

Реферат

Пояснительная записка дипломного проекта содержит 90 страниц, 46 иллюстраций, 16 таблиц, 20 использованных источников, 6 приложений.

JAVASCRIPT, TYPESCRIPT, ANGULAR, GOLANG, ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ, СУБД POSTGRESQL, ПРОЕКТИРОВАНИЕ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА, СЕБЕСТОИМОСТЬ ПРИЛОЖЕНИЯ

Цель дипломного проекта – разработка интернет-сервиса для учета и контроля выполнения дипломного проектирования в вузе.

В результате решения поставленных задач, был разработан интернет-сервис, повышающий эффективность работы заведующего кафедрой и руководителей дипломных проектов.

В первой главе проведен аналитический обзор предметной области по теме дипломного проекта, поставлены задачи для достижения цели, а также описаны основные технические требования к проекту.

Вторая глава посвящена процессу проектирования интернет-сервиса. В ней описана архитектура интернет-сервиса, а также база данных.

В третьей главе описан процесс разработки и программной реализации приложения.

В четвертой главе приведено описание процесса тестирования созданного интернет-сервиса.

В пятой главе приводится руководство программиста.

В шестой главе дипломного проекта приводится расчет экономических параметров и себестоимости данного приложения.

В заключении приведены результаты проделанной работы.

Объем графической части в пересчете на формат А1 – один лист.

				БГТУ 00.00.ПЗ						
	ФИО	Подпись	Дата							
Разраб.	Криштопчик А.Ю.			Реферат			Лит.		Лист	Листов
Пров.	Смелов В.В.								1	1
Консульт.							74417011, 2020			
Н. контр.	Рыжанкова А.С.									
Утв.	Смелов В.В.									

Abstract

The explanatory note of the diploma project contains 90 pages, 46 illustrations, 16 tables, 20 sources used, 6 appendices.

JAVASCRIPT, TYPESCRIPT, ANGULAR, GOLANG, DATABASE DESIGN, DATABASE MANAGEMENT SYSTEM POSTGRESQL, DESIGNING A USER INTERFACE, COST PRICE.

The purpose of the graduation project is an internet service development for accounting and monitoring the implementation of perform diploma projects at the university.

As a result of the solution of the tasks, they were developed an Internet service that increases the efficiency of the department head and thesis project managers.

The first chapter provides an analytical overview of the subject area for the diploma projects, setting tasks for achieving goals, also defining the basic technical requirements for the project.

The second chapter is devoted to the process of designing an Internet service. It describes the architecture of the Internet service, and the database.

The third chapter describes the development process and software implementation of the application.

The fourth chapter describes the testing process of the created Internet service.

The fifth chapter provides a programmer's guide.

The sixth chapter of the diploma project provides a calculation of economic parameters and the cost of this application.

In conclusion, the results of the work done are presented.

The volume of the graphic part in terms of the A1 format is one sheet.

				БГТУ 00.00.ПЗ						
	ФИО	Подпись	Дата							
Разраб.	Криштопчик А.Ю.			Abstract			Лит.	Лист	Листов	
Пров.	Смелов В.В.								1	1
Консульт.							74417011, 2020			
Н. контр.	Рыжанкова А.С.									
Утв.	Смелов В.В.									

Содержание

Введение	7
1 Постановка задачи и обзор аналогичных решений	8
1.1 Постановка задачи	8
1.2 Обзор аналогичных решений.....	8
1.3 Патентный поиск по теме дипломного проекта	10
1.4 Выбор программной платформы и инструментов для разработки	11
1.5 Вывод по разделу	15
2 Проектирование интернет-сервиса	16
2.1 Функциональное наполнение интернет-сервиса	16
2.2 Архитектура интернет-сервиса	17
2.3 База данных интернет-сервиса	19
2.4 Вывод по разделу	22
3 Разработка интернет-сервиса.....	23
3.1 Разработка серверной части интернет-сервиса	23
3.2 Разработка клиентской части интернет-сервиса	26
3.3 Разработка базы данных интернет-сервиса.....	28
3.4 Вывод по разделу	30
4 Тестирование интернет-сервиса	31
4.1 Просмотреть списки	31
4.2 Добавить в список.....	33
4.3 Удалить из списков.....	34
4.4 Изменить данные в списках.....	36
4.5 Фильтровать списки	37
4.6 Вывести в текстовый формат	38
4.7 Вывод по разделу	39
5 Руководство программиста.....	40
5.1 Установка серверной части интернет-сервиса	40
5.2 Установка клиентской части интернет-сервиса	42
5.3 Установка базы данных интернет-сервиса.....	45
5.4 Вывод по разделу	47
6 Экономический раздел	48
6.1 Общая характеристика разрабатываемого программного средства.....	48
6.2 Исходные данные и маркетинговый анализ	48
6.3 Методика обоснования цены	49
6.3.1 Объем программного средства.....	50
6.3.2 Основная заработная плата.....	51
6.3.3 Дополнительная заработная плата	51

				БГТУ 00.00.ПЗ			
	ФИО	Подпись	Дата	Содержание			
Разраб.	Криштопчик А.Ю.						
Пров.	Смелов В.В.						
Консульт.							
Н. контр.	Рыжанкова А.С.						
Утв.	Смелов В.В.			74417011, 2020			
					Лит.	Лист	Листов
					1	2	

6.3.4 Отчисления в Фонд социальной защиты населения	52
6.3.5 Расходы на материалы.....	52
6.3.6 Расходы на оплату машинного времени.....	53
6.3.7 Прочие прямые затраты	53
6.3.8 Накладные расходы	53
6.3.9 Сумма расходов на разработку программного средства	53
6.3.10 Расходы на сопровождение и адаптацию	54
6.3.11 Полная себестоимость	54
6.4 Выводы по разделу	54
Заключение	56
Список использованных источников	57
Приложение А Диаграмма вариантов использования	59
Приложение Б Общая схема взаимодействия компонентов	60
Приложение В Логическая схема базы данных.....	61
Приложение Г Листинги кода серверной части	62
Приложение Д Листинги кода клиентской части.....	75
Приложение Е Таблица экономических показателей	90

Введение

На кафедре информационных систем и технологий Белорусского государственного технологического университета ежегодно защищают дипломные проекты и работы около 100 выпускников. Контроль за ходом дипломного проектирования представляет собой отдельную непростую задачу, требующую назначения темы, распределение студентов руководителям дипломного проектирования, назначение руководителей, комиссии, нормоконтролеров, председателей дипломной комиссии, рецензентов, а также мониторинг процесса выполнения работы студентом.

Целью дипломного проекта является повышение эффективности работы заведующего кафедрой и руководителей дипломных проектов.

Для достижения цели сформированы следующие задачи:

- исследовать преимущества и недостатки аналогичных решений;
- разработать функционал интернет-сервиса;
- разработать прототип интернет-сервиса на основе интерфейс;
- разработать структуру базы данных;
- разработать архитектуру программной реализации интернет-сервиса.
- разработать интернет-сервис для учета и контроля выполнения дипломного проектирования в вузе.

Целевой аудиторией данного дипломного проекта являются заведующие кафедрой и руководители дипломных проектов.

При разработке данного приложения был произведен обзор аналогичных продуктов с целью устранения слабых и добавления сильных сторон приложений схожей тематики в дипломный проект. Структура приложения спроектирована таким образом, чтобы позволить использовать интернет-сервис для различных кафедр университета.

Для реализации данного интернет-сервиса используется база данных *PostgreSQL* [1], фреймворк *Angular* [2] для разработки динамических веб-приложений, а также для разработки серверной части язык программирования *Golang* [3]. Выбор фреймворка для разработки веб-приложения пал в пользу *Angular* по причине того, что он позволяет создавать быстродействующие, легко оптимизируемые приложения. Он также предоставляет возможность широкой настройки компонентов. Также фреймворк обзавелся довольно большим сообществом разработчиков.

База данных *PostgreSQL* была выбрана, так как она является объектно-реляционной, поддерживает пользовательские объекты и их поведение, включая типы данных, функции, операции, домены и индексы. Это делает *PostgreSQL* невероятно гибкой и надежной.

				БГТУ 00.00.ПЗ						
	ФИО	Подпись	Дата							
Разраб.	Криштопчик А.Ю.			Введение			Лит.	Лист	Листов	
Пров.	Смелов В.В.								1	1
Консульт.							74417011, 2020			
Н. контр.	Рыжанкова А.С.									
Утв.	Смелов В.В.									

1 Постановка задачи и обзор аналогичных решений

1.1 Постановка задачи

Целью дипломного проекта является повышение эффективности работы заведующего кафедрой и руководителей дипломных проектов путем автоматизации контроля за ходом дипломного проектирования, представляющего собой отдельную не простую задачу, требующую назначения темы, распределение студентов руководителям дипломных работ, назначение руководителей, комиссии, нормоконтролеров, председателей дипломной комиссии, рецензентов, а также мониторинг процесса выполнения работы студентом.

Для достижения цели были поставлены следующие задачи:

- исследовать преимущества и недостатки аналогичных решений;
- разработать функционала интернет-сервиса;
- разработать прототип интернет-сервиса на основе интерфейс;
- разработать структуру базы данных;
- разработать архитектуру программной реализации интернет-сервиса.
- разработать интернет-сервис для учета и контроля выполнения дипломного проектирования в вузе.

1.2 Обзор аналогичных решений

Для того, чтобы создать интернет-сервис, который соответствовал бы современным стандартам, выглядел согласно новым тенденциям в области дизайна, а также имел исчерпывающий набор функционала, был изучен ИТ-рынок и проведен небольшой обзор аналогов разрабатываемого интернет-сервиса.

В этой главе описаны некоторые существующие аналоги приложения для автоматизации учебного процесса в вузе, а также были поставлены цели и задачи дипломного проекта.

USU [4] – это специализированное программное обеспечение, действие которого направлено на оптимизацию всего устройства. Автоматизация контроля учебного процесса возьмет на себя все прежде контролируемые единицы организации, станет напоминать об истекающих товарах, необходимых для обучения. Контроль результативности проводимых занятий и их посещаемость. Возможность составления расписания занятий внутри ПО, позволит составить его правильно, в соответствии с рациональным и последовательным использованием аудиторий.

Автоматизация учебного процесса *usi* подходит как для небольших образовательных отделов, мини-центров, обучающих дошколят, курсов по английскому, математике, физике и других интересных предметов, так и для ВУЗов, колледжей, лицеев и самих школ. Управление внутри системы осуществляет администратор,

				БГТУ 01.00.ПЗ			
	ФИО	Подпись	Дата				
Разраб.	Криштопчик А.Ю.			1 Постановка задачи и обзор аналогичных решений	Лит.	Лист	Листов
Пров.	Смелов В.В.					1	8
Консульт.					74417011, 2020		
Н. контр.	Рыжанкова А.С.						
Утв.	Смелов В.В.						

именно он распределяет обязанности и полномочия внутри софта Автоматизации. И может ограничивать доступ к некоторой информации для определенных подчиненных.

Интерфейс программы максимально прост и представлен на рисунке 1.1.

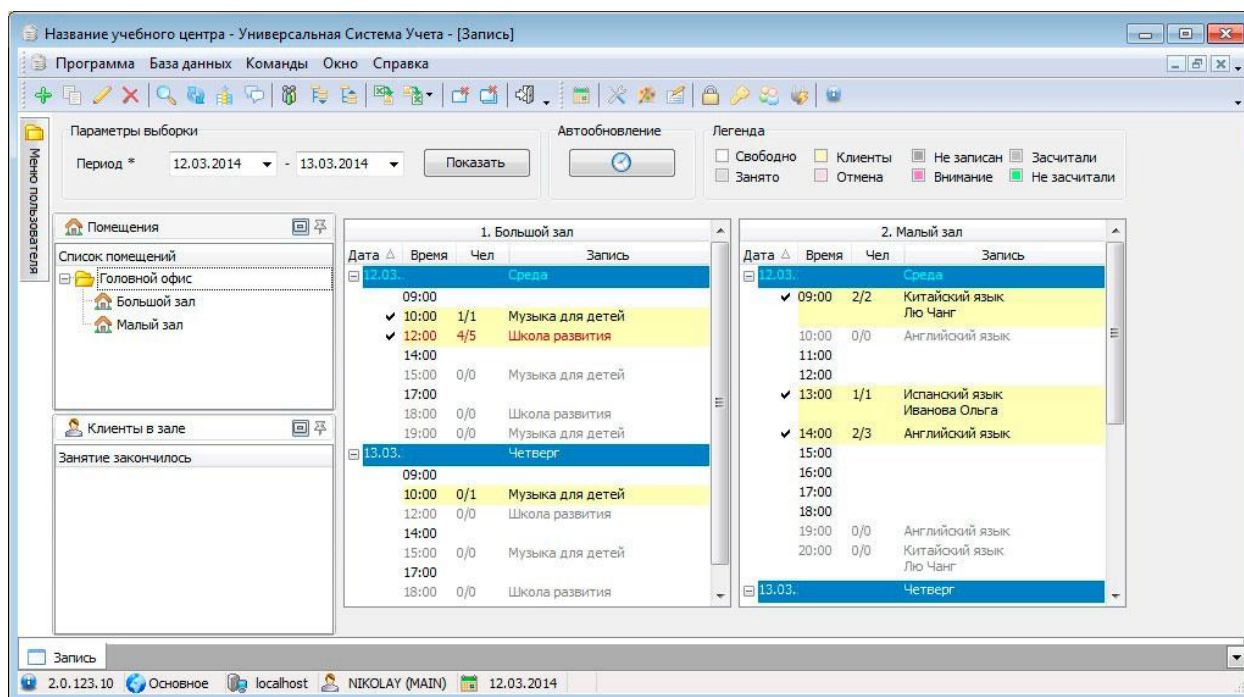


Рисунок 1.1 – Обзор аналога usu

К преимуществам данного приложения можно отнести его функционал и поддержку нескольких языков, а к его недостаткам: высокая цена, устаревший дизайн и то, что приложение является десктопным.

GS-Ведомости: Online [5] - это *web* система, которая позволяет работать с базой данных, установленной на сервере в образовательном учреждении, через сеть Интернет. Модуль работает на HTTP-сервере *Apache*. Пользователи веб-интерфейса системы подразделяются на следующие категории: администратор, преподаватель, студент, родители студента, абитуриент. Права доступа каждой категории пользователей настраиваются в модуле «Администратор» системы «GS-Ведомости».

В число реализованных на данный момент онлайн-модулей входят:

- модуль «*Online: Абитуриент*». Основной функционал онлайн-модуля заключается в возможности для абитуриентов подавать заявление на поступление на сайте образовательного учреждения. Заполненное заявление автоматически попадает в базу данных системы «GS-Ведомости». Данные о ходе приемной кампании, а также поля анкеты абитуриента можно настраивать исходя из потребностей и задач образовательного учреждения;

- модуль «*Online: Контингент учащихся*». С помощью веб-интерфейса можно отслеживать данные об успеваемости и посещаемости студентов, итоги сессий, выполненные контрольные работы, лицевые счета студентов, обучающихся на платной основе. В данный модуль также входит онлайн-интерфейс расширения «Журнал успеваемости»;

– модуль «*Online: Расписание занятий*» отображает расписание занятий в разрезе группы студентов. Информацию о расписании занятий через *web*-интерфейс могут получать пользователи категории «студент» и «родитель студента». Пользователи категории «преподаватель» могут смотреть расписание своих занятий во всех группах;

– модуль «*Online: Учебные планы*». Отображает учебный план по специальности для пользователя категории «студент». Если войти в модуль «*Online: Учебные планы*» под логином преподавателя, отобразится перечень учебных планов по группам соответственно распределенной учебной нагрузке;

– модуль «Блокнот» можно использовать в качестве планировщика и дневника;

Интерфейс, вышерассмотренного аналога, представлен на рисунке 1.2

GS-Ведомости Online Суперпользователь ТГУ

Главная Модули Настройки Администрирование Справка [Выйти](#)

Расписание

Группа:

Преподаватель:

Дата:

<<< На неделю назад На текущую неделю На неделю вперед >>>

Формат таблицы:

	01.10.2012 Пн	02.10.2012 Вт	03.10.2012 Ср	04.10.2012 Чт	05.10.2012 Пт	06.10.2012 Сб	07.10.2012 Вс
1 пара 09:00 - 10:20	Физическая культура	История политических и правовых учений	Криминалистика	Международное право	Экологическое право		
2 пара 10:30 - 11:50	Экологическое право	Земельное право		Прокурорский надзор	Российское предпринимательское право		
3 пара 12:20 - 13:40		Судебная психиатрия	Международное частное право				
4 пара 14:00 - 15:20	Право социального обеспечения						
5 пара							

Рисунок 1.2 – Обзор аналога GS-Ведомости: Online

К достоинствам приложения можно отнести его функционал и доступ через интернет. К недостаткам – высокую цену, сложность в обучении и устаревший дизайн.

1.3 Патентный поиск по теме дипломного проекта

В соответствии с темой дипломного проекта был проведен патентный поиск в области программных систем для автоматизации учебного процесса. Для проведения патентного поиска был использован «Федеральный институт промышленной собственности» который позволяет провести поиск патентов программ с 2013 года.

В результате патентного поиска было найдено два патента. Предметом поиска (объектом исследования или его составной части) была указана «автоматизация учебного процесса».

Первым найденным патентом является система автоматизации учебного процесса «Автор-ВУЗ» за номером *RU2016611794* от 10.02.2016, заявителем которого являются Губенко Игорь Олегович, Меренков Антон Сергеевич и Тареев Денис Сергеевич. Программа предназначена для автоматизации действий работников образовательных организаций. Программа обеспечивает выполнение следующих функций: формирование штатного приказа, анализ наличия вакантных ставок на кафедрах; создание учебных планов, анализ их на соответствие стандартам, выгрузка для отправки на экспертизу; формирование в программе методических материалов, анализ обеспеченности материалами дисциплин, анализ качества их подготовки; составление расчета разных видов нагрузки, учет фактически выполненной нагрузки, контроль выполнения норм нагрузки; составление расписаний учебных занятий, возможность выгрузки занятости преподавателей в календари; выполнение функций электронной информационно-образовательной среды. Для реализации данного программного обеспечения использовались такие языки программирования как *PHP* [6] и *JavaScript* [7].

Вторым найденным патентом является автоматизированная информационная система «Управление учебным процессом» за номером *RU2019664129* от 30.10.2019. Программа предназначена для автоматизации деятельности институтов повышения квалификации и учреждений дополнительного образования. Область применения: управление учебным процессом. Функциональные возможности: ведение базы данных слушателей, прием заявок слушателей на курсы и мероприятия через личный кабинет в Интернете, формирование групп слушателей и учебно-тематических планов, составление расписания, планирование занятости аудиторий с учетом имеющегося оборудования, ведение межкурсовой подготовки, учет преподавательского состава и их часовой нагрузки. Программа не содержит персональные данные. Тип ЭВМ: *IBM PC*-совместимый ПК, ОС: *Windows XP/7/8/8.1/10*. Для реализации данной информационной системы использовались такие языки программирования как *C#* [8], *HTML* [9], *CSS* [10] и *JavaScript*.

1.4 Выбор программной платформы и инструментов для разработки

Основной операционной системой для приложения была выбрана *Windows Server 2016* [11]. Разработка веб-интерфейса выполняется в среде *WebStorm* [12] с помощью платформы *Angular*. Языком программирования для написания кода является *JavaScript*. Для хранения и предоставления данных системы разработана база данных. В данном случае выбрана реляционная база данных *PostgreSQL*. Для связи веб-приложения с базой данных является среда выполнения *Golang*.

Основные инструменты, технологии, библиотеки и их назначение приведены в таблице 1.1.

