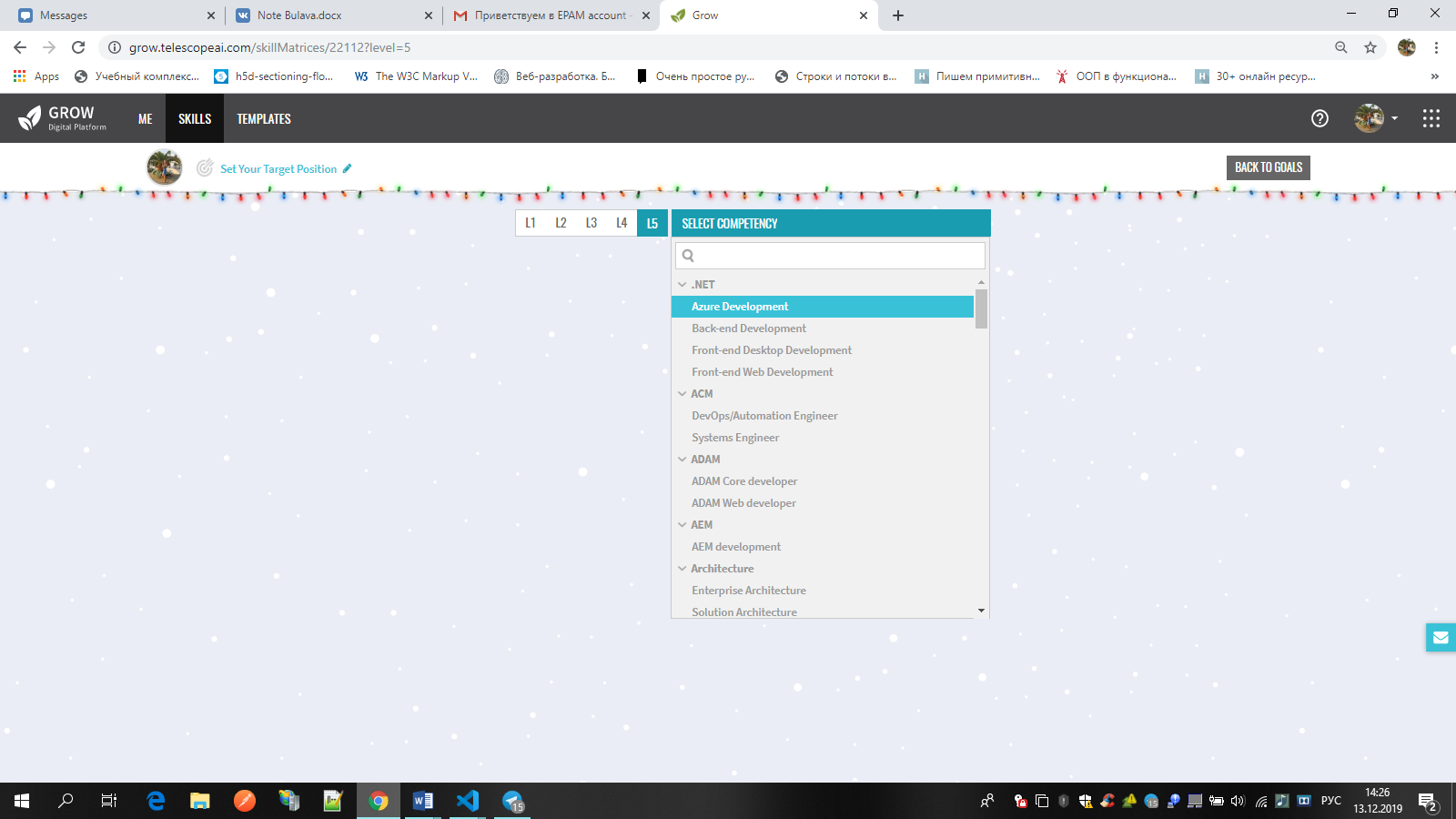


Рисунок 1.4 – Страница навыков веб-приложения «Grow»

Таким образом, данный аналог очень похож на приложение «Skills Score».

* + 1. **Приложение Goal Setting Daily Planner**

Данный продукт представляет мобильное приложение, которое позволяет пользователям создавать цели, к которым они могут стремиться.

Данное приложение рассчитано для личного пользования, а не для компании. Программный продукт предоставляет возможность создать длительные цели или же задать распорядок с расписанием. Окно с выбором первой цели представлено на рисунке 1.5.

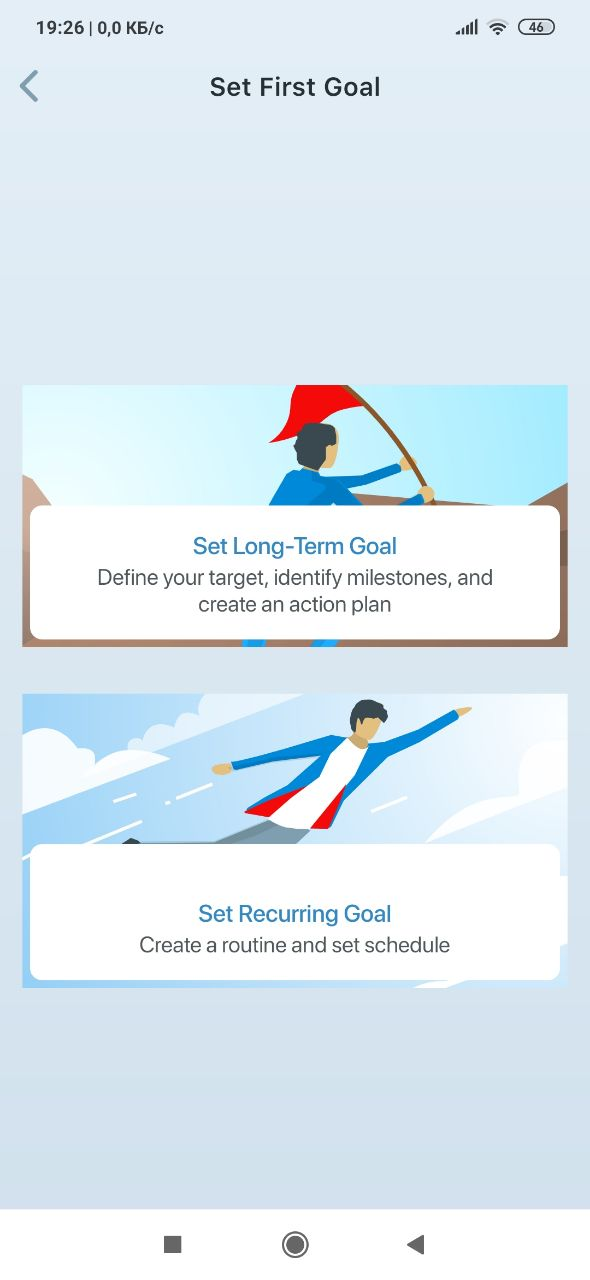


Рисунок 1.5 – Выбор первой цели

При выборе пункта, связанного с расписанием, можно задать категорию данного режима или же создать свою собственную.

Приложение позволяет добавлять заметки, создавать подзадачи, задавать количество повторений, время, когда цель должна быть достигнута, различные ссылки.

Преимуществом данного приложения является возможность добавления приоритета к цели. Он может принимать следующие уровни:

* normal;
* high;
* highest.

Также, пользователям предоставляется возможность добавить напоминание на свое устройство или же в Google календарь.

В приложении можно выделить несколько основных вкладок: «Today», «Goals», «Activities», «Journal», каждая из которых позволяет отслеживать процесс достижения цели.

После добавления цели, она отображается вместе с другими на вкладке «Goals». Окно с данной функцией представлено на рисунке 1.6.

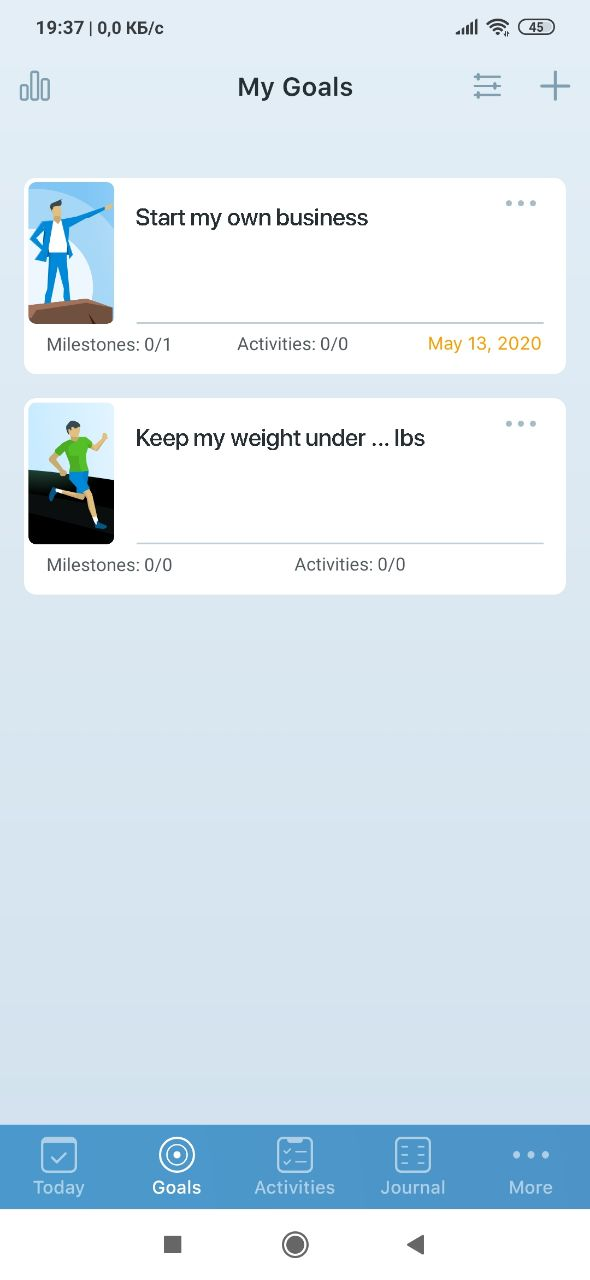


Рисунок 1.6 – Окно «Goals»

На экране, изображенном на рисунке 1.6, отображаются цели с представлением главной информации по каждой из них.

Таким образом, были рассмотрены основные аналоги разрабатываемого веб-приложения.

Разбор функциональности приложений позволил сформировать общее впечатление о том, что должно быть обязательно реализовано в дипломном проекте.

* 1. **Постановка задачи и выбор технологий для разработки приложения. Основные технические требования к разработке**

В результате сравнительного анализа существующих приложений, было решено создать веб-приложение, которое сможет объединить некоторые возможности всех аналогов, чтобы сделать приложение максимально удобным и простым в использовании.

Приложение должно выполнять следующие функции:

* позволять пользователю зарегистрироваться в системе;
* разграничивать права на использование приложения между тремя ролями пользователей: менеджер, ментор и студент;
* заполнение менеджером списка навыков и списка менторов;
* просмотр студентом доступным навыков для изучения;
* отправка запроса ментору;
* подтверждение или отклонение ментором запроса на обучение;
* просмотр текущих навыков и находящихся на стадии изучения.
  1. **Подходящие технологии для клиентского приложения**

Далее будут рассмотрены различные библиотеки, технологии и подходы, которые могут быть использованы в разработке клиентского приложения.

* + 1. **Язык программирования JavaScript**

JavaScript – это полноценный динамический язык программирования, который применяется к HTML документу, и может обеспечить динамическую интерактивность на веб-сайтах. Его разработал Brendan Eich, сооснователь проекта Mozilla, Mozilla Foundation и Mozilla Corporation.

JavaScript сам по себе довольно компактный, но очень гибкий. Разработчиками написано большое количество инструментов поверх основного языка JavaScript, которые разблокируют огромное количество дополнительных функций с очень небольшим усилием [1].

JavaScript – это легковесный, интерпретируемый или JIT-компилируемый, объектно-ориентированный язык с функциями первого класса. Наиболее широкое применение находит как язык сценариев веб-страниц, но также используется и в других программных продуктах, например, node.js или Apache CouchDB. JavaScript – это прототипно-ориентированный, мультипарадигменный язык с динамической типизацией, который поддерживает объектно-ориентированный, императивный и декларативный (например, функциональное программирование) стили программирования [2].

* + 1. **Библиотека ReactJS**

React – это JavaScript-библиотека для разработки пользовательского интерфейса. Его главная задача – обеспечение вывода на экран того, что можно видеть на веб-страницах. React значительно облегчает создание интерфейсов благодаря разбиению каждой страницы на небольшие фрагменты, которые называются компонентами. Каждый компонент – это JavaScript-функция, которая возвращает кусок кода, представляющего фрагмент страницы.

React использует язык программирования, называемый JSX, который похож на HTML, но работает внутри JavaScript, что отличает его от HTML.

Компоненты могут хранить информацию о текущей ситуации. Эта информация называется состоянием (state), она хранится в JS-объекте. Состояние – это инструмент, позволяющий обновлять пользовательский интерфейс, основываясь на событиях.

Компоненты могут «общаться» друг с другом, используя так называемые свойства (props). Свойства – это информация, коллективно используемая родительским компонентом и компонентами-потомками [3].

* + 1. **Стандарт ECMAScript**

ECMAScript — это встраиваемый расширяемый не имеющий средств ввода-вывода язык программирования, используемый в качестве основы для построения других скриптовых языков. Стандартизирован международной организацией ECMA в спецификации ECMA-262. JavaScript является расширением данного языка.

Разработка первой редакции спецификации началась в ноябре 1996 года. Принятие спецификации состоялось в июне 1997 года. В июне 1998 года общим собранием ECMA была принята вторая редакция ECMA-262, соответствующая ISO/IEC 16262. Третья редакция спецификации отличалась от предыдущей введением поддержки регулярных выражений, улучшением поддержки строк, введением новых управляющих конструкций, механизма исключений, форматирования при численном вводе и некоторыми другими изменениями [4].

Переломным моментом в истории EcmaScript является появление ES6, основными особенностями которого являются:

* стрелочные функции;
* классы;
* деструктурирующее присваивание;
* в шаблонных строках можно объявлять строки с помощью обратных кавычек;
* let и const – альтернативы var для объявления переменных;
* итератор и протокол итерации теперь определяют способ перебора любого объекта, а не только массивов. Symbol используется для присвоения итератора к любому объекту;
* добавлены функции-генераторы. Они используют yield для создания последовательности элементов. Функции-генераторы могут использовать yield для делегирования в другую функцию генератора, кроме этого они могут возвращать объект генератора, который реализует оба протокола;
* добавлены промисы.

В связи с вышеперечисленными пунктами можно сделать вывод, что данный стандарт значительно упрощает процесс написания кода и добавляет языку больше гибкости [5].

* 1. **Подходящие технологии для серверного приложения**

В данном разделе будут рассмотрены различные библиотеки, технологии и подходы, которые могут быть использованы в разработке серверного приложения.

* + 1. **Платформа Node.js**

Node.js – это серверная платформа, которая работает на движке Google Chrome – V8, который умеет компилировать JavaScript код в машинный код.

Node.js использует событийно-ориентированную модель и неблокирующую ввод/вывод архитектуру, что делает его легковесным и эффективным. Это не фреймворк и не библиотека, а среда выполнения JavaScript [6].

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.

* + 1. **Веб-фреймворк Express**

Express – это один из самых популярных фрейморков Node.js. Он предоставляет такие механизмы, как:

* написание обработчиков запросов HTTP с различными маршрутами;
* интеграция с механизмами рендеринга «view» для генерации ответов путем вставки данных в шаблоны;
* задание общих параметров веб-приложения, таких как порт, используемый для подключения, и расположение шаблонов, используемых для визуализации ответа;
* добавление дополнительной обработки запросов «middleware» в любой точке конвейера обработки запросов [7].
  1. **Вывод по разделу**

Таким образом, были рассмотрены аналоги разработанного веб-приложения, которые имеют свои преимущества и недостатки. На основе них было принято решение о том, что необходимо включить в функциональную часть приложения, чтобы оно было удобно для пользователя.