Таблица 1.1 – Инструменты, технологии, библиотеки

Тип	Наименование	Назначение
Среда разработки	WebStorm	Написание в данной среде пользовательской части интернет-сервиса
Среда разработки	GoLand	Написание в данной среде серверной части интернет-сервиса
Среда разработки	DataGrip	Написание в данной среде базы данных для интернет-сервиса
Язык	TypeScript	Язык для написания пользовательской части веб-приложения
Язык	Golang	Язык для написания сервисов для веб-приложения
Пакет	Gorilla	Пакет разработчика специально для упрощения создания веб-приложений на языке Go
Фреймворк	Angular	Фреймворк для упрощения и упорядочения пользовательской части веб приложения
Библиотека	AngularMaterial	Библиотека компонентов для клиентской части веб-приложения
Метаязык	SASS	Метаязык на основе CSS, предназначенный для увеличения уровня абстракции CSS кода и упрощения файлов каскадных таблиц стилей

WebStorm – это одна из самых популярных сред разработки для *JavaScript*, разработанная на основе платформы *IntelliJ IDEA*.

WebStorm обеспечивает автодополнение, анализ кода на лету, навигацию по коду, рефакторинг, отладку, и интеграцию с системами управления версиями. Важным преимуществом интегрированной среды разработки *WebStorm* является работа с проектами (в том числе, рефакторинг кода *TypeScript* [13], находящегося в разных файлах и папках проекта, а также вложенного в *HTML*). Поддерживается множественная вложенность (когда в документ на *HTML* вложен скрипт на *JavaScript*, в который вложен другой код *HTML*, внутри которого вложен *Javascript*) – то есть в таких конструкциях поддерживается корректный рефакторинг. Данная среда разработки доступна для *Windows*, *OS X* и *Linux*.

Goland – это одна из самых популярных сред разработки для *Golang*, разработанная на основе платформы *IntelliJ IDEA*.

DataGrid – это одна из самых популярных сред разработки для *SQL*, разработанная на основе платформы *IntelliJ IDEA*.

TypeScript – язык программирования, представленный *Microsoft* в 2012 году и позиционируемый как средство разработки веб-приложений, расширяющее возможности *JavaScript*.

Разработчиком языка *TypeScript* является Андерс Хейлсберг, создавший ранее *Turbo Pascal*, *Delphi* и *C#*.

TypeScript является обратно совместимым с *JavaScript* и компилируется в последний. Фактически, после компиляции программу на *TypeScript* можно выполнять в любом современном браузере или использовать совместно с серверной платформой *Node.js*. Код экспериментального компилятора, транслирующего *TypeScript* в *JavaScript*, распространяется под лицензией *Apache*. Его разработка ведется в публичном репозитории через сервис *GitHub*.

TypeScript отличается от *JavaScript* возможностью явного статического назначения типов, поддержкой использования полноценных классов (как в традиционных объектно-ориентированных языках), а также поддержкой подключения модулей, что призвано повысить скорость разработки, облегчить читаемость, рефакторинг и повторное использование кода, помочь осуществлять поиск ошибок на этапе разработки и компиляции, и, возможно, ускорить выполнение программ.

Планируется, что в силу полной обратной совместимости адаптация существующих приложений на новый язык программирования может происходить поэтапно, путем постепенного определения типов.

Разработчики *TypeScript* искали решение, которое не будет нарушать совместимость со стандартом и его кроссплатформенной поддержкой. Зная, что только стандарт *ECMAScript* предлагает поддержку в будущем для программирования на базе классов (*Class-based programming*), *TypeScript* был основан на этом предположении. Это привело к созданию компилятора *JavaScript* с набором синтаксических языковых расширений, увеличенным на основе предложения, которое трансформирует расширения в *JavaScript*. В этом смысле *TypeScript* является представлением того, что ожидать от *ECMAScript* 6. Уникальный аспект не в предложении, а в добавлении в *TypeScript* статической типизации, что позволяет статически анализировать язык, облегчая оснастки и *IDE* поддержку.

Angular – это прогрессивный фреймворк для создания пользовательских интерфейсов. В отличие от фреймворков-монолитов, *Angular* 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, *Angular* полностью подходит и для создания сложных одностраничных приложений (*SPA*, *Single-Page Applications*), если использовать его совместно с современными инструментами и дополнительными библиотеками.

Важной концепцией *Angular* являются компоненты. Эта абстракция позволяет собирать большие приложения из маленьких «кусочков». Они представляют собой пригодные к повторному использованию объекты. Если подумать, почти любой интерфейс можно представить, как дерево компонентов.

Angular Material [14] – библиотека компонентов для *Angular*, которая является интеграция между спецификациями *Angular* и *Material Design*. Ее можно легко настроить для удовлетворения всех потребностей с помощью простого *API*.

Преимущества *Angular Material*:

- компоненты интернационализированы и доступны, чтобы все пользователи могли их использовать;
- простые *API*, в которых невозможно запутаться;

- поведение хорошо протестировано как с модульными, так и с интеграционными тестами;
- настраивается в пределах спецификации *Material Design*;
- сведение к минимуму эксплуатационных затрат;
- код является чистым и хорошо документированным, чтобы служить хорошим примером для *Angular* разработчиков;
- поддержка большинства браузеров.

Sass [15] это расширение *CSS*, которое придает мощи и элегантности этому простому языку. *Sass* даст вам возможность использовать переменные, вложенные правила, миксины, инлайновые импорты и многое другое, все с полностью совместимым с *CSS* синтаксисом. *Sass* помогает сохранять огромные таблицы стилей хорошо организованными, а небольшим стилям работать быстро, особенно с помощью библиотеки стилей *Compass*.

Для *Sass* доступно два синтаксиса. Первый, известный как *SCSS* (*Sassy CSS*) и используемый повсюду в этой статье — это расширенный синтаксис *CSS*. Это означает, что каждая валидная таблица стилей *CSS* это валидный *SCSS* файл, несущий в себе ту же самую логику. Более того, *SCSS* понимает большинство хаков в *CSS* и вендорные синтаксисы, например, такой как синтаксис фильтра в старом *IE*. Этот синтаксис улучшен *Sass* функционалом, описанным ниже. Файлы использующие этот синтаксис имеют *.scss* расширение.

Второй и более старый синтаксис, также известный как краеный синтаксис или иногда просто *Sass*, дает более сжатую возможность работы с *CSS*. Он использует отступы вместо скобок, что отделить вложения селекторов и новые строки вместо точек с запятой для разделения свойств. Иногда люди находят такой способ проще для понимания и быстрее для написания, чем *SCSS*. По факту, такой синтаксис имеет тот же функционал, хотя некоторые из них имеют слегка другой подход. Файлы используемые этот синтаксис имеют расширение *.sass*.

Любой синтаксис может импортировать файлы, написанные в другом. Файлы могут быть автоматически сконвертированы из одного в другой, используя *sass-convert* команду.

Golang — компилируемый многопоточный язык программирования, разработанный внутри компании *Google*. Разработка *Golang* началась в сентябре 2007 года, его непосредственным проектированием занимались Роберт Гризмер, Роб Пайк и Кен Томпсон, занимавшиеся до этого проектом разработки операционной системы *Inferno*. Официально язык был представлен в ноябре 2009 года. На данный момент поддержка официального компилятора, разрабатываемого создателями языка, осуществляется для операционных систем *FreeBSD*, *OpenBSD*, *Linux*, *macOS*, *Windows*, *DragonFly BSD*, *Plan 9*, *Solaris*, *Android*. Также *Golang* поддерживается набором компиляторов *gcc*, существует несколько независимых реализаций. Ведется разработка второй версии языка.

Язык *Golang* разрабатывался как язык программирования для создания высокоэффективных программ, работающих на современных распределенных системах и многоядерных процессорах. Он может рассматриваться как попытка создать замену языкам *Си* и *C++*. По словам Роба Пайка, «*Golang* был разработан для решения

реальных проблем, возникающих при разработке программного обеспечения в *Google*».

Golang создавался в расчете на то, что программы на нем будут транслироваться в объектный код целевой аппаратной и программной платформы и в дальнейшем исполняться непосредственно, не требуя виртуальной машины, поэтому одним из критериев выбора архитектурных решений была возможность обеспечить быструю компиляцию в эффективный объектный код и отсутствие чрезмерных требований к динамической поддержке.

В результате получился язык, «который не стал прорывом, но тем не менее явился отличным инструментом для разработки крупных программных проектов».

Хотя для *Golang* доступен и интерпретатор, практически в нем нет большой потребности, так как скорость компиляции достаточно высока для обеспечения интерактивной разработки.

Gorilla пакет разработчика специально для упрощения создания веб-приложений на языке *Go*, который, в свою очередь, включает ряд пакетов:

- *gorilla/mux*: позволяет определять более сложные маршруты, которые могут использовать регулярные выражения

- *gorilla/reverse*: используется для создания регулярных выражений для маршрутов

В данном случае задействуем возможности по созданию маршрутов с помощью *gorilla/mux*

1.5 Вывод по разделу

1. Осуществлена постановка задачи, в которой сформированы цели дипломного проекта и задачи для ее достижения.

2. Выполнен обзор двух аналогичных решений, выявлены основные недостатки, такие как высокая цена, устаревший дизайн, и преимущества, которым является их функционал.

3. Выполнен патентный поиск, который позволил выявить насколько изучена тема по автоматизации учебного процесса.

4. Выполнен выбор программной платформы и инструментария для разработки, в результате которого была выбрана фреймворк *Angular*, выбраны языки программирования *TypeScript* и *Golang*. Для дизайна пользовательского интерфейса интернет-сервиса был выбран *AngularMaterial* так как он реализует принципы *Google Material Design*, для упрощения написания стилей выбран метаязык *sass*. В качестве серверной части выступает среда выполнения *Golang*. База данных, предназначенная для хранения всей нужной информации, предоставляемой интернет-сервисом, будет использована *PostgreSQL*.

2 Проектирование интернет-сервиса

2.1 Функциональное наполнение интернет-сервиса

Интернет-сервис, разрабатываемый в рамках данного дипломного проекта предназначен для создания удобной в использовании системы учета и контроля хода выполнения дипломного проектирования в вузе.

Диаграмма вариантов использования для интернет-сервиса представлена на рисунке 2.1 и в приложении А.

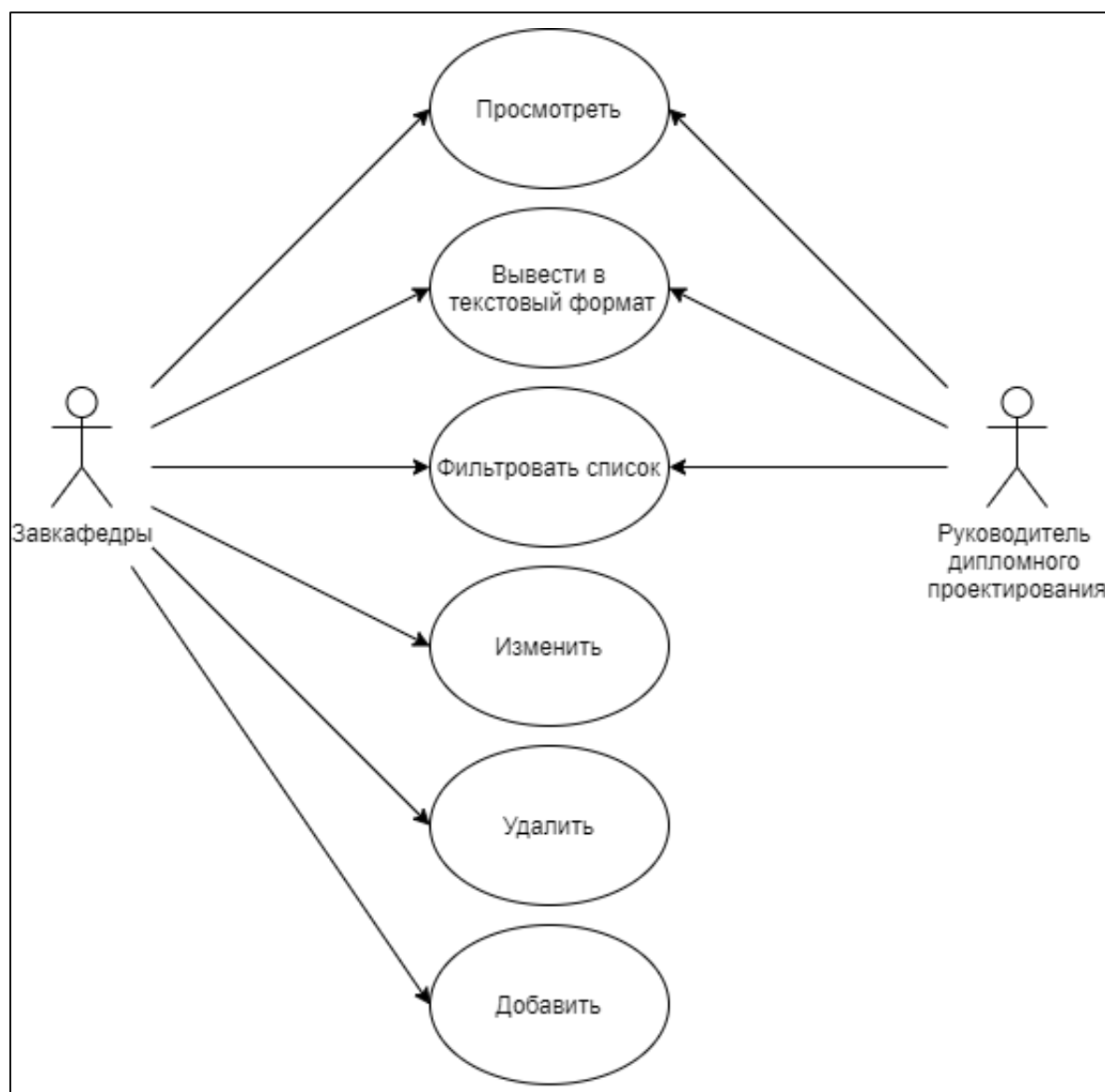


Рисунок 2.1 – Диаграмма вариантов использования

				БГТУ 02.00.ПЗ			
	ФИО	Подпись	Дата				
Разраб.	Криштопчик А.Ю.			2 Проектирование интернет-сервиса		Лит.	Лист
Пров.	Смелов В.В.						1
Консульт.							7
Н. контр.	Рыжанкова А.С.					74417011, 2020	
Утв.	Смелов В.В.						

В таблице 2.1 представлены роли интернет-сервиса и их назначение.

Таблица 2.1 – Виды ролей в приложении

Роль	Назначение
Заведующий кафедрой	Просмотр, изменение, удаление, добавление данных о дипломных проектах или работах, руководителей дипломного проектирования, комиссии, нормоконтролеров, председателей, рецензентов, приказов и специальностей. Фильтрация данных. Вывод данных в документ.
Руководитель дипломного проектирования	Просмотр и фильтрация данных о дипломных проектах или работах. Вывод данных в документ.

В таблице 2.2 представлены функции, доступные пользователям в интернет-сервисе.

Таблица 2.2 – Функции интернет-сервиса

Функция	Пояснение
Просмотр	Просмотр дипломных проектов или работ, руководителей дипломного проектирования, комиссии, нормоконтролеров, председателей, рецензентов, приказов и специальностей.
Вывести в текстовый формат	Вывод списка дипломных проектов или работ в текстовом формате с учетом фильтрации.
Фильтровать список	Фильтрация списка дипломных проектов или работ по приказам, специальностям, студентам, темам, руководителям, рецензентам, нормоконтролерам, председателям, комиссии, сроку исполнения, дате защиты, номеру в очереди и оценке.
Изменить	Изменение дипломных проектов или работ, руководителей дипломного проектирования, комиссии, нормоконтролеров, председателей, рецензентов, приказов или специальностей.
Удалить	Удаление дипломных проектов или работ, руководителей дипломного проектирования, комиссии, нормоконтролеров, председателей, рецензентов, приказов или специальностей.
Добавить	Добавление дипломных проектов или работ, руководителей дипломного проектирования, комиссии, нормоконтролеров, председателей, рецензентов, приказов или специальностей.

2.2 Архитектура интернет-сервиса

Интернет-сервис представляют собой особый тип программ, построенных по трехуровневой архитектуре [15]. Особенность данной архитектуры заключается в

том, что в отличие от распространенной архитектуры «клиент-сервер» предполагается наличие трех компонентов: клиента, серверного приложения и сервера баз данных. Клиент инициирует обращение к программному обеспечению, расположенному на сервере приложений. Сервер приложений формирует запросы к БД на языке *SQL* [16], то есть по сети от сервера приложений к серверу БД передается лишь текст запроса. Результат выполнения запроса копируется на сервер приложений, который возвращает результат в клиентское приложение. Приложение отображает результат выполнения запросов. Передача запросов и результатов их обработки происходит через Интернет, что представлено на рисунке 2.2 и в приложении Б.

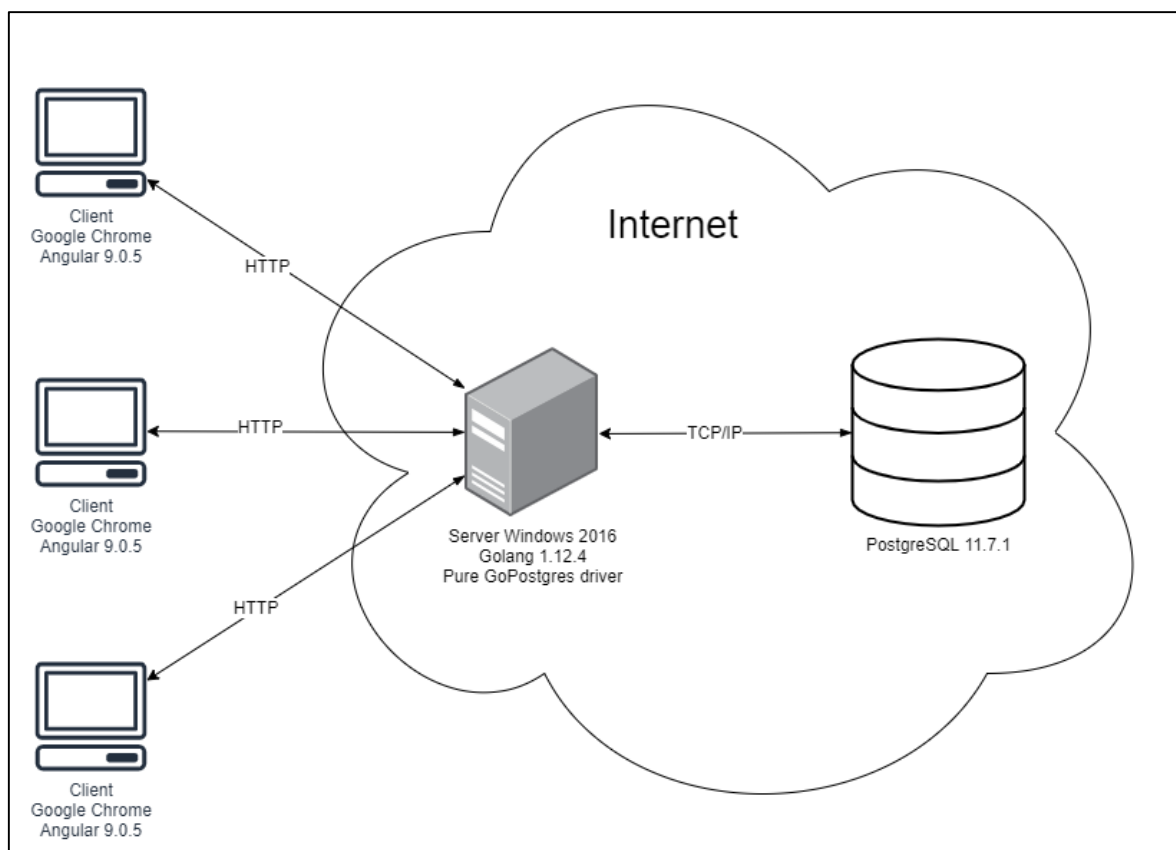


Рисунок 2.2 – Общая схема взаимодействия компонентов

В нашем случае клиентом выступает браузер *Google Chrome* [17], который при помощи протокола *http* [18] обращается к серверной части интернет-сервиса, ответом которого является статические данные, после получения которых клиенту предоставляется возможность взаимодействовать с интерфейсом, написанном на языке программирования *TypeScript* при помощи фреймворка *Angular*.