Можно выделить разные подходы для построения приложений с использованием различных технологий. В данном дипломном проекте было принято решение использовать Node.js и Express для реализации серверной части в соответствии с многоуровневой архитектурой. Клиентская часть будет выполнена с использованием React.js.

1. **Проектирование**

ФИО

Подпись

Дата

Лист

1

БГТУ 02.00.ПЗ

Разраб.

*Лещенко А.О.*

Провер.

*Сухорукова И.Г.*

Н. контр.

*Рыжанкова А.С.*

Утверд.

Пацей Н.В.

2 Проектирование

Лит.

Листов

7

*74417024, 2020*

У

Проектирование программного обеспечения – процесс создания проекта программного обеспечения. Целью проектирования является определение внутренних свойств системы и детализации её внешних (видимых) свойств на основе выданных заказчиком требований к ПО.

* 1. **Проектирование базы данных**

Одним из ключевых моментов проектирования является создание схемы базы данных. Она включает в себя описание содержания, структуры и ограничений целостности, используемые для создания и поддержки базы данных.

PostgreSQL является объектно-реляционной системой управления базами данных (ОРСУБД) на основе POSTGRES, версия 4.2, разработанной в Университете Калифорнии в Беркли департаменте компьютерных наук.

PostgreSQL является open source потомком оригинального кода Berkeley. Он поддерживает большую часть стандарта SQL и предлагает множество современных функций,таких как:

* сложные запросы;
* внешние ключи;
* триггеры;
* представление;
* транзакционная целостность (transactional integrity);
* управление конкурентным доступом с помощью многоверсионности.

Кроме того, PostgreSQL может быть расширен пользователем во многих отношениях, например, путем добавления новых:

* + - типов данных;
    - функций;
    - операторов;
    - агрегатных функций;
    - индекс методов;
* процедурных языков [8].

В проекте была разработана схема базы данных, состоящая из 6 таблиц. Логическая схема базы данных изображена на ДП 01.00.ГЧ. Логическая схема базы данных изображена в приложении А.

* + 1. **Таблица «Users»**

Таблица «Users» хранит в себе информацию о пользователе. В таблице 2.1 рассмотрен перечень полей данной сущности.

Таблица 2.1 – Поля таблицы «Users»

|  |  |  |
| --- | --- | --- |
| Наименование поля | Тип данных | Описание |
| Id | Integer | Идентификатор, первичный ключ |
| Email | Char | Электронная почта пользователя, уникальное поле |
| Password | Char | Пароль пользователя |
| FirstName | Char | Имя пользователя |
| LastName | Char | Фамилия пользователя |
| MiddleName | Char | Отчество пользователя |
| Role | Char | Роль пользователя, принимает одно из четырех значений: «admin», «manager», «mentor», «student» |

Обязательными полями в таблице «Users» являются email и password, а id генерируется средствами базы данных.

* + 1. **Таблица «Mentors»**

Таблица «Mentors» хранит информацию о пользователях, являющихся руководителями. Перечень полей сущности «Mentor» представлен в таблице 2.2.

Таблица 2.2 – Поля таблицы «Mentors»

|  |  |  |
| --- | --- | --- |
| Наименование поля | Тип данных | Описание |
| Id | Integer | Идентификатор, первичный ключ |
| User\_id | Integer | Внешний ключ на таблицу «Users» поля id |
| Status | Boolean | Флаг, отображающий занятость руководителя |

Отношение таблицы «Users» к таблице «Mentors» – «один-к-одному».

* + 1. **Таблица «Skills»**

Таблица «Skills» хранит информацию о навыках. В ней находятся поля, представленные в таблице 2.3.

Таблица 2.3 – Поля таблицы «Skills»

|  |  |  |
| --- | --- | --- |
| Наименование поля | Тип данных | Описание |
| Id | Integer | Идентификатор, первичный ключ |
| Name | Char | Название навыка |

Окончание таблицы 2.3

|  |  |  |
| --- | --- | --- |
| Level\_name | Char | Название уровня |
| Time\_level | Integer | Количество месяцев, необходимое на освоение навыка |
| Description | Char | Описание навыка |

Поле «Level\_name» может принимать одно из трех значений: «junior», «middle» или «senior».

* + 1. **Таблица «UserSkills»**

Таблица «UserSkills» предназначена для организации связи «многие-ко-многим» между сущностями «User» и «Skill», поэтому она содержит поля, представленные в таблице 2.4.

Таблица 2.4 – Поля таблицы «UserSkills»

|  |  |  |
| --- | --- | --- |
| Наименование поля | Тип данных | Описание |
| Id | Integer | Идентификатор, первичный ключ |
| User\_id | Integer | Внешний ключ на таблицу «Users» поля id |
| Skill\_id | Integer | Внешний ключ на таблицу «Skills» поля id |
| Mentor\_id | Integer | Внешний ключ на таблицу «Mentors» поля id |
| Is\_approved\_request | Boolean | Флаг, отображающий подтвержденность запроса на освоение навыка |
| Is\_approved\_skill | Boolean | Флаг, отображающий подтвержденность прохождения навыка |

В данной таблице группа полей user\_id-skill\_id является уникальной.

* + 1. **Таблица «Students»**

Таблица «Students» хранит информацию о студентах и содержит поля, представленные в таблице 2.5.

Таблица 2.5 – Поля таблицы «Students»

|  |  |  |
| --- | --- | --- |
| Наименование поля | Тип данных | Описание |
| Id | Integer | Идентификатор, первичный ключ |

Окончание таблицы 2.5

|  |  |  |
| --- | --- | --- |
| Userskill\_id | Integer | Внешний ключ на таблицу «UserSkills» поля id |
| Start\_time\_mentor | Timestamp with time zone | Дата и время, когда у студента появился руководитель |
| Finish\_time\_mentor | Timestamp with time zone | Дата и время, когда студент закончил освоение навыка |

Таблица «UserSkills» имеет отношение «один-ко-многим» с таблицей «Students».

* + 1. **Таблица «Goals»**

Таблица «Goals» хранит информацию о целях пользователя и содержит поля, представленные в таблице 2.6.

Таблица 2.6 – Поля таблицы «Goals»

|  |  |  |
| --- | --- | --- |
| Наименование поля | Тип данных | Описание |
| Id | Integer | Идентификатор, первичный ключ |
| Student\_id | Integer | Внешний ключ на таблицу «Students» поля id |
| Date\_review | Timestamp with time zone | Дата, когда будет решаться, подтвердил ли студент выбранный навык |

Таблица «Students» имеет отношение «один-ко-многим» с таблицей «Goals».

* 1. **Проектирование системы**

Для описания функционального назначения системы была разработана UML Use-Case диаграмма. Для начала, необходимо обозначить нотации, характерные для данного вида диаграммы и продемонстрированные в данном проекте: актеры, прецеденты, подсистема.

Актеры – множество ролей, которые пользователи могут осуществлять в процессе работы, причем в качестве пользователя может выступать как человек, так и система.

Прецедент – нотация диаграммы, которая отвечает на вопрос «что можно сделать?».

Подсистема – прямоугольник, в котором располагаются все прецеденты [9].

Построенная схема изображена на ДП 02.00.ГЧ. На ней можно выделить четырех актеров: «Пользователь», «Менеджер», «Студент» и «Руководитель».

«Пользователь» – это актер, от которого наследуются остальные, и ему доступен только один прецедент: «Войти в систему».

Прецеденты для актера «Менеджер» следующие:

* просмотреть таблицу «Skills»:
* добавить навык;
* изменить навык;
* удалить навык;
* просмотреть таблицу «Mentors»:
* добавить руководителя;
* просмотреть навыки руководителя;
* удалить руководителя.

Прецеденты для актера «Студент»:

* просмотреть текущие навыки;
* просмотреть доступные навыки:
* выбрать ментора:
* отправить запрос на изучение навыка;
* просмотреть очередь запросов на изучение навыка:
* отменить запрос;
* просмотреть подтвержденные запросы.

Прецеденты для актера «Руководитель»:

* просмотреть запросы на изучение навыка:
* принять запрос;
* отклонить запрос;
* просмотреть текущих студентов:
* подтвердить навык.

В целом, диаграмма вариантов использования преследует следующие цели:

* + определение общих границ и контекста моделируемой предметной области на начальных этапах проектирования;
  + формулировка общих требований к функционированию поведения информационной системы;
  + разработка исходной концептуальной модели системы для ее последующей детализации в форме логических и физических моделей;
  + подготовка исходной документации для взаимодействующих разработчиков системы с ее заказчиками и пользователями [10].

В настоящее время тема безопасности очень важна для всех пользователей приложений, независимо от платформы и вида приложения.

Для обеспечения ограничения доступа к веб-приложению неавторизованных пользователей в данном проекте используется аутентификация.

Аутентификация – это процедура проверки подлинности введенных пользователем данных. С данным понятием также связаны такие термины, как идентификация и авторизация.

Идентификация – это заявление пользователя о себе, у каждого пользователя есть свой идентификатор.

Авторизация – предоставление пользователю прав на выполнение определенных действий в системе.

Выделяют различные виды аутентификации: Forms, Basic, Digest, OAuth, OpenID и некоторые другие.

В данном проекте продемонстрирована аутентификация на основе JWT-токенов.

JSON Web Token (JWT) – это JSON объект, который определен в открытом стандарте RFC 7519. Он считается одним из безопасных способов передачи информации между двумя участниками. Для его создания необходимо определить заголовок (header) с общей информацией по токену, полезные данные (payload), такие как id пользователя, его роль и так далее и подписи (signature) [11].

* 1. **Проектирование серверной части**

В программной инженерии многоуровневая архитектура или многослойная архитектура – клиент-серверная архитектура, в которой разделяются функции представления, обработки и хранения данных. Наиболее распространённой разновидностью многоуровневой архитектуры является трёхуровневая архитектура.

N-уровневая архитектура приложения предоставляет модель, по которой разработчики могут создавать гибкие и повторно-используемые приложения. Разделяя приложение на уровни абстракции, разработчики приобретают возможность внесения изменений в какой-то определённый слой, вместо того, чтобы перерабатывать все приложение целиком. Трехуровневая архитектура обычно состоит из уровня представления, уровня бизнес-логики и уровня хранения данных [12].

В данном дипломном проекте было решено разделить серверную часть на четыре уровня: работа с данными, сервисы, контроллеры и роутеры.

Для аутентификации будет использоваться промежуточный слой, способный фильтровать запросы и контролировать доступ к ресурсам приложения.

* 1. **Проектирование клиентской части**

На заре появления веб-технологий типичный сайт выглядел как набор статичных HTML-страниц и старый веб работал таким образом, что браузеру нужно было запрашивать каждую страницу сайта отдельно, заставляя сервер генерировать HTML-страницы и отсылать их обратно. Поэтому каждый новый запрос на сервер сопровождался бесконечной перезагрузкой страниц.

С развитием технологий все серверные операции стали выполняться на лету, а страницы сайта в веб-браузере стали обновляться мгновенно. В популярность вошли одностраничные приложения, состоящие из одной HTML страницы-оболочки с динамически подгружаемым контентом. Отличным примером одностраничного приложения может быть Gmail или страницы социальных сетей, где переключение между вкладками происходит практически моментально, без перезагрузки страницы [13].

Отличительными особенностями одностраничных приложений являются:

* разделение на клиентскую и серверную часть, которые могут разрабатываться, при необходимости, отдельными командами, а в случае, если нужно поменять интерфейс приложения, не нужно затрагивать бизнес-логику, или наоборот;
* высокая скорость загрузки.

Для построения клиентской части веб-приложения была выбрана библиотека React.js, позволяющая создать одностраничное приложение.

Основные преимущества React перед другими фреймворками следующие:

* virtual DOM может повысить производительность высоконагруженных приложений, что может снизить вероятность возникновения возможных неудобств и улучшает пользовательский опыт;
* использование изоморфного подхода помогает производить рендеринг страниц быстрее, тем самым позволяя пользователям чувствовать себя более комфортно во время работы с приложением. Поисковые системы индексируют такие страницы лучше. Поскольку один и тот же код может быть использован как в клиентской, так и в серверной части приложения, нет необходимости в дублировании одного и того же функционала. В результате время разработки и затраты снижаются;
* благодаря переиспользованию кода стало гораздо проще создавать мобильные приложения. Код, который был написан во время создания сайта, может быть снова использован для создания мобильного приложения. Если вы планируете использовать не только сайт, но и мобильное приложение, нет необходимости нанимать две большие команды разработчиков.
  1. **Вывод по разделу**

В данном разделе были подробно рассмотрены шаги проектирования веб-приложения для повышения квалификации сотрудников.

В результате этапа проектирования информационной системы была разработана схема базы данных и основная архитектура приложения.

Для написания серверной части будет использоваться подход, использующий N-layer архитектуру, так как это обеспечит гибкость всего приложения в дальнейшем и позволит расширять его.

Клиентская часть будет представлять одностраничное приложение, которое является более производительным и гибким.

В целом, архитектура данного веб-приложения предполагает максимальную независимость всех ее частей с целью более грамотного и гибкого разделения при необходимости внесения изменений.

1. **Реализация программного средства**

ФИО

Подпись

Дата

Лист

1

БГТУ 03.00.ПЗ

Разраб.

*Лещенко А.О.*

Провер.

*Сухорукова И.Г.*

Н. контр.

*Рыжанкова А.С.*

Утверд.

Пацей Н.В.

3 Реализация программного средства

Лит.

Листов

10

*74417024, 2020*

У

Разработка архитектуры проекта – одна из важнейших задач в процессе работы над приложением, так как в зависимости от неё определяется уровень зависимости компонентов приложения и насколько легко будут расширяемы её составные части.

* 1. **Разработка серверной части приложения**

Серверная часть программного продукта представляет собой веб-приложение с многоуровневой архитектурой, где можно выделить несколько слоев:

* данные;
* бизнес-логика;
* контроллеры;
* маршрутизаторы;
* промежуточный слой.
  + 1. **Реализация слоя данных**

Уровень доступа к данным содержит сущности, соответствующие таблицам базы данных, а также схемы взаимодействия друг с другом.

Для управления и отслеживания изменений, связанных с базой данных, используются миграции. С помощью них можно трансформировать существующую базу данных в другое состояние и откатывать изменения. Данные состояния сохраняются в миграционных файлах, в которых присутствует описание того, как прийти к новому состоянию и вернуться к предыдущему.

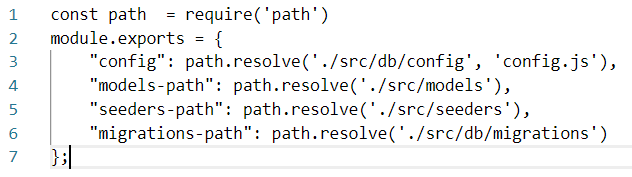
Для реализации данного подхода используется Sequelize Command-Line Interface (CLI).

Для построения общей структуры данного слоя используется команда «npx sequelize-cli init». В результате ее выполнения генерируются следующие папки:

* config, содержит конфигурационный файл, в котором указывается информация для CLI, чтобы подключиться к базе данных;
* models, содержит все модели;
* migrations, содержит все миграционные файлы;
* seeders, содержит файлы начальной инициализации.

Для задания переменных среды используется файл с названием «.env», в котором записываются строки в формате «КЛЮЧ=ЗНАЧЕНИЕ», причем пустые строки игнорируются. Таким образом можно задать значения для номера порта, на котором будет работать сервер или другие конфигурационные параметры.

Также, для изменения пути к папкам config, models, migrations и seeders можно создать специальный конфигурационный файл «.sequelizerc», пример которого изображен на листинге 3.1.



Листинг 3.1 – Пример специального конфигурационного файла

С помощью файла, расположенного в папке «models», можно задать структуру таблицы и отобразить связи между ними. Пример файла моделей представлен на листинге 3.2.