Серверная-часть – это часть интернет-сервиса, представленная *REST*-сервисами расположенных на *Server Windows 2016*, разработанные на языке программирования *Golang* версии 1.12.4, предоставляющими функциональность, описанную в пункте 2.1. При инициализации соединения клиентом, в зависимости от запроса, отправленного по протоколу *http*, сервер обращается к серверу базы данных, после получения необходимых данных возвращает их клиенту, либо, возвращает статические данные для взаимодействия клиента и серверной части.

2.3 База данных интернет-сервиса

В качестве базы данных был выбран— *PostgreSQL*. Для взаимодействия с базой данных была выбрана *Pure GoPostgres driver* – пакет драйверов, позволяющий взаимодействовать с базой данных напрямую при помощи языка запросов *SQL*.

На рисунке 2.3 и в приложении В представлена логическая схема базы данных.

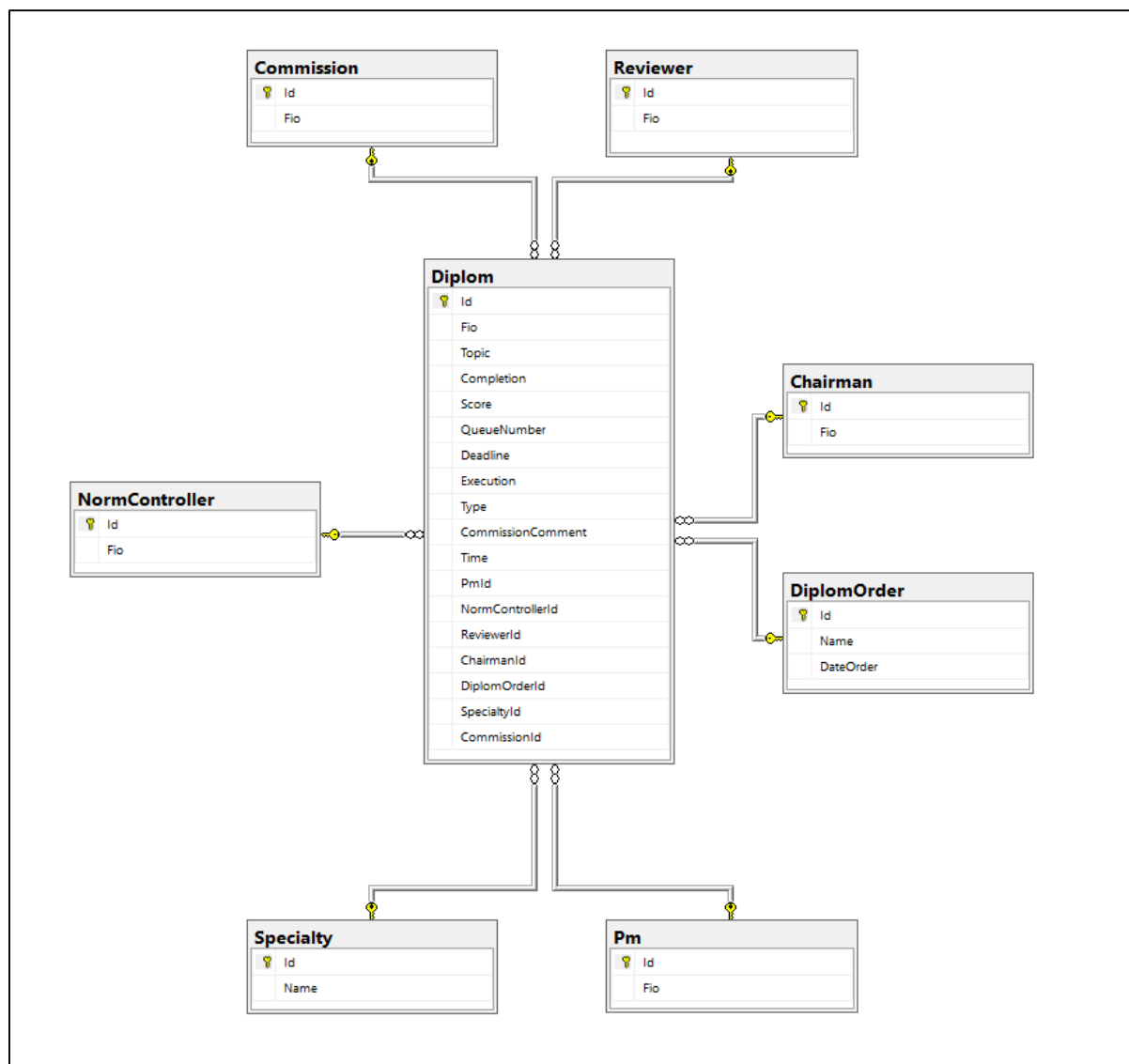


Рисунок 2.3 – Логическая схема базы данных

Таблица *Pm* предназначена для хранения руководителей дипломных проектов. Перечень полей таблицы *Pm* приведено в таблице 2.3.

Таблица 2.3 – Описание полей таблицы *Pm*

Название	Типы данных, ограничение целостности
Id	Integer primary key
Fio	Varchar(30)

Свойство *Id* отвечает за идентификацию руководителя.

Свойство *Fio* содержит имя руководителя.

Таблица *Normcontroller* предназначена для хранения нормоконтролеров. Описание полей *Normcontroller* приведено в таблице 2.4.

Таблица 2.4 – Описание полей таблицы *Normcontroller*

Название	Типы данных, ограничение целостности
Id	Integer primary key
Fio	Varchar(30)

Свойство *Id* отвечает за идентификацию нормоконтролера.

Свойство *Fio* содержит имя нормоконтролера.

Таблица *Reviewer* предназначена для хранения рецензентов. Описание полей *Reviewer* приведено в таблице 2.5.

Таблица 2.5 – Описание полей таблицы *Reviewer*

Название	Типы данных, ограничение целостности
Id	Integer primary key
Fio	Varchar(30)

Свойство *Id* отвечает за идентификацию рецензента.

Свойство *Fio* содержит имя рецензента.

Таблица *Chairman* предназначена для хранения председателей. Описание полей *Chairman* приведено в таблице 2.6.

Таблица 2.6 – Описание полей таблицы *Chairman*

Название	Типы данных, ограничение целостности
Id	Integer primary key
Fio	Varchar(30)

Свойство *Id* отвечает за идентификацию председателя.

Свойство *Fio* содержит имя председателя.

Таблица *Diplomorder* предназначена для хранения приказов. Описание полей *Diplomorder* приведено в таблице 2.7.

Таблица 2.7 – Описание полей таблицы *Diplomorder*

Название	Типы данных, ограничение целостности
Id	Integer primary key
Name	Varchar(30)
Dateorder	Date

Свойство *Id* отвечает за идентификацию приказа.

Свойство *Name* содержит название приказа.

Свойство *Dateorder* содержит дату приказа.

Таблица *Specialty* предназначена для хранения специальностей. Описание полей *Specialty* приведено в таблице 2.8.

Таблица 2.8 – Описание полей таблицы *Specialty*

Название	Типы данных, ограничение целостности
Id	Integer primary key
Name	Varchar(30)

Свойство *Id* отвечает за идентификацию специальности.

Свойство *Name* содержит название специальности.

Таблица *Commission* предназначена для хранения комиссии. Описание полей *Commission* приведено в таблице 2.9.

Таблица 2.9 – Описание полей таблицы *Commission*

Название	Типы данных, ограничение целостности
Id	Integer primary key
Fio	Varchar(30)

Свойство *Id* отвечает за идентификацию комиссии.

Свойство *Name* содержит имя комиссии.

Таблица *Diplom* предназначена для хранения данных о дипломном проекте или работе. Описание полей *Diplom* приведено в таблице 2.10.

Таблица 2.10 – Описание полей таблицы *Diplom*

Название	Типы данных, ограничение целостности
Id	Integer foreign key
Fio	Varchar(30)
Topic	Varchar(100)
Completion	Integer
Score	Integer
Deadline	Date
Queuenummer	Integer
Pmid	Integer foreign key
Normcontrollerid	Integer foreign key
Reviewerid	Integer foreign key
Chairmanid	Integer foreign key
Diplomorderid	Integer foreign key
Specialtyid	Integer foreign key
Execution	Date
Type	Integer
Commissioncomment	Varchar(100)
Time	Integer

Свойство *Id* отвечает за идентификацию дипломного проекта или работы.

Свойство *Fio* содержит имя студента.

Свойство *Topic* содержит название дипломного проекта или работы.

Свойство *Completion* содержит процент завершения дипломного проекта или работы.

Свойство *Score* содержит оценку за дипломный проект или работу.

Свойство *Deadline* содержит дату защиты дипломного проекта или работы.

Свойство *Queue number* содержит номер в очереди.

Свойство *Pmid* внешний ключ, указывающий на руководителя дипломного проекта или работы.

Свойство *Normcontrollerid* внешний ключ, указывающий на нормоконтролера дипломного проекта или работы.

Свойство *Reviewerid* внешний ключ, указывающий на рецензента дипломного проекта или работы.

Свойство *Chairmanid* внешний ключ, указывающий на председателя дипломного проекта или работы.

Свойство *Diplomorderid* внешний ключ, указывающий на приказ дипломного проекта или работы.

Свойство *Specialtyid* внешний ключ, указывающий на специальность студента.

Свойство *Execution* содержит дату исполнения.

Свойство *Type* содержит тип диплома.

Свойство *Commissioncomment* содержит комментарий к дипломному проекту или работе от председателя комиссии.

Свойство *Time* содержит время защиты.

2.4 Вывод по разделу

1. Разработана диаграмма вариантов использования которая включает в себя две роли: заведующий кафедры и руководитель дипломного проектирования, а также шести функций: просмотреть, вывести в текстовый формат, фильтровать список, изменить, удалить, добавить.

2. Разработана архитектура интернет-сервиса представляющая из себя трех-уровневую модель и состоит из трех частей: клиентская, серверная и база данных. Описаны способы взаимодействия частей между друг другом.

3. Разработана логическая схема базы данных. База данных состоит из восьми таблиц: *Pm*, *Normcontroller*, *Reviewer*, *Chairman*, *Diplomorder*, *Specialty*, *Commission* и *Diplom*. Описано взаимодействие таблиц, их поля и назначение каждого поля.

3 Разработка интернет-сервиса

Архитектура интернет-сервиса представлена в главе 2 и включает три основных части: клиентскую часть, серверную часть и базу данных.

3.1 Разработка серверной части интернет-сервиса

Для реализации серверной части был выбран *Golang*. Структура директориев серверной-части представлена на рисунке 3.1.

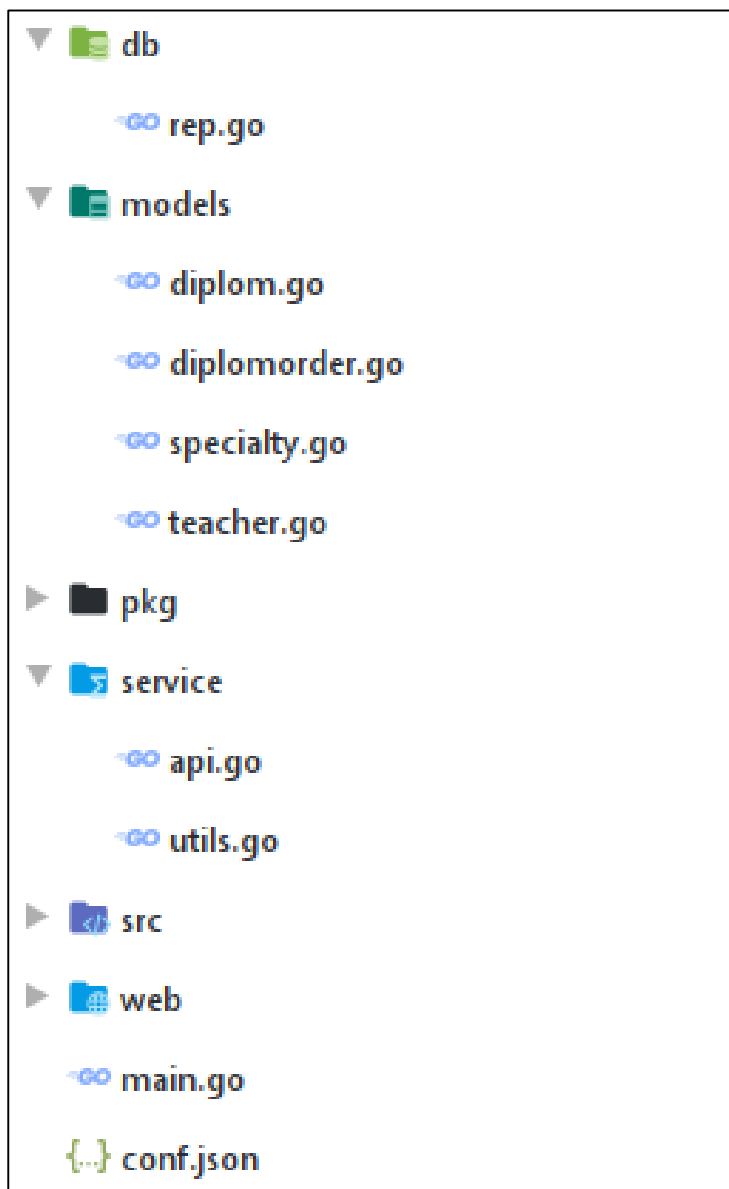


Рисунок 3.1 – Структура директориев серверной части

				БГТУ 03.00.ПЗ						
	ФИО	Подпись	Дата							
Разраб.	Кришточник А.Ю.			3 Разработка интернет-сервиса			Лит.	Лист	Листов	
Пров.	Смелов В.В.								1	8
Консульт.							74417011, 2020			
Н. контр.	Рыжанкова А.С.									
Утв.	Смелов В.В.									

Разработка на *Golang* сводится к созданию приложения при помощи пакетов. Основные пакеты приведены в таблице 3.1.

Таблица 3.1 – Описание основных пакетов серверной части

Пакет	Функционал
main	Стартовая точка приложения
db	Пакет описывающий работу с базой данных
service	Роутер для разделения обработки логики согласно маршруту и создание файла регистрации
models	Пакет для создание необходимых структур объектов

Для создание сервера, первоначально запускаемым файлом является *main.go*, исходный код пакета *main* представлен в приложении Г (а), это стартовая точка приложения, в нем создается файл регистрации при помощи *InitConfiguration* из пакета *service*, исходный код пакета *service* представлен в приложении Г (б), для отслеживания состояния сервера после чего для обработки входящих запросов создается обработчик маршрутов роутера при помощи вызова *RunRest* из пакета *service*. Вызов функций представлен на рисунке 3.2.

```
func main() {
    defer func() {
        if err := recover(); err != nil {
            log.Error(err)
        }
    }()
    err := service.InitLog()
    if err != nil {
        log.Error("Cannot init logs: ", err)
    }
    log.Info("Starting...")
    var cnf = flag.String("c", "conf.json", "Config file name (in a current dir)")
    flag.Parse()
    log.Info("Loading with config: ", *cnf)
    service.InitConfiguration(*cnf, service.Conf)
    service.RunRest()
}
```

Рисунок 3.2 – Создание сервера

При вызове *RunRest* из пакета *server*, происходит создание обработчиков маршрутов. При указании в *url* описанных запроса происходит вызов обработчиков. Также создается обработчик, который обращается к серверу для раздачи статических файлов, по этому пути будет раздаваться клиентской части интернет-сервиса. Создание обработчика маршрутов представлено на рисунке 3.3.

```
func RunRest() {
    r := mux.NewRouter()
    r.HandleFunc(path: "/api/diploms", getDiploms).Methods(methods...: "GET")
    r.HandleFunc(path: "/api/diploms/{id:[0-9]+}", getDiplom).Methods(methods...: "GET")
    r.HandleFunc(path: "/api/diploms/{id:[0-9]+}", deleteDiplom).Methods(methods...: "DELETE")
    ...
    r.PathPrefix(tpl: "/").Handler(http.FileServer(http.Dir(Config.StaticPath)))
}
```

Рисунок 3.3 – Создание обработчика маршрутов и привязка обработчиков

Для реализации работы с базой данных в данном дипломном проекте был выбран пакет *pq* [8]. Пакет *pq* - это драйвер *Go Postgres* для пакета *database/sql*. Для работы с базой данных с помощью *pq* необходимо выполнить следующие этапы.

Создать подключение к базе данных. Для этого необходимо создать строку подключения указав необходимые параметры, исходный код пакета *db* представлен в приложении Г (в). На рисунке 3.4 представлено создание подключения к базе данных.

```
func createConnectin() *sql.DB {
    connStr := "user=postgres password=123123 dbname=postgres sslmode=disable"
    db, err := sql.Open(driverName: "postgres", connStr)
    if err != nil {
        panic(err)
    }
    return db
}
```

Рисунок 3.4 – Создание подключения к базе данных

Данный объект подключения в дальнейшем используется для работы с базой данных. Например, для выполнения *sql* [9] запроса используется метод *db.Query*. Пример вызова *sql* конструкции представлен на рисунке 3.5.

```
func GetCommissionn() {
    db := createConnectin()
    rows, err := db.Query(query: "select * from commission")
    if err != nil {
        panic(err)
    }
    defer rows.Close()
}
```

Рисунок 3.5 – Вызов sql конструкции

В результате выполнения данной функции в переменную *rows* запишутся все данные из таблицы *commission*.

3.2 Разработка клиентской части интернет-сервиса

Клиентская часть представляет из себя веб-приложение, написанное на *TypeScript*. Для ускорения разработки был выбран фреймворк *Angular*. Структура директориев клиентской части представлена на рисунке 3.6.

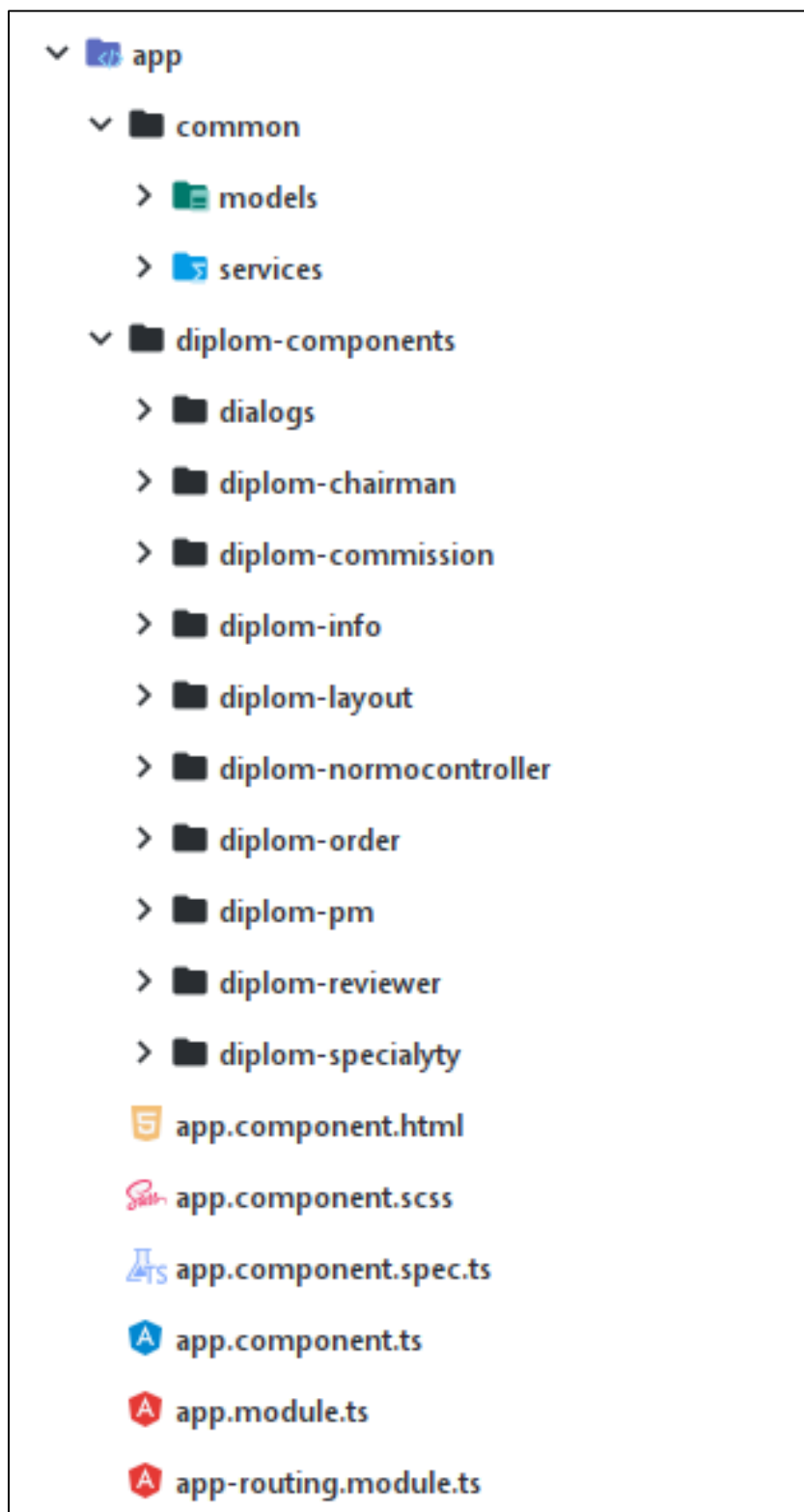


Рисунок 3.6 – Структура директориев клиентской части интернет-сервиса

Разработка клиентской части сводится к созданию компонентов для отображения информации в окне браузера и классов для вспомогательных функций, таких как: взаимодействия по протоколу *http* или модели объектов. Папка *common* предназначена для хранения вспомогательных файлов, так в папке *services* находятся *TypeScript* файлы при помощи которых можно обратиться к серверной части по протоколу *http*, а в папке *models* модели объектов. Компоненты находятся в папке *diplom-components* и приведены в таблице 3.2.

Таблица 3.2 – Описание основных компонентов клиентской части

Компонент	Функционал
diplom-layout	Родительский компонент, контролирующий отображение необходимого компонента при помощи роутера
diplom-info	Компонент содержащий в себе логику связанную с добавлением, обновление, удалением и отображением дипломных проектов или работ
diplom-chairman	Компонент содержащий в себе логику связанную с добавлением, обновление, удалением и отображением председателей
diplom-commission	Компонент содержащий в себе логику связанную с добавлением, обновление, удалением и отображением комиссии
diplom-normocontroller	Компонент содержащий в себе логику связанную с добавлением, обновление, удалением и отображением нормоконтролеров
diplom-order	Компонент содержащий в себе логику связанную с добавлением, обновление, удалением и отображением приказов
diplom-pm	Компонент содержащий в себе логику связанную с добавлением, обновление, удалением и отображением дипломных руководителей
diplom-reviewer	Компонент содержащий в себе логику связанную с добавлением, обновление, удалением и отображением рецензентов
diplom-specialty	Компонент содержащий в себе логику связанную с добавлением, обновление, удалением и отображением специальностей
dialogs	Компонент содержащий в себе логику связанную с добавлением, обновление, удалением и отображением дипломных проектов или работ

Исходный код компонентов клиентской части представлен в приложении Д.

3.3 Разработка базы данных интернет-сервиса

База данных сформированная в разделе 2.3 имеет восемь таблиц, для реализации таблицы *Diplom* создан скрипт, представленный на рисунке 3.7.

```
CREATE TABLE Diplom
(
    Id SERIAL PRIMARY KEY,
    Fio CHARACTER VARYING(30),
    Topic CHARACTER VARYING(30),
    Completion INTEGER,
    Score INTEGER,
    QueueNumber INTEGER,
    Deadline DATE,
    Execution DATE,
    Type INTEGER
    CommissionComment VARYING(30),
    Timer INTEGER,
    PmId INTEGER REFERENCES pm(Id),
    NormControllerId INTEGER REFERENCES normcontroller(Id),
    ReviewerId INTEGER REFERENCES reviewer(Id),
    ChairmanId INTEGER REFERENCES chairman(Id),
    DiplomOrderId INTEGER REFERENCES diplomorder(Id),
    SpecialtyId INTEGER REFERENCES specialty(Id),
    CommissionId INTEGER REFERENCES commission(Id)
);
```

Рисунок 3.7– Скрипт создания таблиц Diplom

Для создания таблицы *Pm* скрипт представлен на рисунке 3.8.

```
CREATE TABLE Pm
(
    Id SERIAL PRIMARY KEY,
    Fio CHARACTER VARYING(30)
);
```

Рисунок 3.8 – Скрипт создания таблиц Pm

Для создания таблицы *NormController* скрипт представлен на рисунке 3.9.

```
CREATE TABLE NormController
(
    Id SERIAL PRIMARY KEY,
    Fio CHARACTER VARYING(30)
);
```

Рисунок 3.9 – Скрипт создания таблиц NormController

Для создания таблицы *Reviewer* скрипт представлен на рисунке 3.10.

```
CREATE TABLE Reviewer
(
    Id SERIAL PRIMARY KEY,
    Fio CHARACTER VARYING(30)
);
```

Рисунок 3.10 – Скрипт создания таблиц Reviewer

Для создания таблицы *Chairman* скрипт представлен на рисунке 3.11.

```
CREATE TABLE Chairman
(
    Id SERIAL PRIMARY KEY,
    Fio CHARACTER VARYING(30)
);
```

Рисунок 3.11 – Скрипт создания таблиц Chairman

Для создания таблицы *Commission* скрипт представлен на рисунке 3.12.

```
CREATE TABLE Commission
(
    Id SERIAL PRIMARY KEY,
    Fio CHARACTER VARYING(30)
);
```

Рисунок 3.12 – Скрипт создания таблиц Commission

Для создания таблицы *DiplomOrder* скрипт представлен на рисунке 3.13.

```
CREATE TABLE DiplomOrder
(
    Id SERIAL PRIMARY KEY,
    Name CHARACTER VARYING(30),
    DateOrder date
);
```

Рисунок 3.13 – Скрипт создания таблиц DiplomOrder

Для создания таблицы *Specialty* скрипт представлен на рисунке 3.14.

```
CREATE TABLE Specialty
(
    Id SERIAL PRIMARY KEY,
    Name CHARACTER VARYING(30)
);
```

Рисунок 3.14 – Скрипт создания таблиц Specialty

Данные в базу данных будут записываться непосредственно из клиентской части при помощи серверной части, описанной в разделе 3.1.

3.4 Вывод по разделу

1. Разработана серверная часть интернет-сервиса. Серверная часть интернет-сервиса включает в себе четыре основных пакета, описан их функционал и способ взаимодействия друг с другом.

2. Разработана клиентская часть интернет-сервиса. Клиентская часть интернет-сервиса включает в себе десять основных компонента, также два дополнительных компонента, для взаимодействия по протоколу *http* и для создания моделей объектов.

3. Разработана база данных интернет-сервиса. База данных состоит из восьми таблиц, описан скрипт для создания каждой таблицы базы данных.

4 Тестирование интернет-сервиса

Для тестирования был выбран тест-кейс план [19].

В данной программе предусмотрены пользователи двух видов:

- завкафедры;
- руководитель дипломного проектирования.

Руководитель дипломного проектирования – это преподаватель, который имеет право только просматривать информацию в данном интернет-сервисе. Завкафедры обладает большим набором функций:

- изменить;
- добавить;
- удалить.

Тест-кейсами будут являться функции, описанные в разделе 2.1.

4.1 Просмотреть списки

Стартовая страница интернет-сервиса для завкафедры представлена на рисунке 4.1.

Дипломы		Руководители		Рецензенты		Нормоконтролеры		Председатели		Комиссия		Приказы		Специальности		Вывод	
Фильтр		Добавить															
Приказ		Пр 1 11.06.2020		[48-С, 04.03.2020]-8-ИСИТ-Булова Анна Фёдоровна/Жилик Н.А.		Редактировать		▼									
Специальность		--не выбран--		[48-С, 04.03.2020]-9-ИСИТ-Андреюк Ирина Сергеевна/Пустовалова Н.Н.		Редактировать		▼									
Студент:		<input type="radio"/> Проект <input type="radio"/> Работа		[Пр2, 11.05.2020]-3-ПОИТ-Дубовик Евгений Игоревич/Толкач И.В.		Редактировать		▼									
Тема				[48-С, 04.03.2020]-1-ИСИТ-Дорогокупцев Никита Геннадьевич/Жилик Н.А.		Редактировать		▼									
Руководитель		--не выбран--		[Пр 1, 11.06.2020]-4-ПОИТ-Криштопчик Артём Юрьевич/Смелов В.В.		Редактировать		^									
Рецензент		--не выбран--		Тема: Интернет-сервис для учета и контроля выполнения дипломного проектирования в вузе													
Нормоконтролер		--не выбран--		Нормоконтролер: --не выбран-- Рецензент: --не выбран-- Срок исполнения: 28.05.2020													
Председатель		--не выбран--		Комиссия: --не выбран-- Комментарий: ошибок нет													
Комиссия		--не выбран--		Дата защиты: 04.06.2020 Номер в очереди: 4 Председатель: --не выбран--													
				Оценка: 0													
				Защита Удалить													

Рисунок 4.1 – Стартовая страница интернет-сервиса для завкафедры

На данной страницы, завкафедры может просмотреть имеющиеся дипломные проекты и работы, изменить их данные, удалить или добавить новые.

				БГТУ 04.00.ПЗ						
	ФИО	Подпись	Дата							
Разраб.	Криштопчик А.Ю.			4 Тестирование интернет-сервиса	Лит.			Лист	Листов	
Пров.	Смелов В.В.							1	9	
Консульт.					74417011, 2020					
Н. контр.	Рыжанкова А.С.									
Утв.	Смелов В.В.									

Стартовая страница для руководителей дипломных проектов выглядит иначе, и представлена на рисунке 4.2.

Дипломы	Вывод	Завкафедры
Фильтр		
Приказ Пр 1 11.06.2020	<input type="checkbox"/>	[48-С, 04.03.2020]-8-ИСИТ-Булова Анна Фёдоровна/Жиляк Н.А.
Специальность --не выбран--	<input type="checkbox"/>	[48-С, 04.03.2020]-9-ИСИТ-Андреюк Ирина Сергеевна/Пустовалова Н.Н.
Студент: <input type="radio"/> Проект <input type="radio"/> Работа	<input type="checkbox"/>	[Пр2, 11.05.2020]-3-ПОИТ-Дубовик Евгений Игоревич/Толкач И.В.
Тема	<input type="checkbox"/>	[48-С, 04.03.2020]-1-ИСИТ-Дорогоголец Никита Геннадьевич/Жиляк Н.А.
Руководитель --не выбран--	<input type="checkbox"/>	[Пр 1, 11.06.2020]-4-ПОИТ-Криштопчик Артём Юрьевич/Смелов В.В.
Рецензент --не выбран--	<input type="checkbox"/>	Тема: Интернет-сервис для учета и контроля выполнения дипломного проектирования в вузе Нормоконтролер: --не выбран-- Рецензент: --не выбран-- Срок исполнения: 28.05.2020 Комиссия: --не выбран-- Комментарий: ошибок нет
Нормоконтролер --не выбран--	<input type="checkbox"/>	Дата защиты: 04.06.2020 Номер в очереди: 4 Председатель: --не выбран-- Оценка: 0
Председатель --не выбран--	<input type="checkbox"/>	
Комиссия --не выбран--	<input type="checkbox"/>	Защита

Рисунок 4.2 – Стартовая страница интернет-сервиса для руководителей дипломного проектирования

На данной страницы, руководителей дипломного проектирования может просмотреть имеющиеся дипломные проекты и работы.

Для просмотра руководителей, необходимо перейти на вкладку Руководители, данная вкладка представлена на рисунке 4.3.

Дипломы	Руководители	Рецензенты	Нормоконтролеры	Председатели	Комиссия	Приказы	Специальности	Вывод
Добавить								
Руководитель:								
Добавить								
Руководитель: Смелов В.В. Редактировать Удалить								
Руководитель: Жиляк Н.А. Редактировать Удалить								
Руководитель: Жук Я.А. Редактировать Удалить								
Руководитель: Пустовалова Н.Н. Редактировать Удалить								
Руководитель: Чурак Е.В. Редактировать Удалить								
Руководитель: Толкач И.В. Редактировать Удалить								
Руководитель: Дубовик М.В. Редактировать Удалить								
Руководитель: Блинова Е.А. Редактировать Удалить								
Руководитель: Бурмакова А.В. Редактировать Удалить								

Рисунок 4.3 – Вкладка руководителей для завкафедры

На данной страницы, завкафедры может просмотреть имеющихся дипломных руководителей, изменить, удалить или добавить новых.

Для просмотра рецензентов, нормоконтролеров, председателей, приказов и специальностей, достаточно переключиться на соответствующую вкладку аналогично руководителям.

4.2 Добавить в список

Для добавления в списки новых дипломов, заведующему кафедрой необходимо перейти на необходимую вкладку, после чего открыть форму, для добавления которая представлена на рисунке 4.4.

Фильтр	Добавить
Приказ	
48-С 04.03.2020	▼
Специальность	
ИСиТ	▼
Студент:	
Криштопчик Артём Юрьевич	
<input type="radio"/> Проект <input type="radio"/> Работа	
Тема:	
Интернет-сервис для учета и контроля выполнения дипломного проектирования в вузе	⌵
Руководитель	
Смелов В.В.	▼
Рецензент	
–не выбран–	▼
Нормоконтролер	
Рыжанкова А. С.	▼
Комиссия	
–не выбран–	▼
Комментарий:	⌵
Председатель	
Дюбков В. К.	▼
Процент выполнения:	
Срок исполнения:	
28.05.2020	📅
Дата защиты:	
04.06.2020	📅 №:
Оценка:	
Добавить	

Рисунок 4.4 – Форма для добавления диплома

Заполнив необходимые поля и нажав кнопку добавить, диплом появится в списке всех дипломов.

Для добавления руководителей дипломного проектирования необходимо перейти на вкладку руководители, после чего заполнить форму, представленную на рисунке 4.5.

Рисунок 4.5 – Форма для добавления руководителя дипломного проектирования

Заполнив необходимое поле и нажав кнопку добавить, руководитель появится в списке всех руководителей. Данная последовательность действий аналогичная для добавления рецензентов, нормоконтролеров, председателей и председателей комиссии, достаточно перейти на необходимую вкладку, заполнить форму и нажать кнопку добавить.

4.3 Удалить из списков

Для удаления дипломного проекта или работы необходима перейти на вкладку дипломы, после чего открыть необходимый диплом получив информацию о дипломе представленную на рисунке 4.6.

[Пр 1, 11.06.2020]-4-ПОИТ-Криштопчик Артём Юрьевич/Смелов В.В.			Редактировать	^
Тема: Интернет-сервис для учета и контроля выполнения дипломного проектирования в вузе				
Нормоконтролер: --не выбран--	Рецензент: --не выбран--	Срок исполнения: 28.05.2020		
Комиссия: --не выбран--	Комментарий: ошибок нет			
Дата защиты: 04.06.2020	Номер в очереди: 4	Председатель: --не выбран--		
Оценка: 0				
			Защита	Удалить

Рисунок 4.6 – Информация о дипломе

При нажатии на кнопку удалить открывается диалоговое окно, представленное на рисунке 4.7.

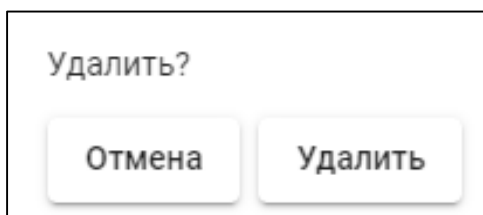


Рисунок 4.7 – Диалоговое окно при удалении

При нажатии на кнопку удалит, дипломный проект удалится из списка.

Для удаления руководителя дипломного проектирования необходима перейти на вкладку дипломы, после чего нажать на кнопку удалить, представленную на рисунке 4.8.

Руководитель: Смелов В.В.	Редактировать	Удалить
Руководитель: Жиляк Н.А.	Редактировать	Удалить
Руководитель: Жук Я.А.	Редактировать	Удалить
Руководитель: Пустовалова Н.Н.	Редактировать	Удалить
Руководитель: Чурак Е.В.	Редактировать	Удалить
Руководитель: Толкач И.В.	Редактировать	Удалить
Руководитель: Дубовик М.В	Редактировать	Удалить
Руководитель: Блинова Е.А.	Редактировать	Удалить
Руководитель: Бурмакова А.В	Редактировать	Удалить

Рисунок 4.8 – Руководители дипломного проектирования

После нажатия на кнопку удалить, откроется диалоговое окно, представленное на рисунке 4.7 и после нажатии на кнопку удалить, руководитель удалится из списка. Данная последовательность действий аналогичная для удаления рецензентов, нормоконтролеров, председателей, председателей комиссии, приказов и специальностей, достаточно перейти на необходимую вкладку, выбрать необходимое поле и нажать кнопку удалить.

4.4 Изменить данные в списках

Для изменения имеющихся дипломных проектов или работ необходимо перейти на вкладку дипломы, после чего, у необходимого диплома нажать на кнопку редактировать, откроется форма, представленная на рисунке 4.9.

Приказ	Пр 1 11.06.2020	▼
Специальность	ИСиТ	▼
Студент:	Криштопчик Артём Юрьевич	
	<input checked="" type="radio"/> Проект	<input type="radio"/> Работа
Тема:	Интернет-сервис для учета и контроля выполнения дипломного проектирования в вузе	
Руководитель	Смелов В.В.	▼
Рецензент	–не выбран–	▼
Нормоконтролер	–не выбран–	▼
Комиссия	–не выбран–	▼
Комментарий:	ошибок нет	
Председатель	–не выбран–	▼
Процент выполнения:	0	
Срок исполнения:	28.05.2020	📅
Дата защиты:	04.06.2020	📅
	№:	4
Оценка:	0	
<div>Изменить</div>		

Рисунок 4.9 – Форма для изменения дипломного проекта или работы

Изменив необходимые поля и нажав кнопку изменить, диплом изменится в списке всех дипломов.

Для изменения руководителей дипломного проектирования необходимо перейти на вкладку руководители, после чего у выбранного дипломного руководителя нажать на кнопку редактировать, после чего откроется форма, представленная на рисунке 4.10.

Рисунок 4.10 – Форма для изменения руководителя дипломного проектирования

Изменив поле и нажав кнопку изменить, руководитель изменится в списке всех руководителей. Данная последовательность действий аналогичная для изменения рецензентов, нормоконтролеров, председателей и председателей комиссии.

Для изменения приказов необходимо перейти на вкладку приказы, после чего у выбранного приказа нажать на кнопку редактировать, после чего откроется форма, представленная на рисунке 4.11.

Рисунок 4.11 – Форма для изменения приказа

Заполнив необходимые поля и нажав кнопку изменить, приказ обновится в списке всех приказов.

4.5 Фильтровать списки

Для фильтрации списка дипломных проектов или работ необходимо перейти на вкладку дипломы, и заполнить форму фильтр представленной на рисунке 4.12.

Фильтр	Добавить
Приказ 48-С 04.03.2020	<input checked="" type="checkbox"/>
Специальность ИСИТ	<input checked="" type="checkbox"/>
Студент:	<input type="checkbox"/>
<input checked="" type="radio"/> Проект <input type="radio"/> Работа	<input type="checkbox"/>
Тема	<input type="checkbox"/>
Руководитель Смелов В.В.	<input checked="" type="checkbox"/>
Рецензент -не выбран-	<input type="checkbox"/>
Нормоконтролер -не выбран-	<input type="checkbox"/>
Председатель -не выбран-	<input type="checkbox"/>
Комиссия -не выбран-	<input type="checkbox"/>
Процент выполнения:	<input type="checkbox"/>
Срок исполнения:	<input type="checkbox"/>
Дата защиты:	<input type="checkbox"/>
Номер в очереди	<input type="checkbox"/>
Оценка	<input type="checkbox"/>
Фильтр	

Рисунок 4.12 – Форма фильтра списка дипломных проектов

После заполнения и выбора необходимых полей при нажатии на кнопку фильтр список дипломных проектов или работ изменится в соответствии с выбранными полями.