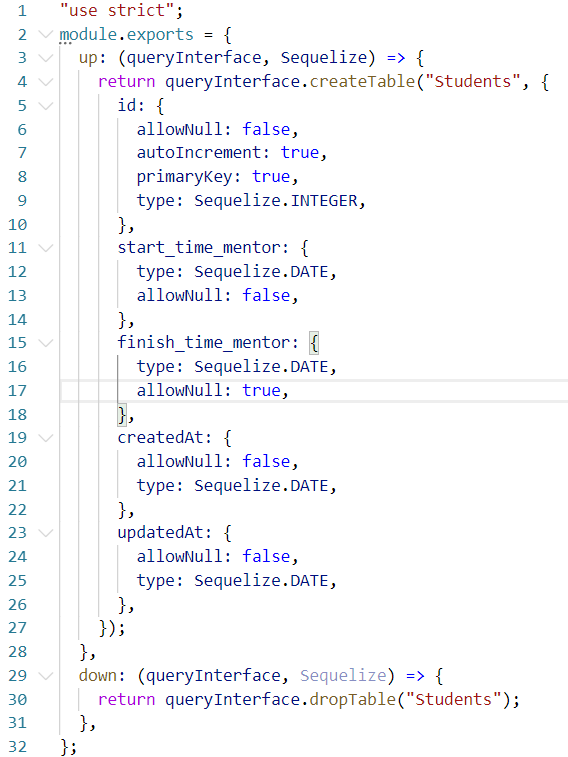


Листинг 3.2 – Пример файла модели

Для определения модели необходимо использовать метод «define», где в качестве параметров передаются ее название и поля с указанием типов данных. Для задания связей служит свойство «associate». Для создания связи «многие-к-одному» используется метод «belongsToMany».

Модели служат лишь для управления таблицами в рамках самого сервера, но для перенесения их в базу данных необходимо создать миграционный файл.

Для создания файла модели и соответствующего миграционного файла используется команда «model:generate», где в качестве параметров можно задать название модели и ее атрибуты, перечисленные в формате «название\_поля:тип\_данных». В результате будет сгенерировано два файла: модели и миграций. Пример миграционного файла представлен на листинге 3.3.



Листинг 3.3 – Пример миграционного файла

Как можно увидеть на рисунке 3.3, в файле миграции можно выделить два метода: up и down. Первый служит для задания названия и полей будущей таблицы с указанием их типов данных, а второй – для удаления таблицы.

Чтобы применить данные файлы для создания таблицы в базе данных, необходимо использовать команду «npx sequelize-cli db:migrate». Данная операция выполняет следующие шаги:

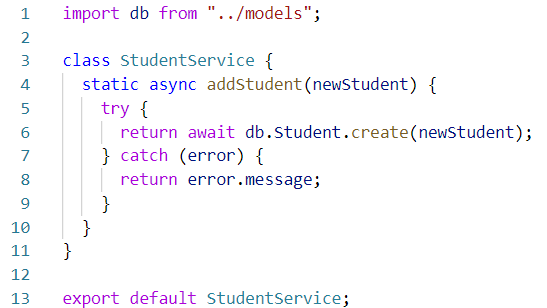
* создает таблицу «SequelizeMeta» в случае, если ее нет, причем она используется для хранения записей о текущих миграциях;
* ищет миграционные файлы, которые еще не были применены, и использует для этого записи из таблицы «SequelizeMeta»;
* создает таблицу с заданным названием.

Для того, чтобы вернуть состояние базы данных в предыдущее значение, необходимо выполнить команду «npx sequelize-cli db:migrate:undo». Причем, можно откатить изменения как для одной таблицы, так и для всех сразу. Также, миграции можно использовать для создания ограничений целостности, например, для задания уникальных ключей.

* + 1. **Реализация слоя бизнес-логики**

Уровень бизнес-логики при тесном взаимодействии с уровнем сервисов являются посредниками между сущностями базы данных и моделями для UI, сочетая в себе сервисы и классы, реализующие основной функционал всего разрабатываемого приложения.

Пример сервиса можно увидеть на листинге 3.4.



Листинг 3.4 – Пример сервиса

Для написания сервиса используется класс, в котором находятся статические методы. Каждый метод выполняет определенную бизнес-задачу. Для удобства используются статические методы, так как для их вызова не нужно создавать экземпляр самого класса.

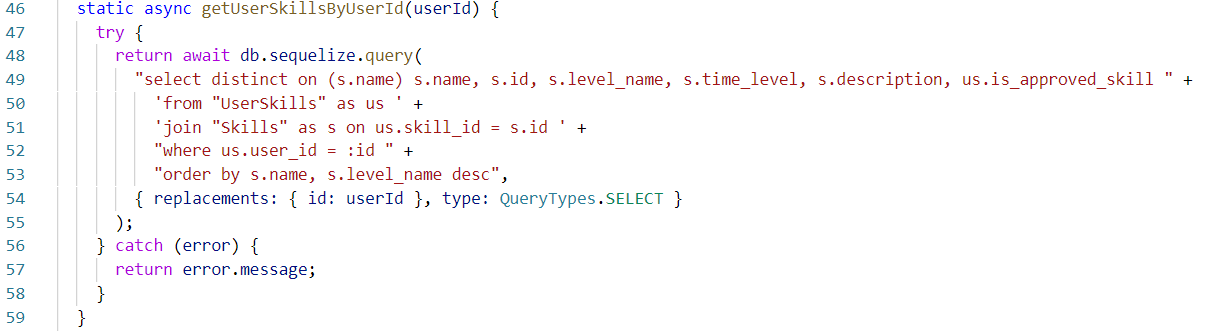
Для использования асинхронных функций используется конструкция «async-await», возвращающая промис. Таким образом, код становится легким в понимании и решается проблема «callback-hell».

В результате выполнения функции сервиса возвращается объект sequelize, соответствующий запрошенной сущности из базы данных.

Для запросов к базе данных используется объект «db», который содержит такие свойства, как объект sequelize и Sequelize, то есть инструменты для доступа к базе данных.

Sequelize позволяет использовать встроенные методы для доступа к данным, например, производить поиск сущности по значению заданного параметра или же по первичному ключу, обновление или удаление.

Также, есть возможность использовать прямой запрос к базе данных с помощью объекта «query», где применяется язык SQL для доступа к данным. То есть, можно использовать большой перечень функциональности, например, «left outer join», «inner join», «cross join», секции «distinct», «order by» и другие. Пример такого запроса приведен на листинге 3.5.



Листинг 3.5 – Пример запроса, использующего объект query

Также, стоит отметить, что слой сервисов тесно взаимодействует со слоем данных.

* + 1. **Реализация слоя контроллеров**

Над уровнем сервисов располагается уровень контроллеров. На нем находятся классы со статическими методами, параметрами которых являются объекты request и response. В случае успеха, в результате работы метода контроллера может возвращаться ответ с успешным кодом или же с кодом ошибки с указанием текста. Фрагмент контроллера продемонстрирован на листинге 3.6.



Листинг 3.6 – Фрагмент контроллера

Как видно на рисунке 3.6, в методе контроллера вызывается соответствующий метод уровня бизнес-логики, то есть, уровень контроллеров ничего не знает об уровне доступа к данным.

Также, параметры могут извлекаться как из тела запроса, так и из параметров. После доступа к соответствующему атрибуту происходит присваивание поля к соответствующей переменной.

* + 1. **Реализация слоя маршрутизаторов**

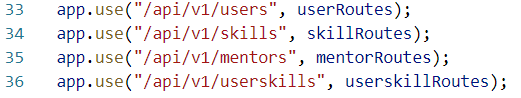
Для реализации слоя маршрутизаторов используется объект библиотеки express – Router. Его пример можно увидеть на листинге 3.7.



Листинг 3.7 – Пример маршрутизатора

Объект router позволяет обрабатывать методы http, например, такие как get, post, put и delete. Первый параметр – маршрут, который будет обрабатывать маршрутизатор, далее идут промежуточные обработчики и последним идет метод, который будет выполняться. На уровне роутеров используются методы, лежащие на уровень ниже, то есть, контроллеров.

Далее созданные роутеры подключаются к серверу, написанному на express. Фрагмент данной функциональности представлен на листинге 3.8.



Листинг 3.8 – Подключение роутеров к серверу

Первый параметр – это путь, а второй – его обработчик.

* + 1. **Реализация промежуточного слоя**

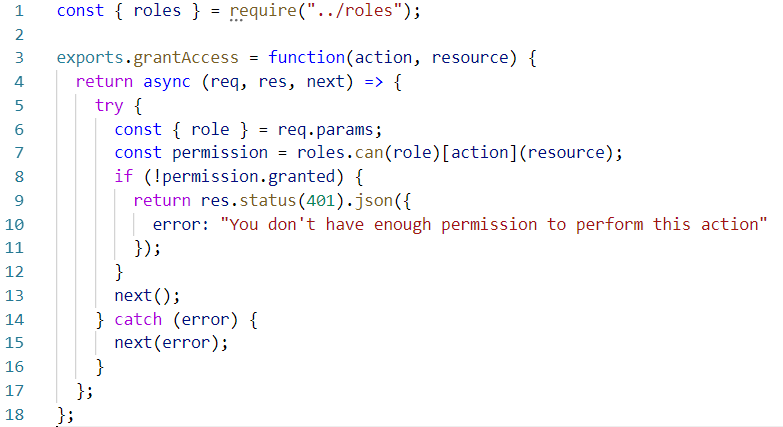
Промежуточный слой служит для фильтрации запросов, например, для ограничения доступа к ресурсу при отсутствии у пользователя прав на него.

В данном дипломном проекте был разработан слой, который контролирует доступ к ресурсам и позволяет перейти пользователю на страницу профиля только после успешной аутентификации. Код файла, который обеспечивает выполнение данной логики представлен на листинге 3.9.



Листинг 3.9 – Промежуточный слой для аутентификации

Также, для ограничения доступа к ресурсам была применена библиотека «access-control», исполнение которой представлено на листинге 3.10.



Листинг 3.10 – Промежуточный слой для ограничения доступа к ресурсам

* 1. **Разработка клиентской части приложения**

Для реализации клиентской части проекта был разработан проект, имеющий структуру, представленную на рисунке 3.1.

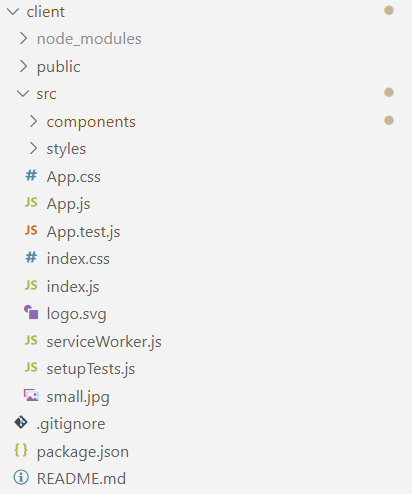


Рисунок 3.1 – Структура клиентского приложения

Все основные файлы содержатся в папке «components». Компонент – это элемент библиотеки React. Компоненты бывают классовые и функциональные. В данном проекте продемонстрированы функциональные компоненты. Компоненты можно создавать, а можно использовать готовые.

Папка «styles» содержит файлы стилей. Ее содержимое представлено на рисунке 3.12.

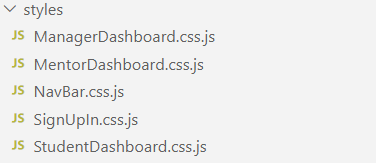
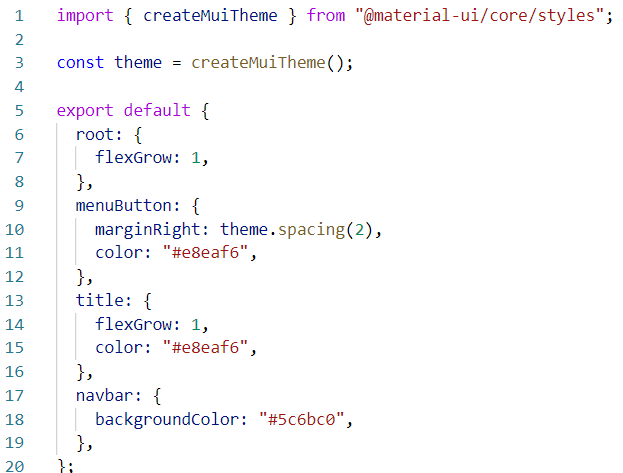


Рисунок 3.2 – Содержимое папки со стилями

Файлы, находящиеся в данной папке имею расширение «css.js».

Чтобы использовать стили в данных файлах, необходимо в первую очередь импортировать библиотеку «@material-ui/core/styles». Код файла со стилями представлен на листинге 3.11.



Листинг 3.11 – Пример файла со стилями

Также, можно прописывать стили в файлах с самими компонентами, но это сильно увеличивает количество кода и, следовательно, ухудшает его читаемость.

Все компоненты являются дочерними по отношению к главному компоненту в приложении – «App». Данный элемент создается по умолчанию в каждом новом проекте и может быть изменен в процессе написания кода.

Для обращения к серверу была использована библиотека «axios». Она является легковесной и предоставляет основные функции, с помощью которых можно отправить http-запросы, которые могут содержать заголовки или параметры. Пример запроса представлен на листинге 3.12.



Листинг 3.12 – Пример «axios» запроса

Первым параметром метода является url, а вторым – заголовки, если они есть.

Компонент «App» использует внутри себя элемент «Router», который отвечает за маршрутизацию. Все маршруты записываются с помощью компонента «Route». В атрибуте «component» содержится название компонента, который будет отображаться при переходе по маршруту, указанному в атрибуте «path». Фрагмент данного файла находится на листинге 3.13.



Листинг 3.13 – Фрагмент файла App.js

Для организации приложения, которое будет проще понять, необходимо максимально разделять компоненты и выделять мелкие части, которые затем включаются в составные.

* 1. **Вывод по разделу**

В данном разделе были описаны элементы, из которых состоит серверная и клиентская части разрабатываемого приложения.

Для серверной части была описана архитектура приложения. Также, была описана схема маршрутизации, применяемая в приложении.

Для клиентской части были описаны элементы, из которых она состоит: компоненты, события, обработчики событий. Также, была приведена и описана структура клиентской части приложения с возможностями, которые она предоставляет пользователю.

1. **Аа**
2. **Ббб**
3. **Технико-экономическое обоснование дипломного проекта**

ФИО

Подпись

Дата

Лист

1

БГТУ 06.00.ПЗ

Разраб.

*Лещенко А.О.*

Прове р.

*Сухорукова И.Г.*

Н. контр.

*Рыжанкова А.С.*

Утверд.

Пацей Н.В.

6 Технико-экономическое обоснование проекта

Лит.

Листов

8

*74417024, 2020*

У

* 1. **Общая характеристика разрабатываемого программного средства**

Во время разработки дипломного проекта использовалась технология Node.js, совместно с библиотекой React.js. Разработанное программное средство представляет собой веб-приложение для управления повышениями квалификаций сотрудников в рамках IT-компаний.

Данный раздел служит для определения затрат, произведенных на всех стадиях разработки программного средства.

* 1. **Исходные данные и маркетинговый анализ**

Источниками исходных данных для расчетов выступают действующие законы и нормативно-правовые акты. Исходные данные для расчета стоимости приведены в таблице 6.1.

Таблица 6.1 – Исходные данные для расчёта

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование показателя | Единица измерения | Условные обозначения | Норматив |
| Численность разработчиков | чел. | Чр | 1 |
| Норматив дополнительной заработной платы | % | Ндз | 15 |
| Ставка отчислений в Фонд социальной защиты населения | % | Нфсзн | 34 |
| Ставка отчислений в БРУСП «Белгосстрах» | % | Нбгс | 0,4 |
| Цена одного машино-часа | руб. | Смч | 0,06 |
| Норматив прочих затрат | % | Нпз | 17 |
| Норматив накладных расходов | % | Нобп,обх | 148 |
| Норматив расходов на сопровождение и адаптацию | % | Нрса | 17 |
| Ставка НДС | % | Нндс | 20 |

В ходе проведения маркетингового анализа, была выявлена стоимость разработки веб приложения для управление повышением навыков работников.