4.6 Вывести в текстовый формат

Для вывода данных в текстовый файл для заведующего кафедрой и руководителей дипломного проектирования доступна вкладка «Вывод», после нажатия на которую отфильтрованный список дипломных работ и проектов будет скачен в качестве текстового файла, результат вывода приведен на рисунке 4.13.

Дипломы		Руководители	Рецензенты	Нормоконтролеры	Председатели	Комиссия	Приказы	Специальности	Вывод
Фильтр	Добавить								
Приказ 48-С. 04.03.2020	<input type="checkbox"/>	[48-С, 04.03.2020]-2-ИСИТ-Акшевская Екатерина Игоревна/Жук Я.А.							
Специальность ИСИТ	<input checked="" type="checkbox"/>	[48-С, 04.03.2020]-3-ИСИТ-Граховский Денис Витальевич/Жияк Н.А.							
Студент:	<input type="checkbox"/>	[48-С, 04.03.2020]-5-ИСИТ-Бобр Дмитрий Александрович/Чурак Е.В.							
<input checked="" type="radio"/> Проект <input type="radio"/> Работа	<input checked="" type="checkbox"/>	[48-С, 04.03.2020]-2-ИСИТ-Вольский Дмитрий Михайлович/Жияк Н.А.							
Тема	<input type="checkbox"/>	[Пр 1, 11.06.2020]-3-ИСИТ-Житковский Глеб Сергеевич/Жияк Н.А.							
Руководитель Смелов В.В.	<input type="checkbox"/>	[Пр 1, 11.06.2020]-4-ИСИТ-Ингинен Алексей Валерьевич/Жияк Н.А.							
Рецензент --не выбран--	<input type="checkbox"/>	[Пр 1, 11.06.2020]-6-ИСИТ-Вечер Максим Леонидович/Смелов В.В.							
Нормоконтролер --не выбран--	<input type="checkbox"/>	[48-С, 04.03.2020]-8-ИСИТ-Булова Анна Фёдоровна/Жияк Н.А.							
Председатель --не выбран--	<input type="checkbox"/>	[48-С, 04.03.2020]-9-ИСИТ-Андреек Ирина Сергеевна/Пустовалова Н.Н.							
	<input type="checkbox"/>	[48-С, 04.03.2020]-1-ИСИТ-Дорогокупцев Никита Геннадьевич/Жияк Н.А.							
	<input type="checkbox"/>	[Пр 1, 11.06.2020]-4-ИСИТ-Криштопчик Артём Юрьевич/Смелов В.В.							

Рисунок 4.13 – Результат вывода в текстовый формат

В результате видно, что все функции интернет-сервиса работают корректно.

4.7 Вывод по разделу

1. Протестирован способ просмотра дипломных проектов или работ, руководителей, нормоконтролеров, рецензентов, председателей, комиссии, приказов в интернет-сервисе для руководителей дипломного проектирования и заведующего кафедрой. В результате теста ошибок выявлено не было.

2. Протестирован способ добавления новой информации, данная функция имеется только у заведующего кафедрой и включает в себя добавление новых дипломов, руководителей, нормоконтролеров, рецензентов, председателей, комиссии, приказов и специальностей. В результате теста ошибок выявлено не было.

3. Протестирован способ изменения данных, данная функция имеется только у заведующего кафедрой и включает в себя изменения дипломных проектов или работ, руководителей, нормоконтролеров, рецензентов, председателей, комиссии, приказов и специальностей. В результате теста ошибок выявлено не было.

4. Протестирован способ удаления данных, данная функция имеется только у заведующего кафедрой и включает в себя удаления дипломных проектов или работ, руководителей, нормоконтролеров, рецензентов, председателей, комиссии, приказов и специальностей. В результате теста ошибок выявлено не было.

5. Протестирован способ фильтрации дипломных проектов или работ, данная функция имеется у руководителей дипломного проектирования и заведующего кафедрой. В результате теста ошибок выявлено не было.

6. Протестирован способ вывода в текстовый файл отфильтрованного списка дипломных проектов или работ. В результате мы получили текстовый файл с необходимыми данными.

5 Руководство программиста

Так как разработанный интернет-сервис состоит из трех компонентов: клиентская часть, серверная и база данных, рассмотрим их установка по отдельности.

5.1 Установка серверной части интернет-сервиса

Для запуска серверной части интернет-сервиса необходима установить компилятор языка программирования *Golang*, это возможно сделать, перейдя на официальную страницу и выбрать пункт *Downlad Go* представленную на рисунке 5.1.

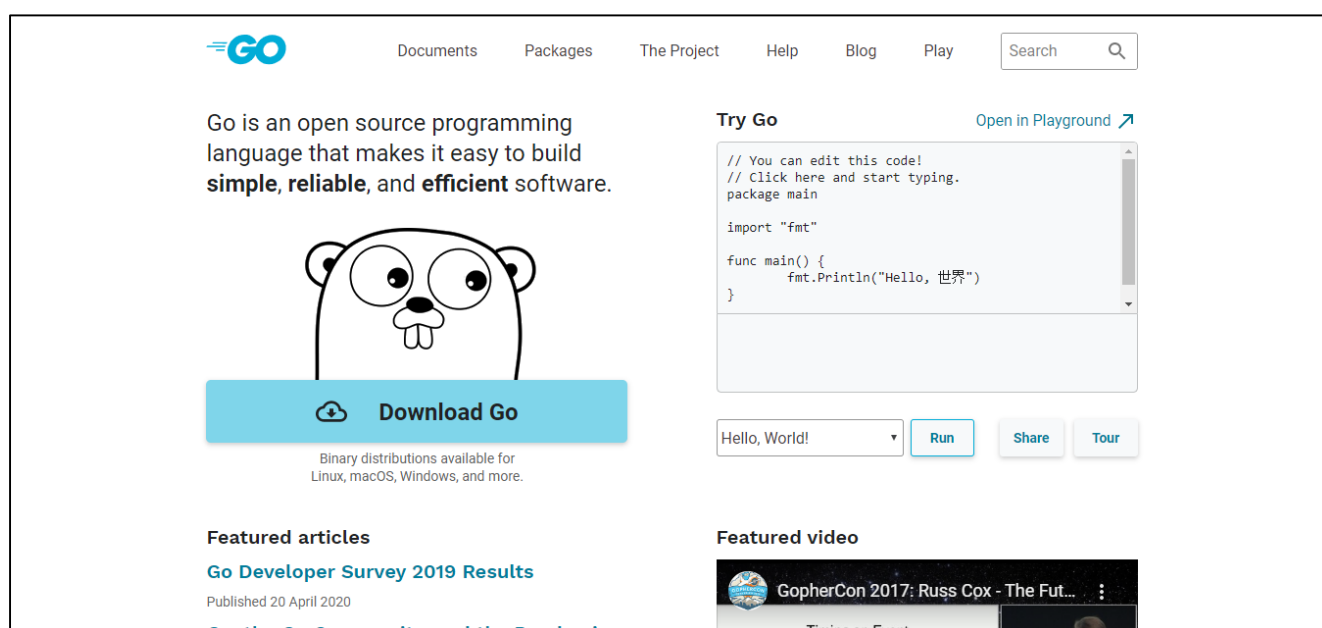


Рисунок 5.1 – Официальная страница Golang

На данной страницы прямо в браузере мы можем попробовать запустить свою первую программу на языке *Golang*, так же тут представлена документация, блог разработчиков, песочница для разработки, и перечень основных пакетов. Для скачивания нам необходимо нажать на кнопку *Downlad Go*, после чего выбираем пункт скачать для *Windows*, так как операционной системой нашего сервера является *Server Windows 2016*. После установки *Golang* необходимо перейти к установке *Git* который позволит нам получить проект с репозитория *github*.

Для установки *Git* переходим на его официальный сайт, после чего на страницу для установки, которая представлена на рисунке 5.2.

				БГТУ 05.00.ПЗ							
	ФИО	Подпись	Дата								
Разраб.	Криштопчик А.Ю.			5 Руководство программиста	Лит.			Лист		Листов	
Пров.	Смелов В.В.							1		8	
Консульт.					74417011, 2020						
Н. контр.	Рыжанкова А.С.										
Утв.	Смелов В.В.										

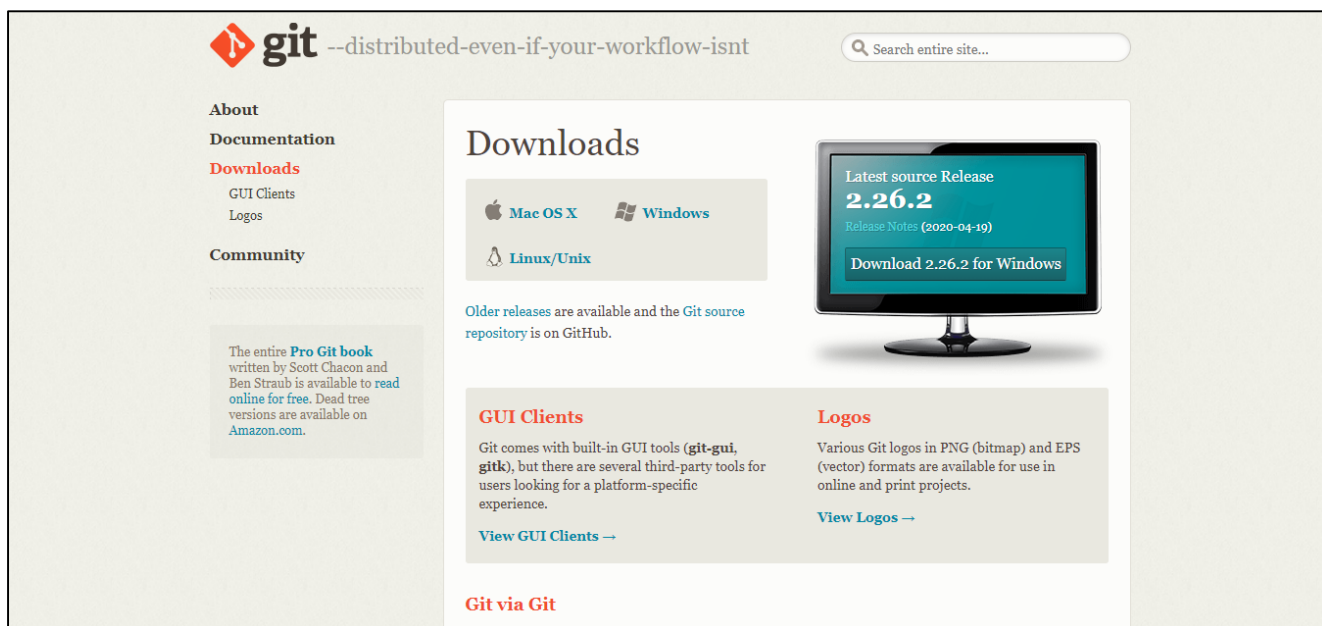


Рисунок 5.2 – Страница для скачивания git

Выбираем скачать для *Windows*, после скачивания нас встречает стандартный установщик, представленный на рисунке 5.3.



Рисунок 5.3 – Страница для скачивания git

После необходимо проверить правильность установки, это возможно сделать несколькими способами, самый простой из них будет, перейдя в консоль и введя команду «*git --version*» мы увидим результат, представленный на рисунке 5.4


```
C:\Users\Администратор>git --version
git version 2.26.2.windows.1
```

Рисунок 5.4 – Страница для скачивания git

После установки *git* можно перейти к скачиванию приложения, для это переходим в консоль и прописываем команду «*git clone https://github.com/Krishtopchik/diplom*», результат выполнения данной команды представлен на рисунке 5.5.

```
Q:\>git clone https://github.com/Krishtopchik/diplom
Cloning into 'diplom'...
remote: Enumerating objects: 13, done.
remote: Counting objects: 100% (13/13), done.
remote: Compressing objects: 100% (12/12), done.
Receiving objects: 100% (837/837), 53.95 MiB | 2.22 MiB/s, done.
Resolving deltas: 100% (432/432), done.
```

Рисунок 5.5 – Страница для скачивания git

Данный результат говорит нам о том, что приложение корректно скопировалось на наш сервер. После этого переходим в папке в которую мы скачивали приложение и при помощи команды «*go run main.go*» проверяем верность установки, результат данной команды приведен на рисунке 5.6.

```
Q:\diplom\go>go run main.go
time="13:09:45.196 26-05-2020" level=info msg=Starting...
time="13:09:45.290 26-05-2020" level=info msg="Loading with config: conf.json"
time="13:09:45.292 26-05-2020" level=info msg="Loading config: &{:8181 web//dist//web}"
time="13:09:45.292 26-05-2020" level=info msg="starting REST server on :8181"
```

Рисунок 5.6 – Запуск серверной части интернет-сервиса

Данный вывод в консоль говорит нам о том, что серверная часть установлена верно она запущена, но для экономии памяти репозиторий был загружен на *github* не полностью, для полной установки необходимо приступить к установке клиентской части интернет-сервиса.

5.2 Установка клиентской части интернет-сервиса

Для разработки клиентских приложений нам необходима платформа *Node.js*. Так же при установке *Node.js* [11] установится *npm* [12]. *Npm* – менеджер пакетов, входящий в состав *Node.js*. Поэтому после установки *Node.js* у нас автоматически будет установлен *npm*. *Npm* необходим нам для установки пакетов используемых в клиентской части интернет-сервиса которую можно скачать с официального сайта, представленного на рисунке 5.7.

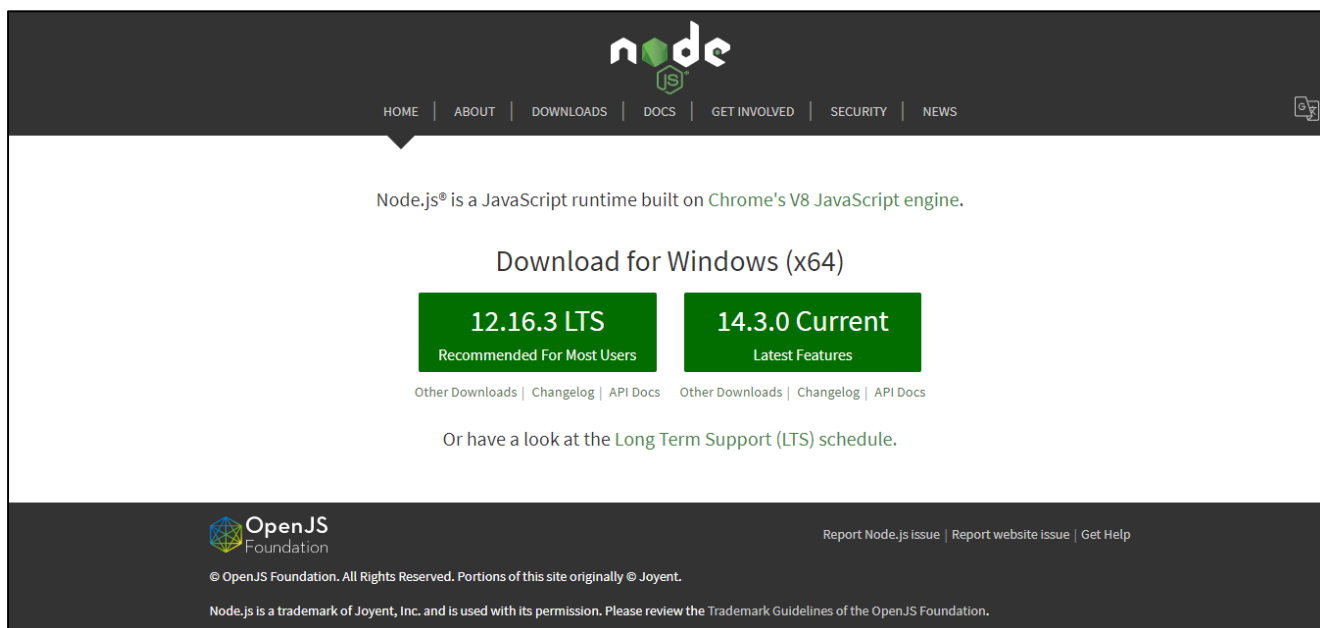


Рисунок 5.7 – Главная страница сайта Node.js

На выбор предоставляться 2 версии: *lts* – стабильная версия, рекомендуемая большинству пользователей и *current*, в которой представлены последний функционал, но возможны возникновение ошибок. Для наших целей достаточно стабильной версии. После установки, которой, для проверки в консоли разработчика необходима ввести 2 команды: «*node -v*» и «*npm -v*». Результат данных команд представлен на рисунке 5.8.

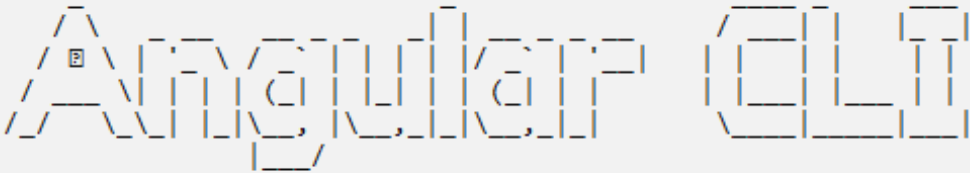
```
C:\Users\Администратор>node -v
v12.16.2

C:\Users\Администратор>npm -v
6.14.4
```

Рисунок 5.8 – Результат выполнения команд

После того как мы удачно установили платформу *Node.js*, и менеджер пакетов *npm* можно переходить к установке *Angular Cli*. Для компиляции приложения мы будем использовать инфраструктуру *Angular CLI*. *Angular CLI* упрощает создание приложения и его компиляцию. *Angular CLI* распространяется как пакет *npm*, который мы уже установили. Для установки *Angular CLI* откроем строку и выполним в ней следующую команду: «*npm install -g @angular/cli*». В результате чего данная команда установит пакет *@angular/cli* в качестве глобального модуля. Для проверки верности установки *Angular Cli* перейдем в командную строку и выполним команду «*ng --version*», результат ее выполнения представлен на рисунке 5.9.

```
C:\Users\Администратор>ng --version
```



```
Angular CLI: 9.1.2
Node: 12.16.2
OS: win32 x64

Angular:
...
Ivy Workspace:
```

Package	Version
@angular-devkit/architect	0.901.2
@angular-devkit/core	9.1.2
@angular-devkit/schematics	9.1.2
@schematics/angular	9.1.2
@schematics/update	0.901.2
rxjs	6.5.4

Рисунок 5.9 – Результат выполнения команды

Abgular Cli вместе с менеджером пакетов *npm* позволит нам развернуть клиентскую часть интернет-сервиса, для этого перейдем в папку *web* и пропишем команду «*npm install*», результат выполнения команды представлен на рисунке 5.10.

```
Q:\it\project\diplom\track_system\web>npm install
npm WARN @angular/material-moment-adapter@9.2.3 requires a peer of @angular/material@9.2.3
npm WARN bootstrap@4.4.1 requires a peer of jquery@1.9.1 - 3 but none is installed. You must
npm WARN bootstrap@4.4.1 requires a peer of popper.js@^1.16.0 but none is installed. You must
npm WARN ngx-mask@9.0.2 requires a peer of @angular/common@^9.1.0 but none is installed. You must
npm WARN ngx-mask@9.0.2 requires a peer of @angular/core@^9.1.0 but none is installed. You must
npm WARN ngx-mask@9.0.2 requires a peer of @angular/forms@^9.1.0 but none is installed. You must
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.1.2 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.1.2: want
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.11 (node_modules\webpack-dev-s
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.11: war
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.11 (node_modules\watchpack\nod
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.11: war

audited 1486 packages in 49.011s

37 packages are looking for funding
  run `npm fund` for details

found 79 vulnerabilities (76 low, 1 moderate, 2 high)
  run `npm audit fix` to fix them, or `npm audit` for details
```

Рисунок 5.10 – Результат установки пакетов

Данная команда установила все необходимые пакеты для работы клиентской части приложения, но это еще не все, наш интернет-сервис работает таким образом, что нам не обязательно запускать клиентскую и серверную часть отдельно, для этого

на серверной части создан обработчик запроса, который позволяет вернуть статические файлы в ответе, чтобы сформировать их нам необходимо выполнить команду «*ng build --prod*», результат которой представлен на рисунке 5.11.

```
Q:\it\project\diplom\track_system\web>ng build --prod
Your global Angular CLI version (9.1.2) is greater than your local
version (9.0.5). The local Angular CLI version is used.

To disable this warning use "ng config -g cli.warnings.versionMismatch false".

chunk {} runtime.689ba4fd6cadb82c1ac2.js (runtime) 1.45 kB [entry] [rendered]
chunk {1} main.d69fca24f12bd29e8be1.js (main) 1.35 MB [initial] [rendered]
chunk {2} polyfills.8c82375d55a050a664f1.js (polyfills) 44.8 kB [initial] [rendered]
chunk {3} polyfills-es5.ba2938ad5fa6672a17f5.js (polyfills-es5) 128 kB [initial] [rendered]
chunk {4} styles.1c877b6a1f93ec049180.css (styles) 223 kB [initial] [rendered]
Date: 2020-05-26T10:29:23.650Z - Hash: ffb0ed9c046b565a9b38 - Time: 182895ms
```

Рисунок 5.11 – Результат сборки клиентской части

В результате *Angular Cli* соберет проект таким образом, из большого количества разработанных нами файлов остался один *html* файл, четыре *JavaScript* файла и один *css* со стилями, которые будут отдаваться клиентам по запросу к нашей серверной части.