Средняя цена разработки серверной части приложения составляет 11 500 рублей, клиентской – 10 000 рублей. Общая стоимость разработки данного приложения составляет 21 500 рублей.

*Евлаш А.И.*

Консульт.

* 1. **Методика обоснования цены**

В современных рыночных экономических условиях программное средство выступает преимущественно в виде продукции организаций, представляющей собой функционально завершенное и имеющее товарный вид программное средство, реализуемые покупателям по рыночным отпускным ценам.

Разработка проектов программных средств требует затрат разнообразных и не редко значительных объемов ресурсов (трудовых, материальных, финансовых). В связи с этим, разработка и реализация каждого проекта обосновывается, как технически, так и экономически.

Проект стоит разрабатывать, если он дает определенные преимущества по сравнению с известными передовыми аналогами или, в крайнем случае, по сравнению с существующей практикой. Поэтому, до того, как приступить к разработке проекта программного средства, специалисты, используя соответствующие методы, находят наиболее рациональные программные решения, обеспечивающее высокий технический уровень программы и дающие существенную экономию ресурсов.

У разработчика экономический эффект выступает в виде чистой прибыли от реализации веб-приложения, остающейся в распоряжении организации, а у пользователя – в виде экономии трудовых, материальных и финансовых ресурсов, получаемой за счет:

* снижения трудоемкости расчетов и алгоритмизации программирования и отладки программ;
* снижения расходов на материалы;
* ускорение ввода в эксплуатацию новых систем;
* улучшения показателей основной деятельности в результате использования веб-приложения.

Стоимостная оценка веб-приложения у разработчиков предполагает определение затрат, что включает следующие статьи:

* заработная плата исполнителей – основная и дополнительная;
* отчисления в фонд социальной защиты населения;
* отчисления по обязательному страхованию от несчастных случаев на производстве и профессиональных заболеваний;
* расходы на материалы и комплектующие;
* расходы на оплату машинного времени;
* прочие прямые затраты;
* накладные расходы.

На основании затрат рассчитывается себестоимость и отпускная цена веб-приложения.

* + 1. **Определение объема программного средства**

Для оценки объема программного средства, все его функции классифицируются с использованием специального каталога функций, который определяет их объем.

Общий объем программного средства *V*o, вычисляется как сумма объемов *Vi* каждой из *n* его функций (формула 6.1).

 (6.1)

Для расчета общего объема программного средства в таблице 6.2 представлены функции, присутствующие в рассматриваемом программном средстве и соответствующий им объем в условных машино-командах (УМК).

Таблица 6.2 – Содержание и объем функций в программном средстве

|  |  |  |
| --- | --- | --- |
| Номер функции | Содержание функции | Объем, условных машино-команд |
| 101 | Организация ввода информации | 530 |
| 102 | Контроль, предварительная обработка | 350 |
| 111 | Управление вводом/выводом | 1300 |
| 202 | Взаимодействие между компонентами системы | 1630 |
| 401 | Взаимодействие с базой данных | 960 |
| 402 | Вспомогательные методы | 200 |
| 506 | Обработка ошибочных и сбойных ситуаций | 340 |
| 707 | Графический вывод результатов | 560 |

Опираясь на данные таблицы 6.2, можно определить объем программного средства, разработанного в ходе дипломного проектирования:

*Vо* = 530 + 350 + 1300 + 1630 + 960 + 200 + 340 + 560 = 5870 УМК

Уточненный объем программного средства *V*o*/* равен произведению объема программного средства *Vо* на коэффициент изменения скорости обработки информации Кск (формула 6.2).

|  |  |  |
| --- | --- | --- |
|  | *Vo/ = Vo ·* Кск | (6.2) |

Исходя из вычисленного объема программного средства, можно определить уточненный объем программного средства:

*Vo/* = 5870  0,6 = 3522 УМК

Исходя из вычисленного объема программного средства, был определен уточненный объем программного средства.

* + 1. **Основная заработная плата**

Для определения величины основной заработной платы, было проведено исследование величин заработных плат для специалистов в сфере разработки веб-приложенией на JavaScript. В итоге было установлено, что средняя месячная заработная плата на позиции junior/middle составляет 1000 рублей.

Проект разрабатывался одним разработчикам на протяжении трех месяцев. Таким образом, основная заработная плата будет рассчитываться по формуле 6.3:

|  |  |
| --- | --- |
| , | (6.3) |

где Соз – основная заработная плата, руб.;

Траз – время раработки, месяцев;

Краз – количество разработчиков, человек;

Сзп – средняя месячная заработная плата, руб.

руб.

После исследования велечин заработных плат в сфере разработки приложений, была рассчитана величина основной заработной платы.

* + 1. **Дополнительная заработная плата**

Дополнительная заработная плата на конкретное программное средство включает выплаты, предусмотренные законодательством о труде, и определяется по нормативу в процентах к основной заработной плате по формуле 6.4:

|  |  |
| --- | --- |
|  | (6.4) |

где Соз – основная заработная плата, руб.;

Ндз – норматив дополнительной заработной платы, %.

руб.

Изучив дополнительные выплаты, предусмотренные законодательством о труде, была рассчитана дополнительная заработная плата.

* + 1. **Отчисления в Фонд социальной защиты населения**

Отчисления в Фонд социальной защиты населения (ФСЗН) определяются в соответствии с действующими законодательными актами по нормативу в про-центном отношении к фонду основной и дополнительной зарплаты исполнителей и вычисляются по формуле 6.5:

|  |  |
| --- | --- |
|  | (6.5) |

где Соз – основная заработная плата, руб.;

Сдз – дополнительная заработная плата на конкретное ПС, руб.;

Нфсзн – норматив отчислений в Фонд социальной защиты населения, %.

Отчисления в БРУСП «Белгосстрах» вычисляются по нормативу в процентном отношении к фонду основной и дополнительной зарплаты исполнителей и вычисляются по формуле 6.6:

|  |  |
| --- | --- |
|  | (6.6) |

руб.

руб.

В соответствии с формулой и действующими законодательными актами, были рассчитаны отчисления в фонд социальной защиты населения и БРУСП «Белгосстрах».

* + 1. **Расходы на материалы**

Сумма расходов на материалы См определяется как произведение нормы расхода материалов в расчете на сто строк исходного кода Нм на уточненный объем программного средства *V*o*/* (формула 6.7).

|  |  |  |
| --- | --- | --- |
|  |  | (6.7) |

Учитывая, что норма расхода материалов в расчете на сто строк исходного кода равна 0,46 руб., можно определить сумму расходов на материалы:

См = 0,46 ⋅ 3522 / 100 = 16,2 руб.

Сумма расходов на материалы была вычислена на основе данных, приведенных в таблице 6.1 данного дипломного проекта.

* + 1. **Расходы на оплату машинного времени**

Сумма расходов на оплату машинного времени Смв определяется как произведение стоимости одного машино-часа Смч на уточненный объем программного средства *Vo/* и на норматив расхода машинного времени на отладку ста строк исходного кода Нмв (формула 6.8).

|  |  |
| --- | --- |
|  | (6.8) |

Учитывая, что норматив машинного времени на отладку ста строк исходного кода равен 15, можно определить сумму расходов на оплату машинного времени:

Смв = 0,06  3522  15 / 100 = 31,7 руб.

Изучив норматив машинного времени на отладку, были рассчитаны расходы на оплату машинного времени.

* + 1. **Прочие прямые затраты**

Сумма прочих затрат Спз определяется как произведение основной заработной платы исполнителей на конкретное программное средство Соз на норматив прочих затрат в целом по организации Нпз. К прочим прямым затратам предприятия относятся: расходы на командировки, расходы на аудиторские, юридические, консультационные и информационные услуги, расходы по набору работников, расходы на подготовку и переподготовку кадров.