После этого мы сможем увидеть наше приложение в браузере, но взаимодействовать с ним еще рано, так как еще негде хранить данные, которые будут использоваться, для этого нам необходимо перейти к установке базы данных.

5.3 Установка базы данных интернет-сервиса

В качестве базы данных у нас выступает *PostgreSQL*, установка которой не займет много времени, для этого необходимо перейти на официальный сайт представленный на рисунке 5.12.

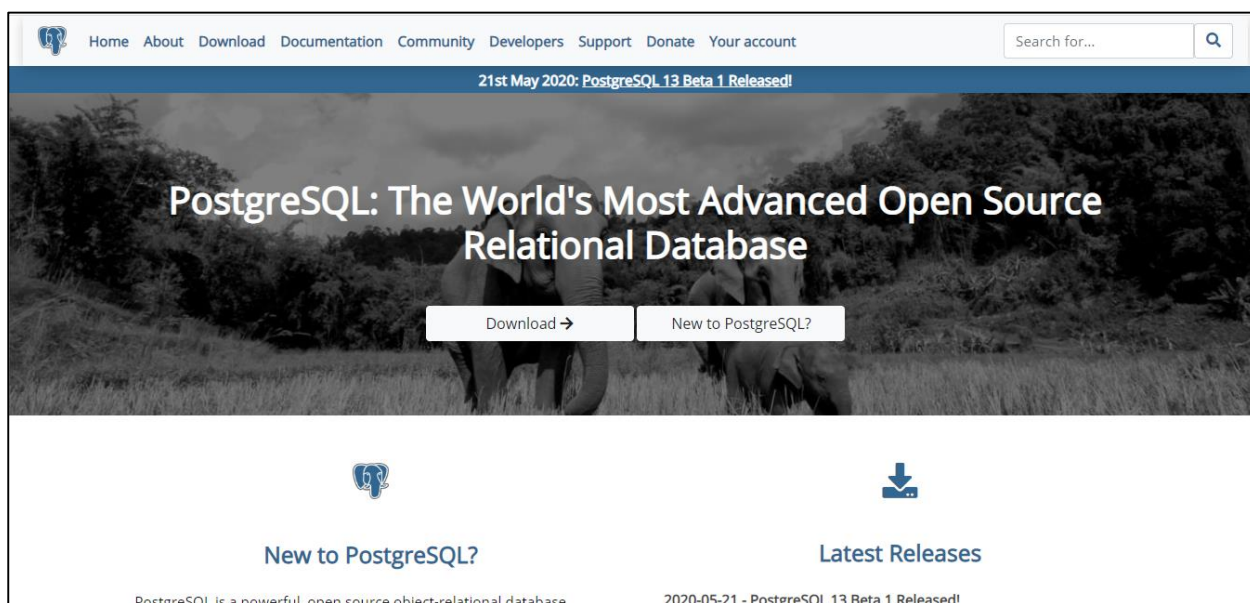


Рисунок 5.12 – Главная страница PostgreSQL

После выбираем пункт *Downlad*, в перечне операционных систем выбираем windows и необходимую нам версию, рассмотренную в пункте 2.2. Во время установки нам предложат создать пользователя базы данных, обязательно запоминаем логин и пароль, так как они понадобятся для соединения нашей серверной части и базы данных. После установки необходимо создать соединение с базой данных для создания таблиц. Для этого я буду использовать среду разработки *Goland*. Для этого переходим к вкладке *Database* и указав необходимые, представленные на рисунке 5.13.

The image shows a configuration window for a database connection. At the top, there's a 'Name' field with the value 'diploms@localhost' and a 'Comment' field. Below these are tabs: 'General', 'Options', 'SSH/SSL', 'Schemas', and 'Advanced'. The 'General' tab is active. It shows 'Connection type: default' and 'Driver: PostgreSQL'. The 'Host' is 'localhost' and 'Port' is '5432'. 'Authentication' is set to 'User & Password'. The 'User' is 'postgres' and the 'Password' is masked as '<hidden>'. The 'Save' dropdown is set to 'Forever'. The 'Database' is 'postgres'. The 'URL' field contains 'jdbc:postgresql://localhost:5432/postgres' with a note 'Overrides settings above'. A 'Test Connection' button is at the bottom.

Рисунок 5.13 – Параметры для подключения к базе данных

Для подключения нам необходимо указать путь к базе данных, способ подключения, в моем случае это имя пользователя и пароль, после чего проверить соединения и, если соединения корректно подключится к базе данных. После подключения мы можем начать манипулировать базой данных при помощи *sql* скриптов. Нам необходимо создать таблицы, описанные в главе 2.3 при помощи скриптов описанных в главе 3.3. Для этого переходим в *Query Console* в *Goland*, куда вставляем необходимые скрипты, после выполнения которых создадим необходимые таблицы в базе данных.

После перейдя по *ip* адресу сервера указав порт, записанный в конфигурационном файле серверной части интернет-сервиса, в браузере откроется стартовая страница интернет-сервиса, представленная на рисунке 5.14.

Рисунок 5.14 – Стартовая страница интернет-сервиса в браузере

После чего можно приступить к полноценному использованию интернет-сервиса.

5.4 Вывод по разделу

1. Рассмотрена установка серверной части интернет-сервиса, для этого нам понадобилось установить *git*, для получения интернет-сервиса с *github*, после чего установить *Golang* для запуска приложения.

2. Рассмотрена установка клиентской части интернет-сервиса которая включила в себя, установку платформы *Node.js*, менеджера пакетов *npm*, для сборки *Angular* проекта понадобилась установка *Angular Cli*, после чего были установлены необходимые пакеты и собрана клиентская часть интернет-сервиса.

3. Рассмотрена установка базы данных интернет-сервиса. Для этого понадобилось скачать *PostgreSql*, после чего подключится и создать необходимые таблицы для интернет-сервиса. Так же был протестирован интернет-сервис после установки всех 3 компонентов.

6 Экономический раздел

6.1 Общая характеристика разрабатываемого программного средства

Темой дипломного проекта является разработка интернет-сервис для учета и контроля выполнения дипломного проектирования в вузе. В связи с тем, что контроль за ходом дипломного проектирования представляет собой отдельную непростую задачу, требующую назначение темы, распределение студентов руководителям дипломного проектирования, комиссии, нормоконтролеров, председателей дипломной комиссии, рецензентов, а также мониторинг процесса выполнения работы студентом появилась необходимость в интернет-сервисе позволяющего автоматизировать учет и контроль дипломного программирования.

Интернет-сервис будет использовано для внутреннего использования кафедрой. Данный раздел служит для определения затрат, произведенных на всех стадиях разработки программного средства.

6.2 Исходные данные и маркетинговый анализ

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

Таблица 6.1 – Исходные данные для расчета

Наименование показателя	Единица измерения	Условные обозначения	Норматив
Численность разработчиков	чел.	Ч_p	1
Норматив дополнительной заработной платы	%	$\text{Н}_{\text{дз}}$	12
Ставка отчислений в Фонд социальной защиты населения	%	$\text{Н}_{\text{фсзн}}$	34
Ставка отчислений в БРУСП «Белгосстрах»	%	$\text{Н}_{\text{бгс}}$	0,4
Цена одного машино-часа	руб.	$\text{С}_{\text{мч}}$	0,06
Норматив прочих затрат	%	$\text{Н}_{\text{пз}}$	10,5
Норматив накладных расходов	%	$\text{Н}_{\text{обп,обх}}$	122
Норматив расходов на сопровождение и адаптацию	%	$\text{Н}_{\text{рса}}$	14
Ставка НДС	%	$\text{Н}_{\text{ндс}}$	20

				БГТУ 06.00.ПЗ						
	ФИО	Подпись	Дата							
Разраб.	Криштопчик А.Ю.			6 Экономический раздел			Лит.	Лист	Листов	
Пров.	Смелов В.В.								1	8
Консульт.	Евлаш А.И.						74417011, 2020			
Н. контр.	Рыжанкова А.С.									
УТВ.	Смелов В.В.									

В ходе проведения маркетингового анализа, была выявлена стоимость разработки интернет-сервиса для учета и контроля выполнения дипломного проектирования в вузе. Средняя цена разработки аналогичного продукта составляет 2000-4000 рублей. Таким образом, общая стоимость разработки данного программного средства, выбранного в качестве базы сравнения составит 3000 руб.

Оценить реальную стоимость сайта-аналога не представляется возможным, т.к. администрация не представляет данную информацию. Однако можно найти информацию о примерной стоимости у разработчиков. Так, разработчики с сайта *inprocess* [19] оценивают разработку уникального сайта [20] в 3000 руб. Уникальный сайт выбран в качестве примера, так как наш интернет-сервис имеет нестандартный функционал который требуют необычных решений.

6.3 Методика обоснования цены

В современных рыночных экономических условиях программное средство (ПС) выступает преимущественно в виде продукции организаций, представляющей собой функционально завершенные и имеющие товарный вид, реализуемые покупателям по рыночным отпускным ценам. Все завершенные разработки являются научно-технической продукцией.

Широкое применение вычислительных технологий требует постоянного обновления и совершенствования ПС. Выбор эффективных проектов ПС связан с их экономической оценкой и расчетом экономического эффекта, который может определяться как у разработчика, так и у пользователя.

У разработчика экономический эффект выступает в виде чистой прибыли от реализации ПС, 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;
- сокращения расходов на оплату машинного времени и других ресурсов на отладку программных средств;
- оптимизации программных средств;
- ускорения ввода в эксплуатацию новых систем;
- улучшения показателей основной деятельности в результате использования передовых программных средств.

Стоимостная оценка программных средств 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 оплату машинного времени;
- прочие прямые затраты;

На основании затрат рассчитывается себестоимость и отпускная цена конечного программного средства.

6.3.1 Объем программного средства

Для общей оценки объема программного средства, функции приложения оцениваются с помощью специальной классификационной таблицы, представленной в приложении, в которой определяется объем каждой функции. Общий объем программного средства V_o , вычисляется как сумма объемов V_i каждой из n его функций формуле 6.1.

$$V_o = \sum_{i=1}^n V_i, \quad (6.1)$$

где V_i – объем i -ой функции ПС, условных машинных команд;
 n – общее число функций.

В ходе рассмотрения классификационной таблицы были выбраны функции, присутствующие в итоговом программном средстве. В таблице 6.2 представлены функции в условных машино-командах.

Таблица 6.2 – Содержание и объем функций в программном средстве

Номер функции	Содержание функции	Объем, условных машино-команд
101	Организация ввода информации	550
102	Контроль, предварительная обработка	420
111	Управление вводом/выводом	1450
202	Взаимодействие между компонентами системы	1600
401	Взаимодействие с базой данных	1250
402	Вспомогательные методы	320
506	Обработка ошибочных и сбойных ситуаций	360
707	Графический вывод результатов	550

Исходя из данных таблицы 6.2, можно рассчитать объем программного средства, разработанного в процессе дипломного проектирования:

$$V_o = 530 + 400 + 1400 + 1690 + 1240 + 340 + 350 + 550 = 6500 \text{ (маш.команд)}.$$

Уточненный объем программного средства V_o' равен произведению объема программного средства V_o на коэффициент изменения скорости обработки информации $K_{ск}$.

$$V_o' = V_o \cdot K_{ск}. \quad (6.2)$$

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

$$V_o' = 6500 \cdot 0,6 = 3900 \text{ (маш.команд)}.$$

6.3.2 Основная заработная плата

Для определения величины основной заработной платы, было проведено исследование величин заработных плат для специалистов в сфере веб-программирования на языке программирования *Golang* и *JavaScript* для разработчиков ПО в вузе. Источником данных служили открытые веб-порталы, различные форумы, официальная отчетность, а также общий средний уровень заработка в сфере информационных технологий. Итогом изучения и анализа полученных данных, стала информация о том, что средняя месячная заработная плата для позиций *junior* составляет 200 рублей.

Проект разрабатывался одним человеком на протяжении двух месяцев. Основная заработная плата рассчитывается по формуле 6.3:

$$C_{\text{оз}} = T_{\text{раз}} \cdot K_{\text{раз}} \cdot C_{\text{зп}}, \quad (6.3)$$

где $C_{\text{оз}}$ – основная заработная плата, руб.;

$T_{\text{раз}}$ – время разработки, месяцев;

$K_{\text{раз}}$ – количество разработчиков, человек;

$C_{\text{зп}}$ – средняя месячная заработная плата.

$$C_{\text{оз}} = 2 \cdot 1 \cdot 200 = 400 \text{ руб.}$$

Основная заработная плата, рассчитана исходя из данных представленных в таблице 6.1.

6.3.3 Дополнительная заработная плата

Законодательство о труде предусматривает наличие выплат, которые определяются по нормативу в процентах к основной заработной плате по формуле 6.4:

$$C_{\text{дз}} = \frac{C_{\text{оз}} \cdot H_{\text{дз}}}{100}, \quad (6.4)$$

где $C_{\text{оз}}$ – основная заработная плата, руб.;

$H_{\text{дз}}$ – норматив дополнительной заработной платы, %.

$$C_{\text{дз}} = 400 \cdot 12 / 100 = 48 \text{ руб.}$$

Исходя из основной заработной платы, а также норматива дополнительной заработной платы, можно рассчитать сумму дополнительной заработной платы:

6.3.4 Отчисления в Фонд социальной защиты населения

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

$$C_{\text{фсзн}} = \frac{(C_{\text{оз}} + C_{\text{дз}}) \cdot N_{\text{фсзн}}}{100}, \quad (6.5)$$

где $C_{\text{оз}}$ – основная заработная плата, руб.;

$C_{\text{дз}}$ – дополнительная заработная плата на конкретное ПС, тыс. руб.;

$N_{\text{фсзн}}$ – норматив отчислений в Фонд социальной защиты населения, %.

Отчисления в БРУСП «Белгосстрах» вычисляются по формуле 6.6:

$$C_{\text{бгс}} = \frac{(C_{\text{оз}} + C_{\text{дз}}) \cdot N_{\text{бгс}}}{100}. \quad (6.6)$$

$$C_{\text{фсзн}} = (400 + 48) \cdot 34 / 100 = 152.32 \text{ руб.}$$

$$C_{\text{бгс}} = (400 + 48) \cdot 0,4 / 100 = 1,79 \text{ руб.}$$

Таким образом, общие отчисления в БРУСП «Белгосстрах» составили 1,79 руб., а в фонд социальной защиты населения – 152,32 руб.

6.3.5 Расходы на материалы

Сумма расходов на материалы $C_{\text{м}}$ определяется как произведение нормы расхода материалов в расчете на сто строк исходного кода $N_{\text{м}}$ на уточненный объем программного средства V_o' .

$$C_{\text{м}} = N_{\text{м}} \cdot \frac{V_o'}{100}. \quad (6.7)$$

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

$$C_{\text{м}} = 0,46 \cdot 3900 / 100 = 17,94 \text{ руб.}$$

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

6.3.6 Расходы на оплату машинного времени

Сумма расходов на оплату машинного времени $C_{\text{мв}}$ определяется как произведение стоимости одного машино-часа $C_{\text{мч}}$ на уточненный объем программного средства V_o' и на норматив расхода машинного времени на отладку ста строк исходного кода $N_{\text{мв}}$.

$$C_{\text{МВ}} = C_{\text{МЧ}} \cdot \frac{V_o'}{100} \cdot H_{\text{МВ}}. \quad (6.8)$$

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

$$C_{\text{МВ}} = 0,06 \cdot 3900 \cdot 15 / 100 = 35,1 \text{ руб.}$$

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

6.3.7 Прочие прямые затраты

Сумма прочих затрат $C_{\text{ПЗ}}$ определяется как произведение основной заработной платы исполнителей на конкретное программное средство $C_{\text{ОЗ}}$ на норматив прочих затрат в целом по организации $H_{\text{ПЗ}}$.

$$C_{\text{ПЗ}} = \frac{C_{\text{ОЗ}} \cdot H_{\text{ПЗ}}}{100}. \quad (6.9)$$

$$C_{\text{ПЗ}} = 400 \cdot 10,5 / 100 = 42 \text{ руб.}$$

6.3.8 Накладные расходы

Сумма накладных расходов $C_{\text{обп,обх}}$ – произведение основной заработной платы исполнителей на конкретное программное средство $C_{\text{ОЗ}}$ на норматив накладных расходов в целом по организации $H_{\text{обп,обх}}$.

$$C_{\text{ПЗ}} = \frac{C_{\text{ОЗ}} \cdot H_{\text{обп, обх}}}{100}. \quad (6.10)$$

Все данные необходимые для вычисления есть, поэтому можно определить сумму накладных расходов:

$$C_{\text{обп,обх}} = 400 \cdot 122 / 100 = 488 \text{ руб.}$$

6.3.9 Сумма расходов на разработку программного средства

Сумма расходов на разработку программного средства C_p 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.

$$C_p = C_{\text{ОЗ}} + C_{\text{ДЗ}} + C_{\text{фсзн}} + C_{\text{бгс}} + C_{\text{М}} + C_{\text{МВ}} + C_{\text{ПЗ}} + C_{\text{обп,обх}} \quad (6.11)$$

$$C_p = 400 + 48 + 152,32 + 1,79 + 17,94 + 35,1 + 42 + 488 = 1185,15 \text{ руб.}$$

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

6.3.10 Расходы на сопровождение и адаптацию

Сумма расходов на сопровождение и адаптацию программного средства $C_{рса}$ определяется как произведение суммы расходов на разработки на норматив расходов на сопровождение и адаптацию $H_{рса}$.

$$C_{рса} = \frac{C_p \cdot H_{рса}}{100}. \quad (6.12)$$

$$C_{рса} = 1185,15 \cdot 14 / 100 = 165,92 \text{ руб.}$$

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

Все проведенные выше расчеты необходимы для вычисления полной себестоимости проекта.

6.3.11 Полная себестоимость

Полная себестоимость $C_{п}$ определяется как сумма двух элементов: суммы расходов на разработку C_p и суммы расходов на сопровождение и адаптацию программного средства $C_{рса}$.

$$C_{п} = C_p + C_{рса}. \quad (6.13)$$

$$C_{п} = 1185,15 + 165,94 = 1351,07 \text{ руб.}$$

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

6.4 Выводы по разделу

Необходимость разработки интернет-сервиса, обусловлена сложностью контроля за ходом дипломного проектирования в университете. В таблице 6.3 и в приложении Е представлены результаты расчетов для основных показателей данного раздела.

Таблица 6.3 – Результаты расчетов

Наименование показателя	Значение
Время разработки, мес.	2
Количество программистов, чел.	1
Зарплата с отчислениями, руб.	602,11
Расходы на материалы, оплату машинного времени, прочие, руб.	95,04
Накладные расходы, руб.	488
Себестоимость разработки программного средства, руб.	1185,15

Окончание таблицы 6.3

Наименование показателя	Значение
Расходы на сопровождение и адаптацию, руб.	165,92
Полная себестоимость, руб.	1351,07
Цена аналога, руб.	3000

Контроль за ходом дипломного проектирования представляет собой отдельную не простую задачу, требующую назначение темы, распределение студентов руководителям дипломных работ, назначение руководителей, комиссии, нормоконтролеров, председателей дипломной комиссии, рецензентов, а также мониторинг процесса выполнения работы студентом.

Чтобы решить данную проблему было разработан интернет-сервис, который повышает эффективность работы заведующего кафедрой и руководителей дипломных проектов.

Разработка программного средства, осуществляемая одним программистом в течении двух месяцев, при заданных условиях обойдется в 1351,07 руб. В то же время приобретение аналога обошлось бы университету примерно в 3000 руб. Разработка данного интернет-сервиса силами работника университета, экономит 1648,93 руб. В результате чего данный проект поможет преподавателям университета, и так же не затребует денежных средств, представляемых другими внешними разработчиками.

Заключение

1. Осуществлена постановка задач, выполнен обзор основных аналогов, рассмотрены их преимущества и недостатки, произведен патентный поиск по теме дипломного проекта и выбор программной платформы и инструментария для разработки.

2. Разработана диаграмма вариантов использования которая включает в себя 2 роли, а также 6 функций, разработана архитектура интернет-сервиса представляющая из себя трехуровневую модель, разработана логическая схема базы данных состоящая из 8 таблиц.

3. Разработана серверная и клиентская часть интернет-сервиса, а также база данных интернет-сервиса. Интернет-сервис разработан на языках *Golang* и *TypeScript* при помощи фреймворка *Angular*, библиотек *AngularMaterial* и *Gorilla*. В качестве базы данных использовался *PostgreSQL*.

4. Протестирован способ просмотра добавления, изменения, удаления дипломов, руководителей дипломного проектирования, нормоконтролеров, рецензентов, председателей, комиссии, приказов в интернет-сервисе. Протестирован способ фильтрации дипломных проектов или работ, а также вывод данных в текстовый формат. Данное тестирование проведено для нескольких ролей. В результате тестирования ошибок выявлено не было что позволяет конечному пользователю не только достаточно просто ориентироваться в интернет-сервисе, но и работать с ним.

5. Описана установка серверной части интернет-части, что включает в себя *golang* и копирования репозитория с *github* при помощи *git*, клиентской части, включающую в себя установку *Node.js*, *npm*, *Angular* и *Angular Cli*. Для базы данных интернет-сервиса потребовалась установка *PostgreSQL*. Так же рассмотрен способ запуска интернет-сервиса.

6. Были рассчитаны денежные затраты и трудозатраты на разработку интернет-сервиса.

				БГТУ 00.00.ПЗ						
	ФИО	Подпись	Дата							
Разраб.	Криштопчик А.Ю.			Заключение	Лит.			Лист	Листов	
Пров.	Смелов В.В.							1	1	
Консульт.					74218011, 2020					
Н. контр.	Рыжанкова А.С.									
Утв.	Смелов В.В.									

Список использованных источников

- 1 PostgreSQL // Википедия [Электронный ресурс]. – 2020. – Режим доступа: <https://ru.wikipedia.org/wiki/PostgreSQL>. – Дата доступа: 01.05.2020.
- 2 Angular // Github [Электронный ресурс]. – 2020. – Режим доступа: <https://github.com/angular/angular>. – Дата доступа: 01.05.2020.
- 3 Golang // Github [Электронный ресурс]. – 2020. – Режим доступа: <https://github.com/golang/go>. – Дата доступа: 01.05.2020.
- 4 USU // usu [Электронный ресурс]. – 2020. – Режим доступа: http://usu.kz/avtomatizatsiya_uchebnogo_protssessa.php. – Дата доступа: 01.05.2020.
- 5 GS-Ведомости: Online // gs-vedomosti [Электронный ресурс]. – 2020. – Режим доступа: <http://gs-vedomosti.ru/about/online.php>. – Дата доступа: 01.05.2020.
- 6 PHP // Википедия [Электронный ресурс]. – 2020. – Режим доступа: <https://ru.wikipedia.org/wiki/PHP>. – Дата доступа: 01.05.2020.
- 7 JavaScript // learn.javascript [Электронный ресурс]. – 2020. – Режим доступа: <https://learn.javascript.ru/intro>. – Дата доступа: 01.05.2020.
- 8 C Sharp // Википедия [Электронный ресурс]. – 2020. – Режим доступа: https://ru.wikipedia.org/wiki/C_S. – Дата доступа: 01.05.2020.
- 9 HTML // Htmlbook [Электронный ресурс]. – 2020. – Режим доступа: <http://htmlbook.ru/html>. – Дата доступа: 01.05.2020.
- 10 CSS // Википедия [Электронный ресурс]. – 2020. – Режим доступа: <https://ru.wikipedia.org/wiki/CSS>. – Дата доступа: 01.05.2020.
- 11 Windows Server 2016 // softmagazin [Электронный ресурс]. – 2020. – Режим доступа: https://www.softmagazin.ru/blog/windows_server_2016_obzor_redaktsiy/. – Дата доступа: 01.05.2020.
- 12 WebStorm // Википедия [Электронный ресурс]. – 2020. – Режим доступа: <https://ru.wikipedia.org/wiki/WebStorm>. – Дата доступа: 01.05.2020.
- 13 TypeScript // Metanit [Электронный ресурс]. – 2020. – Режим доступа: <https://metanit.com/web/typescript/1.1.php>. – Дата доступа: 01.05.2020.
- 14 Angular Material // Material.angular [Электронный ресурс]. – 2020. – Режим доступа: <https://material.angular.io/>. – Дата доступа: 01.05.2020.
- 15 Трехуровневой архитектуре // Википедия [Электронный ресурс]. – 2020. – Режим доступа: http://wikipedia.org/wiki/Multitier_architecture. – Дата доступа: 01.05.2020.
- 16 SQL // Progopedia [Электронный ресурс]. – 2020. – Режим доступа: <http://progopedia.ru/language/sql/>. – Дата доступа: 01.05.2020.
- 17 Google Chrome // Википедия [Электронный ресурс]. – 2020. – Режим доступа: https://ru.wikipedia.org/wiki/Google_Chrome. – Дата доступа: 01.05.2020.

				БГТУ 00.00.ПЗ			
	ФИО	Подпись	Дата				
Разраб.	Криштопчик А.Ю.			Список использованных источников			
Пров.	Смелов В.В.						
Консульт.							
Н. контр.	Рыжанкова А.С.						
Утв.	Смелов В.В.						
				Лит.	Лист	Листов	
						1	2
				74218011, 2020			

18 HTTP // Habr [Электронный ресурс]. – 2020. – Режим доступа: <https://habr.com/ru/post/215117/>. – Дата доступа: 01.05.2020.

19 Создание сайта // Inprocess [Электронный ресурс]. – 2020. – Режим доступа: <https://inprocess.by/>. – Дата доступа: 01.05.2020.

20 Уникального сайта // scoomg [Электронный ресурс]. – 2020. – Режим доступа: https://www.divier.ru/stati/unikalnyy_dizayn_sayta/. – Дата доступа: 01.05.2020.

ПРИЛОЖЕНИЕ А
Диаграмма вариантов использования

ПРИЛОЖЕНИЕ Б
Общая схема взаимодействия компонентов

ПРИЛОЖЕНИЕ В
Логическая схема базы данных

ПРИЛОЖЕНИЕ Г

Листинги кода серверной части

а) Листинг пакета main

```
package main
import (
    "./service"
    "flag"
    log "github.com/sirupsen/logrus"
)
func main() {
    defer func() {
        if err := recover(); err != nil {
            log.Error(err)
        }
    }()
    err := service.InitLog()
    if err != nil {
        log.Error("Cannot init logs: ", err)
    }
    log.Info("Starting...")
    var cnf = flag.String("c", "conf.json", "Config file name (in a
current dir)")
    flag.Parse()
    log.Info("Loading with config: ", *cnf)
    service.InitConfiguration(*cnf, service.Conf)
    service.RunRest()
}
```

б) Листинг пакета service

```
package service
import (
    "../db"
    "../models"
    "bytes"
    "encoding/json"
    "fmt"
    "github.com/gorilla/handlers"
    "github.com/gorilla/mux"
    log "github.com/sirupsen/logrus"
    "io/ioutil"
    "net/http"
    "os"
    "strconv"
    "time"
)
var Conf = &Config{}
func RunRest() {
    r := mux.NewRouter()
    r.HandleFunc("/api/diploms", getDiploms).Methods("GET")
    r.HandleFunc("/api/diploms/{id:[0-9]+}", getDiplom).Meth-
ods("GET")
}
```

```

    r.HandleFunc("/api/diploms/{id:[0-9]+}", deleteDiplom).Methods("DELETE")
    r.HandleFunc("/api/chairmans", getChairmans).Methods("GET")
    r.HandleFunc("/api/chairmans", createChairmans).Methods("POST")
    r.HandleFunc("/api/chairmans/{id:[0-9]+}", deleteChairmans).Methods("DELETE")
    r.HandleFunc("/api/chairmans", updateChairmans).Methods("PUT")
    r.HandleFunc("/api/commissions", getCommissions).Methods("GET")
    r.HandleFunc("/api/commissions", createCommissions).Methods("POST")
    r.HandleFunc("/api/commissions/{id:[0-9]+}", deleteCommissions).Methods("DELETE")
    r.HandleFunc("/api/commissions", updateCommissions).Methods("PUT")
    r.HandleFunc("/api/diplomorders", getDiplomorders).Methods("GET")
    r.HandleFunc("/api/diplomorders", createDiplomorders).Methods("POST")
    r.HandleFunc("/api/diplomorders/{id:[0-9]+}", deleteDiplomorders).Methods("DELETE")
    r.HandleFunc("/api/diplomorders", updateDiplomorders).Methods("PUT")
    r.HandleFunc("/api/normcontrollers", getNormcontrollers).Methods("GET")
    r.HandleFunc("/api/normcontrollers", createNormcontrollers).Methods("POST")
    r.HandleFunc("/api/normcontrollers/{id:[0-9]+}", deleteNormcontrollers).Methods("DELETE")
    r.HandleFunc("/api/normcontrollers", updateNormcontrollers).Methods("PUT")
    r.HandleFunc("/api/pms", getPms).Methods("GET")
    r.HandleFunc("/api/pms", createPm).Methods("POST")
    r.HandleFunc("/api/pms/{id:[0-9]+}", deletePm).Methods("DELETE")
    r.HandleFunc("/api/pms", updatePm).Methods("PUT")
    r.HandleFunc("/api/reviewers", getReviewers).Methods("GET")
    r.HandleFunc("/api/reviewers", createReviewers).Methods("POST")
    r.HandleFunc("/api/reviewers/{id:[0-9]+}", deleteReviewers).Methods("DELETE")
    r.HandleFunc("/api/reviewers", updateReviewers).Methods("PUT")
    r.HandleFunc("/api/specialtys", getSpecialtys).Methods("GET")
    r.HandleFunc("/api/specialtys", createSpecialtys).Methods("POST")
    r.HandleFunc("/api/specialtys", updateSpecialtys).Methods("PUT")
    r.HandleFunc("/api/specialtys/{id:[0-9]+}", deleteSpecialtys).Methods("DELETE")
    r.HandleFunc("/api/diploms", createDiplom).Methods("POST")
    r.HandleFunc("/api/diploms", updateDiplom).Methods("PUT")
    r.HandleFunc("/api/doc", createDoc).Methods("POST")
    r.PathPrefix("/").Handler(http.FileServer(http.Dir(Config.StaticPath)))
    log.Printf("starting REST server on %s", Config.ListenPort)
    allowedHeaders := handlers.AllowedHeaders([]string{"X-Requested-With", "Content-Type", "Authorization"})
    allowedOrigins := handlers.AllowedOrigins([]string{"*"})
    allowedMethods := handlers.AllowedMethods([]string{"GET", "POST", "PUT", "DELETE", "OPTIONS"})

```

```

    err := http.ListenAndServe(
        Conf.ListenPort,
        handlers.CORS(allowedHeaders, allowedOrigins, allowedMeth-
ods)(r))
    if err != nil {
        log.Fatal(err)
    }
}
func getDiploms(w http.ResponseWriter, r *http.Request) {
    w.Header().Set("Content-Type", "application/json")
    diplom, err := db.GetAllDiploms()
    if err != nil {
        log.Error(err)
        w.WriteHeader(http.StatusInternalServerError)
        return
    }
    json.NewEncoder(w).Encode(diplom)
}
func getDiplom(w http.ResponseWriter, r *http.Request) {
    w.Header().Set("Content-Type", "application/json")
    idStr, ok := mux.Vars(r)["id"]
    if !ok || len(idStr) == 0 {
        w.WriteHeader(http.StatusBadRequest)
        return
    }
    id, err := strconv.Atoi(idStr)
    if err != nil {
        log.Error(err)
        w.WriteHeader(http.StatusBadRequest)
        return
    }
    diplom, err := db.GetDiplom(id)
    if err != nil {
        log.Error(err)
        w.WriteHeader(http.StatusInternalServerError)
        return
    }
    json.NewEncoder(w).Encode(diplom)
}
func deleteDiplom(w http.ResponseWriter, r *http.Request) {
    w.Header().Set("Content-Type", "application/json")
    idStr, ok := mux.Vars(r)["id"]
    if !ok || len(idStr) == 0 {
        w.WriteHeader(http.StatusBadRequest)
        return
    }
    id, err := strconv.Atoi(idStr)
    if err != nil {
        log.Error(err)
        w.WriteHeader(http.StatusBadRequest)
        return
    }
    err = db.DeleteDiplom(id)
    if err != nil {

```

```

        log.Error(err)
        w.WriteHeader(http.StatusInternalServerError)
        return
    }
    json.NewEncoder(w).Encode(err)
}
func getChairmans(w http.ResponseWriter, r *http.Request) {
    w.Header().Set("Content-Type", "application/json")
    chairman, err := db.GetChairman()
    if err != nil {
        log.Error(err)
        w.WriteHeader(http.StatusInternalServerError)
        return
    }
    json.NewEncoder(w).Encode(chairman)
}
func createChairmans(w http.ResponseWriter, r *http.Request) {
    var err error
    defer func() {
        if err != nil {
            log.Error(err)
            w.WriteHeader(http.StatusInternalServerError)
            return
        }
    }()
    body := json.NewDecoder(r.Body)
    var teacherModel models.Teacher
    err = body.Decode(&teacherModel)
    if err != nil {
        return
    }
    teacher, err := db.InsertChairman(teacherModel)
    if err != nil {
        return
    }
    err = json.NewEncoder(w).Encode(teacher)
    if err != nil {
        return
    }
}
func deleteChairmans(w http.ResponseWriter, r *http.Request) {
    w.Header().Set("Content-Type", "application/json")
    idStr, ok := mux.Vars(r)["id"]
    if !ok || len(idStr) == 0 {
        w.WriteHeader(http.StatusBadRequest)
        return
    }
    id, err := strconv.Atoi(idStr)
    if err != nil {
        log.Error(err)
        w.WriteHeader(http.StatusBadRequest)
        return
    }
    err = db.DeleteChairman(id)
}

```

```

    if err != nil {
        log.Error(err)
        w.WriteHeader(http.StatusInternalServerError)
        return
    }
    json.NewEncoder(w).Encode(err)
}

func updateChairmans(w http.ResponseWriter, r *http.Request) {
    var err error
    defer func() {
        if err != nil {
            log.Error(err)
            w.WriteHeader(http.StatusInternalServerError)
            return
        }
    }()
    body := json.NewDecoder(r.Body)
    var chairmanModel models.Teacher
    err = body.Decode(&chairmanModel)
    if err != nil {
        return
    }
    teacher, err := db.UpdateChairman(chairmanModel)
    if err != nil {
        return
    }
    err = json.NewEncoder(w).Encode(teacher)
    if err != nil {
        return
    }
}

func getCommissions(w http.ResponseWriter, r *http.Request) {
    w.Header().Set("Content-Type", "application/json")
    commission, err := db.GetCommissionn()
    if err != nil {
        log.Error(err)
        w.WriteHeader(http.StatusInternalServerError)
        return
    }
    json.NewEncoder(w).Encode(commission)
}

func createCommissions(w http.ResponseWriter, r *http.Request) {
    var err error
    defer func() {
        if err != nil {
            log.Error(err)
            w.WriteHeader(http.StatusInternalServerError)
            return
        }
    }()
    body := json.NewDecoder(r.Body)
    var teacherModel models.Teacher
    err = body.Decode(&teacherModel)
    if err != nil {

```



```

        return
    }
    teacher, err := db.InsertCommission(teacherModel)
    if err != nil {
        return
    }
    err = json.NewEncoder(w).Encode(teacher)
    if err != nil {
        return
    }
}

func deleteCommissions(w http.ResponseWriter, r *http.Request) {
    w.Header().Set("Content-Type", "application/json")
    idStr, ok := mux.Vars(r)["id"]
    if !ok || len(idStr) == 0 {
        w.WriteHeader(http.StatusBadRequest)
        return
    }
    id, err := strconv.Atoi(idStr)
    if err != nil {
        log.Error(err)
        w.WriteHeader(http.StatusBadRequest)
        return
    }
    err = db.DeleteCommissionn(id)
    if err != nil {
        log.Error(err)
        w.WriteHeader(http.StatusInternalServerError)
        return
    }
    json.NewEncoder(w).Encode(err)
}

func updateCommissions(w http.ResponseWriter, r *http.Request) {
    var err error
    defer func() {
        if err != nil {
            log.Error(err)
            w.WriteHeader(http.StatusInternalServerError)
            return
        }
    }()
    body := json.NewDecoder(r.Body)
    var teacherModel models.Teacher
    err = body.Decode(&teacherModel)
    if err != nil {
        return
    }
    teacher, err := db.UpdateCommissionn(teacherModel)
    if err != nil {
        return
    }
    err = json.NewEncoder(w).Encode(teacher)
    if err != nil {
        return
    }
}

```

```

    }
}
func getDiplomorders(w http.ResponseWriter, r *http.Request) {
    w.Header().Set("Content-Type", "application/json")
    diplomorders, err := db.GetDiplomOrder()
    if err != nil {
        log.Error(err)
        w.WriteHeader(http.StatusInternalServerError)
        return
    }
    json.NewEncoder(w).Encode(diplomorders)
}
func createDiplomorders(w http.ResponseWriter, r *http.Request) {
    var err error
    defer func() {
        if err != nil {
            log.Error(err)
            w.WriteHeader(http.StatusInternalServerError)
            return
        }
    }()
    body := json.NewDecoder(r.Body)
    var diplomOrderModel models.DiplomOrder
    err = body.Decode(&diplomOrderModel)
    if err != nil {
        return
    }
    diplomOrder, err := db.InsertDiplomOrder(diplomOrderModel)
    if err != nil {
        return
    }
    err = json.NewEncoder(w).Encode(diplomOrder)
    if err != nil {
        return
    }
}
func deleteDiplomorders(w http.ResponseWriter, r *http.Request) {
    w.Header().Set("Content-Type", "application/json")
    idStr, ok := mux.Vars(r)["id"]
    if !ok || len(idStr) == 0 {
        w.WriteHeader(http.StatusBadRequest)
        return
    }
    id, err := strconv.Atoi(idStr)
    if err != nil {
        log.Error(err)
        w.WriteHeader(http.StatusBadRequest)
        return
    }
    err = db.DeleteDiplomOrder(id)
    if err != nil {
        log.Error(err)
        w.WriteHeader(http.StatusInternalServerError)
        return
    }
}