Для рассчета прочих прямых затрат необходимо использовать формулу 6.9.

|  |  |
| --- | --- |
|  | (6.9) |

Спз = 3000 ⋅ 17 / 100 = 510 руб.

Используя полученные ранее основные заработные платы и норматив прочих затрат, были рассчитаны прочие прямые затраты.

* + 1. **Накладные расходы**

Сумма накладныхрасходовСобп,обх – произведение основной заработной платы исполнителей на конкретное программное средство Соз на норматив накладных расходов в целом по организации Нобп,обх.

|  |  |
| --- | --- |
| . | (6.10) |

Теперь можно определить сумму накладных расходов:

Собп,обх = 3000 × 148 / 100 = 4440  руб.

Используя полученные ранее основные заработные платы и норматив накладных расходов, была рассчитана сумма накладных расходов.

* + 1. **Сумма расходов на разработку программного средства**

Сумма расходов на разработку программного средства Ср определяется как сумма основной и дополнительной заработных плат исполнителей на конкретное программное средство, отчислений на социальные нужды, расходов на материалы, расходов на оплату машинного времени, суммы прочих затрат и суммы накладных расходов.

|  |  |
| --- | --- |
| Ср = Соз + Сдз + Сфсзн + Сбгс + См + Смв + Спз + Собп,обх | (6.11) |

Ср = 3000 + 450 + 1173 + 13,8 + 16,2 + 31,7 + 510 + 4440 = 9634,7 руб.

Сумма расходов на разработку программного средства была вычислена на основе данных, рассчитанных ранее в данном разделе.

* + 1. **Расходы на сопровождение и адаптацию**

Сумма расходов на сопровождение и адаптацию программного средства Срса определяется как произведение суммы расходов на разработки на норматив расходов на сопровождение и адаптацию Нрса.

|  |  |  |
| --- | --- | --- |
|  | . | (6.12) |

Срса = 9634,7⋅ 17 / 100 = 1637,9 руб.

Сумма расходов на сопровождение и адаптацию была вычислена на основе данных, рассчитанных ранее в раном разделе.

Все проведенные выше расчеты необходимы для вычисления полной себестоимости проекта.

* + 1. **Полная себестоимость**

Полная себестоимость Сп определяется как сумма двух элементов: суммы расходов на разработку Ср и суммы расходов на сопровождение и адаптацию программного средства Срса.

|  |  |  |
| --- | --- | --- |
|  | . | (6.13) |

Сп = 9634,7 + 1637,9 = 11 272,6 руб.

Полная себестоимость программного средства была вычислена на основе данных, рассчитанных ранее в данном разделе.

* + 1. **Определение цены, оценка эффективности**

При расчете прибыли будем опираться на среднюю рыночную цену, по которой можно разработать данное программное обеспечение. Средняя рыночная цена составляет 21 500 руб.

Прибыль рассчитывается по формуле 6.14:

|  |  |  |
| --- | --- | --- |
|  | , | (6.14) |

где Ппс – прибыль от реализации программного средства, руб.;

Цр – средняя рыночная цена продукта, руб.;

Сп – полная себестоимость программного средства, руб.;

Ппс = (21 500/1,2) – 11 272,6 = 6644,01 (руб.).

Уровень рентабельности разработанного программного средства определяется по формуле 6.15:

|  |  |
| --- | --- |
| , | (6.15) |

где Урент – уровень рентабельности программного средства, %

Сп – полная себестоимость программного средства, руб.;

Ппс – прибыль от реализации программного средства, руб.

Урент = (6644,01 / 11 272,6) ⋅ 100 = 58,94 %

Рентабельность разработки программного средства была вычислена на основе данных, расчитанных ранее в данном разделе.

* 1. **Вывод по разделу**

В таблице 6.3 представлены результаты расчетов для основных показателей данной главы в краткой форме.

Таблица 6.3 – Результаты расчетов

|  |  |
| --- | --- |
| Наименование показателя | Значение |
| Время разработки, мес. | 3 |
| Количество программистов, чел. | 1 |
| Зарплата с отчислениями, руб. | 4636,8 |
| Расходы на материалы, оплату машинного времени, прочие, руб | 557,9 |
| Накладные расходы, руб | 4440 |
| Себестоимость разработки программного средства, руб. | 9634,7 |
| Расходы на сопровождение и адаптацию, руб. | 1637,9 |
| Полная себестоимость, руб. | 11 272,6 |
| Цена аналога, руб. | 21 000 |

Разработка программного средства, осуществляемая одним программистом в течение трех месяцев при заданных условиях обойдется компании в 11 272,6 руб. Реализации данного веб-приложения при среднерыночной цене в 21 000 руб., принесет прибыль в сумме 6644,01 рублей. Уровень рентабельности составит 58,94%.

Результаты проведенных расчетов показывают, что веб-приложение для управления повышениями квалификаций сотрудников является рентабельным и экономически обоснованным программным продуктом, а отпускная цена, является приемлемой для потребителя.

**Заключение**

ФИО

Подпись

Дата

Лист

1

БГТУ 00.00.ПЗ

Разраб.

*Лещенко А.О.*

Провер.

*Сухорукова И.Г.*

Н. контр.

*Рыжанкова А.С.*

Утверд.

Пацей Н.В.

Заключение

Лит.

Листов

1

*74417024, 2020*

У

**Список использованнх источников**

ФИО

Подпись

Дата

Лист

1

БГТУ 00.00.ПЗ

Разраб.

*Лещенко А.О.*

Провер.

*Сухорукова И.Г.*

Н. контр.

*Рыжанкова А.С.*

Утверд.

Пацей Н.В.

Список использованных источников

Лит.

Листов

1

*74417024, 2020*

У

1. Основы JavaScript [Электронный ресурс] – Режим доступа: https://developer.mozilla.org/ru/docs/Learn/Getting\_started\_with\_the\_web/JavaScript\_basics. – Дата доступа: 06.05.2020.
2. JavaScript [Электронный ресурс] – Режим доступа: https://developer.mozilla.org/ru/docs/Web/JavaScript. – Дата доступа: 06.05.2020.
3. Основы React [Электронный ресурс] – Режим доступа: https://habr.com/ru/company/ruvds/blog/343022/. – Дата доступа: 07.05.2020.
4. ECMAScript [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/ECMAScript. – Дата доступа: 07.05.2020.
5. JavaScript brief history and ECMAScript [Электронный ресурс] – Режим доступа: https://medium.com/@madasamy/javascript-brief-history-and-ecmascript-es6-es7-es8-features-673973394df4. – Дата доступа: 07.05.2020.
6. Все, что вам нужно знать о Node.js [Электронный ресурс] – Режим доступа: https://habr.com/ru/post/460661/. – Дата доступа: 08.05.2020.
7. Express/Node introduction [Электронный ресурс] – Режим доступа: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express\_Nodejs. – Дата доступа: 08.05.2020.
8. PostgreSQL [Электронный ресурс] – Режим доступа: https://habr.com/ru/post/168601/. – Дата доступа: 10.05.2020.
9. Use Case Diagram Tutorial [Электронный ресурс] – Режим доступа: https://online.visual-paradigm.com/diagrams/tutorials/use-case-diagram-tutorial/. – Дата доступа: 10.05.2020.
10. Элементы графической нотации диаграммы вариантов использования [Электронный ресурс] – Режим доступа: https://www.intuit.ru/studies/courses/32/32/lecture/1004. – Дата доступа: 10.05.2020.
11. Пять простых шагов для понимания JSON Web Tokens (JWT) [Электронный ресурс] – Режим доступа: https://habr.com/ru/post/340146/. – Дата доступа: 10.05.2020.
12. Многоуровневая архитектура [Электронный ресурс] – Режим доступа: https://en.wikipedia.org/wiki/Multitier\_architecture. – Дата доступа: 10.05.2020.
13. React и SEO: преимущества изоморфности React для одностраничных приложений [Электронный ресурс] – Режим доступа: https://xbsoftware.ru/blog/react-seo-izomorphnost-react-odnostrannoe-prilozhenie/. – Дата доступа: 10.05.2020.