```

```

    }
    json.NewEncoder(w).Encode(err)
}
func updateDiplomorders(w http.ResponseWriter, r *http.Request) {
    var err error
    defer func() {
        if err != nil {
            log.Error(err)
            w.WriteHeader(http.StatusInternalServerError)
            return
        }
    }()
    body := json.NewDecoder(r.Body)
    var diplomOrderModel models.DiplomOrder
    err = body.Decode(&diplomOrderModel)
    if err != nil {
        return
    }
    diplomOrder, err := db.UpdateDiplomOrder(diplomOrderModel)
    if err != nil {
        return
    }
    err = json.NewEncoder(w).Encode(diplomOrder)
    if err != nil {
        return
    }
}
func getNormcontrollers(w http.ResponseWriter, r *http.Request) {
    w.Header().Set("Content-Type", "application/json")
    normcontrollers, err := db.GetNormcontroller()
    if err != nil {
        log.Error(err)
        w.WriteHeader(http.StatusInternalServerError)
        return
    }
    json.NewEncoder(w).Encode(normcontrollers)
}
func createNormcontrollers(w http.ResponseWriter, r *http.Request) {
    var err error
    defer func() {
        if err != nil {
            log.Error(err)
            w.WriteHeader(http.StatusInternalServerError)
            return
        }
    }()
    body := json.NewDecoder(r.Body)
    var teacherModel models.Teacher
    err = body.Decode(&teacherModel)
    if err != nil {
        return
    }
    teacher, err := db.InsertNormcontroller(teacherModel)
    if err != nil {

```

```

        return
    }
    err = json.NewEncoder(w).Encode(teacher)
    if err != nil {
        return
    }
}

func deleteNormcontrollers(w http.ResponseWriter, r *http.Request) {
    w.Header().Set("Content-Type", "application/json")
    idStr, ok := mux.Vars(r)["id"]
    if !ok || len(idStr) == 0 {
        w.WriteHeader(http.StatusBadRequest)
        return
    }
    id, err := strconv.Atoi(idStr)
    if err != nil {
        log.Error(err)
        w.WriteHeader(http.StatusBadRequest)
        return
    }
    err = db.DeleteNormcontroller(id)
    if err != nil {
        log.Error(err)
        w.WriteHeader(http.StatusInternalServerError)
        return
    }
    json.NewEncoder(w).Encode(err)
}

func updateNormcontrollers(w http.ResponseWriter, r *http.Request) {
    var err error
    defer func() {
        if err != nil {
            log.Error(err)
            w.WriteHeader(http.StatusInternalServerError)
            return
        }
    }()
    body := json.NewDecoder(r.Body)
    var teacherModel models.Teacher
    err = body.Decode(&teacherModel)
    if err != nil {
        return
    }
    teacher, err := db.UpdateNormcontroller(teacherModel)
    if err != nil {
        return
    }
    err = json.NewEncoder(w).Encode(teacher)
    if err != nil {
        return
    }
}

func getPms(w http.ResponseWriter, r *http.Request) {
    w.Header().Set("Content-Type", "application/json")

```

```

pms, err := db.GetPm()
if err != nil {
    log.Error(err)
    w.WriteHeader(http.StatusInternalServerError)
    return
}
json.NewEncoder(w).Encode(pms)
}

```

в) Листинг пакета db

```

package db
import (
    "../models"
    "database/sql"
    "fmt"
    "github.com/lib/pq"
    "sort"
)
func createConnectin() *sql.DB {
    connStr := "user=postgres password=123123 dbname=postgres sslmode=disable"
    db, err := sql.Open("postgres", connStr)
    if err != nil {
        panic(err)
    }
    return db
}
func InsertDiplom(diplom models.Diplom) (models.Diplom, error) {
    db := createConnectin()
    defer db.Close()
    err := db.QueryRow("insert into diplom (fio, topic, completion, score, deadline, queuenumber, pmid, normcontrollerid, reviewerid, chairmanid, diplomorderid, specialtyid, commissionid, execution, type, commissioncomment) values ($1, $2, $3, $4, $5, $6, $7, $8, $9, $10, $11, $12, $13, $14, $15, $16) returning id",
        diplom.Fio, diplom.Topic, diplom.Completion, diplom.Score, diplom.Deadline, diplom.Queue-
        number, diplom.PmId, diplom.NormcontrollerId, diplom.ReviewerId, diplom.ChairmanId, diplom.DiplomorderId, diplom.Special-
        tyId, diplom.CommissionId, diplom.Execution, diplom.Type, diplom.Com-
        missionComment).Scan(&diplom.Id)
    if err != nil{
        panic(err)
    }
    return diplom, err
}
func DeleteDiplom(id int) error {
    db := createConnectin()
    defer db.Close()
    _, err := db.Exec("delete from diplom values where id = $1", id)
    if err != nil{
        panic(err)
    }
}

```

```

    }
    return err
}

func UpdateDiplom(diplom models.Diplom) (models.Diplom, error) {
    db := createConnectin()
    defer db.Close()
    if _, err := db.Exec("update diplom set Fio = $2, Topic = $3, Completion = $4, Score = $5, Deadline = $6, Queue-
number = $7, PmId = $8, NormcontrollerId = $9, Review-
erId = $10, ChairmanId = $11, DiplomorderId = $12, Special-
tyId = $13, CommissionId = $14, execution = $15, type = $16, commis-
sioncomment = $17, time = $18 where id = $1",
        diplom.Id, diplom.Fio,
        diplom.Topic, diplom.Completion,
        diplom.Score, diplom.Deadline,
        diplom.Queuenumber, diplom.PmId,
        diplom.NormcontrollerId, diplom.ReviewerId,
        diplom.ChairmanId, diplom.DiplomorderId,
        diplom.SpecialtyId, diplom.CommissionId,
        diplom.Execution, diplom.Type, diplom.CommissionCom-
ment, diplom.Time);
    err != nil {
        return diplom, err
    }
    return diplom, nil
}

func GetAllDiploms() ([]models.Diplom, error){
    db := createConnectin()
    defer db.Close()
    rows, err := db.Query("select * from diplom order by dead-
line, queue number")
    if err != nil {
        panic(err)
    }
    defer rows.Close()
    diplomAll := []models.Diplom{}
    for rows.Next(){
        p := models.Diplom{}
        err := rows.Scan(&p.Id, &p.Fio, &p.Topic, &p.Comple-
tion, &p.Score, &p.Queuenumber, &p.Deadline, &p.PmId, &p.Normcontrol-
lerId, &p.ReviewerId, &p.ChairmanId, &p.DiplomorderId, &p.Special-
tyId, &p.CommissionId, &p.Execution, &p.Type, &p.CommissionCom-
ment, &p.Time)
        if err != nil{
            fmt.Println(err)
            continue
        }
        diplomAll = append(diplomAll, p)
    }
    return diplomAll, err
}

func GetDiplom(id int) (models.Diplom, error){
    db := createConnectin()

```

```

    defer db.Close()
    p := models.Diplom{}
    err := db.QueryRow("select * from diplom where id = $1 limit 1", id).Scan(&p.Id, &p.Fio, &p.Topic, &p.Completion, &p.Score, &p.QueueNumber, &p.Deadline, &p.PmId, &p.NormcontrollerId, &p.ReviewerId, &p.ChairmanId, &p.DiplomorderId, &p.SpecialtyId, &p.CommissionId, &p.Execution, &p.Type, &p.CommissionComment, &p.Time)
    return p, err
}

func InsertChairman(chairman models.Teacher) (models.Teacher, error) {
    db := createConnectin()
    defer db.Close()
    err := db.QueryRow("insert into chairman (fio) values ($1) returning id",
        chairman.Fio).Scan(&chairman.Id)
    if err != nil {
        panic(err)
    }
    return chairman, err
}

func GetChairman() ([]models.Teacher, error) {
    db := createConnectin()
    defer db.Close()
    rows, err := db.Query("select * from chairman")
    if err != nil {
        panic(err)
    }
    defer rows.Close()
    chairmanAll := []models.Teacher{}
    for rows.Next() {
        p := models.Teacher{}
        err := rows.Scan(&p.Id, &p.Fio)
        if err != nil {
            fmt.Println(err)
            continue
        }
        chairmanAll = append(chairmanAll, p)
    }
    sort.Slice(chairmanAll, func(i, j int) bool {
        return chairmanAll[i].Id < chairmanAll[j].Id
    })
    return chairmanAll, err
}

func DeleteChairman(id int) error {
    db := createConnectin()
    defer db.Close()
    _, err := db.Exec("delete from chairman values where id = $1", id)
    if err != nil {
        panic(err)
    }
    return err
}

```

```

}
func UpdateChairman(chairman models.Teacher) (models.Teacher, error) {
    db := createConnectin()
    defer db.Close()
    if _, err := db.Exec("update chairman set Fio = $2 where id = $1",
        chairman.Id, chairman.Fio);
        err != nil {
            return chairman, err
        }
    return chairman, nil
}
func InsertCommission(commission models.Teacher) (models.Teacher, error) {
    db := createConnectin()
    defer db.Close()
    err := db.QueryRow("insert into commission (fio) values ($1) returning id",
        commission.Fio).Scan(&commission.Id)
    if err != nil {
        panic(err)
    }
    return commission, err
}
func GetCommissionn() ([]models.Teacher, error) {
    db := createConnectin()
    defer db.Close()
    rows, err := db.Query("select * from commission")
    if err != nil {
        panic(err)
    }
    defer rows.Close()
    commissionAll := []models.Teacher{}
    for rows.Next() {
        p := models.Teacher{}
        err := rows.Scan(&p.Id, &p.Fio)
        if err != nil {
            fmt.Println(err)
            continue
        }
        commissionAll = append(commissionAll, p)
    }
    sort.Slice(commissionAll, func(i, j int) bool {
        return commissionAll[i].Id < commissionAll[j].Id
    })
    return commissionAll, err
}

```


ПРИЛОЖЕНИЕ Д

Листинги кода клиентской части

```

import { Component } from '@angular/core';
import { Router } from '@angular/router';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.scss']
})
export class AppComponent {
  tab = 'diplom';
  constructor(
    private router: Router,
  ) {}
  changeTab($event) {
    this.tab = $event.target.getAttribute('rel');
    this.router.navigate([this.tab]);
  }
}
import { Component, OnInit } from '@angular/core';
import { TeacherModel } from '../../common/models/teacher.model';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { DiplomService } from '../../common/services/diplom.service';
import { ToastrService } from 'ngx-toastr';
import { ConfirmDialogComponent } from '../dialogs/confirm-dialog/confirm-dialog.component';
import { MatDialog } from '@angular/material/dialog';
import { DiplomDataService } from "../../common/services/diplom-data.service";
@Component({
  selector: 'app-diplom-chairman',
  templateUrl: './diplom-chairman.component.html',
  styleUrls: ['./diplom-chairman.component.scss']
})
export class DiplomChairmanComponent implements OnInit {
  chairmanList: TeacherModel[];
  chairmanForm: FormGroup;
  buttonTitleAdd = true;
  tab = 'add';
  constructor(
    private diplomService: DiplomService,
    private fb: FormBuilder,
    private toastr: ToastrService,
    public dialog: MatDialog,
    private diplomDataService: DiplomDataService
  ) {}
  ngOnInit(): void {
    this.getPmList();
    this.formInit();
  }
  private getPmList() {
    this.diplomService.getChairmans().subscribe(res => {
      this.chairmanList = res;
    });
  }
}

```

```

    });
}
private formInit() {
    this.chairmanForm = this.fb.group({
        Id: [0],
        Fio: ['', Validators.required],
    });
}
onSubmit() {
    const chairman: TeacherModel = this.chairmanForm.getRawValue();
    if (this.buttonTitleAdd) {
        this.diplomService.createChairman(chairman).subscribe(res => {
            this.toastr.success('Добавлен');
            this.getPmList();
            this.diplomDataService.changeChairman = true;
        });
    } else {
        this.diplomService.updateChairman(chairman).subscribe(res => {
            this.toastr.success('Обновлен');
            this.getPmList();
            this.diplomDataService.changeChairman = true;
        });
    }
    this.formInit();
    this.buttonTitleAdd = true;
    this.tab = 'add';
}
deleteChairman(id: number) {
    const dialogRef = this.dialog.open(ConfirmDialogComponent, {
        width: '250px',
    });
    dialogRef.afterClosed().subscribe(result => {
        if (result) {
            this.diplomService.deleteChairman(id).subscribe(res => {
                this.toastr.success('Удален');
                this.getPmList();
                this.diplomDataService.changeChairman = true;
            });
        }
        this.formInit();
        this.buttonTitleAdd = true;
        this.tab = 'add';
    });
}
changeChairman(id: number) {
    const item = this.chairmanList.find(el => el.Id === id);
    this.chairmanForm.patchValue({
        ...item
    });
    this.buttonTitleAdd = false;
    this.tab = 'change';
}
changeTab(ev) {
    this.formInit();
}

```

```

        this.buttonTitleAdd = true;
        this.tab = 'add';
    }
}

import {Component, DoCheck, OnInit, ViewChild} from '@angular/core';
import {DiplomService} from '../common/services/diplom.service';
import {DiplomModel} from '../common/models/diplom.model';
import {Observable, zip} from 'rxjs';
import {DiplomDataService} from '../common/services/diplom-
data.service';
import {DiplomInfoModel} from '../common/models/diplomInfo.model';
import {SpecialytyModel} from "../common/models/specialyty.model";
@Component({
    selector: 'app-diplom',
    templateUrl: './diplom.component.html',
    styleUrls: ['./diplom.component.scss']
})
export class DiplomComponent implements OnInit, DoCheck {
    isDiplomSelect = false;
    selectDiplomId: number;
    diplomsList: DiplomModel[];
    infoAboutDiplom: DiplomInfoModel;
    load = true;
    constructor(
        private diplomService: DiplomService,
        private diplomDataService: DiplomDataService,
    ) {}
    ngOnInit() {
        const filter = localStorage.getItem('filter');
        if (filter) {
            this.getDiplomsListAndFilter();
        } else {
            this.getDiplomsList();
        }
    }
    zip(
        this.getPmList(),
        this.getNormcontrollerList(),
        this.getReviewerList(),
        this.getChairmanList(),
        this.getDiplomorderList(),
        this.getSpecialtyList(),
        this.getCommissionList(),
    ).subscribe([pm, normcontroller, reviewer, chairman, diplo-
morder, specialty, commission]) => {
        this.infoAboutDiplom = {
            pmList: pm,
            normcontrollerList: normcontroller,
            reviewerList: reviewer,
            chairmanList: chairman,
            diplomorderList: diplomorder,
            specialtyList: specialty,
            commissionList: commission,
        };
    };
}

```

```

        this.load = false;
    });
}
ngDoCheck() {
    this.selectDiplomId = this.diplomDataService.selectDiplomId;
    this.isDiplomSelect = this.diplomDataService.isDiplomSelect;
    if (this.diplomDataService.isDiplomsUpdate) {
        this.getDiplomsList();
    }
    if (this.diplomDataService.diplomsFilter) {
        this.getDiplomsListAndFilter();
    }
    if (this.diplomDataService.changePm) {
        this.getPmList().subscribe(res => {
            this.infoAboutDiplom.p mList = res;
            this.diplomDataService.changePm = false;
        });
    }
    if (this.diplomDataService.changeReviewer) {
        this.getReviewerList().subscribe(res => {
            this.infoAboutDiplom.reviewerList = res;
            this.diplomDataService.changeReviewer = false;
        });
    }
    if (this.diplomDataService.changeChairman) {
        this.getChairmanList().subscribe(res => {
            this.infoAboutDiplom.chairmanList = res;
            this.diplomDataService.changeChairman = false;
        });
    }
    if (this.diplomDataService.changeNormoconntroller) {
        this.getNormcontrollerList().subscribe(res => {
            this.infoAboutDiplom.normcontrollerList = res;
            this.diplomDataService.changeNormoconntroller = false;
        });
    }
    if (this.diplomDataService.changeOrder) {
        this.getDiplomorderList().subscribe(res => {
            this.infoAboutDiplom.diplomorderList = res;
            this.diplomDataService.changeOrder = false;
        });
    }
    if (this.diplomDataService.changeCommission) {
        this.getCommissionList().subscribe(res => {
            this.infoAboutDiplom.commissionList = res;
            this.diplomDataService.changeCommission = false;
        });
    }
    if (this.diplomDataService.changeSpecialty) {
        this.getSpecialtyList().subscribe(res => {
            this.infoAboutDiplom.specialtyList = res;
            this.diplomDataService.changeSpecialty = false;
        });
    }
}

```

```

    this.diplomDataService.diploms = this.diplomsList;
  }
  private getDiplomsList() {
    this.diplomService.getAllDiploms().subscribe(res => {
      this.diplomsList = res;
      this.diplomDataService.isDiplomsUpdate = false;
    });
  }
  private getDiplomsListAndFilter() {
    this.diplomService.getAllDiploms().subscribe(res => {
      const filter = JSON.parse(localStorage.getItem('filter'));
      Object.keys(filter).forEach(key => {
        if (new RegExp(filter[key]).test(key) && key !== 'Execution' && key !== 'Deadline') {
          filter[key] = +filter[key];
        }
      });
      res = res.filter(el => {
        if (key === 'Execution' || key === 'Deadline') {
          if (this.checkDate(filter[key], el[key])) {
            return el;
          }
        }
        return el[key] === filter[key];
      });
      this.diplomsList = res;
      this.diplomDataService.diplomsFilter = false;
    });
  }
  private checkDate(filter, el) {
    const day = (+filter.substring(8, 10) + 1) < 10 ? `0${+filter.substring(8, 10) + 1}` : (+filter.substring(8, 10) + 1).toString();
    return filter.substring(0, 4) === el.substring(0, 4) && filter.substring(5, 7) === el.substring(5, 7) && day === el.substring(8, 10);
  }
  private getPmList() {
    return this.diplomService.getPms();
  }
  private getNormcontrollerList() {
    return this.diplomService.getNormcontrollers();
  }
  private getReviewerList() {
    return this.diplomService.getReviewers();
  }
  private getChairmanList() {
    return this.diplomService.getChairmans();
  }
  private getDiplomorderList() {
    return this.diplomService.getDiplomorders();
  }
  private getSpecialtyList(): Observable<SpecialtyModel[]> {
    return this.diplomService.getSpecialtys();
  }

```

```

    }
    private getCommissionList() {
        return this.diplomService.getCommissions();
    }
}
import { Component, OnInit } from '@angular/core';
import { TeacherModel } from '../../common/models/teacher.model';
import { FormBuilder, FormControl, FormGroup, Validators } from '@angular/forms';
import { DiplomService } from '../../common/services/diplom.service';
import { ToastrService } from 'ngx-toastr';
import { DiplomorderModel } from '../../common/models/diplomorder.model';
import * as _moment from 'moment';
import { MomentDateAdapter, MAT_MOMENT_DATE_ADAPTER_OPTIONS } from '@angular/material-moment-adapter';
import { DateAdapter, MAT_DATE_FORMATS, MAT_DATE_LOCALE } from '@angular/material/core';
import { MatDialog } from "@angular/material/dialog";
import { ConfirmDialogComponent } from "../dialogs/confirm-dialog/confirm-dialog.component";
import { DiplomDataService } from "../../common/services/diplom-data.service";
const moment = _moment;
export const MY_FORMATS = {
    parse: {
        dateInput: 'D.MM.YYYY'
    },
    display: {
        dateInput: 'DD.MM.YYYY',
        monthYearLabel: 'MMMM Y',
        dateAllyLabel: 'LL',
        monthYearAllyLabel: 'MMMM Y'
    }
};
@Component({
    selector: 'app-diplom-order',
    templateUrl: './diplom-order.component.html',
    styleUrls: ['./diplom-order.component.scss'],
    providers: [{
        provide: DateAdapter,
        useClass: MomentDateAdapter,
        deps: [MAT_DATE_LOCALE, MAT_MOMENT_DATE_ADAPTER_OPTIONS]
    },
    {provide: MAT_DATE_FORMATS, useValue: MY_FORMATS},
    ],
})
export class DiplomOrderComponent implements OnInit {
    diplomOrderList: DiplomorderModel[];
    diplomOrderForm: FormGroup;
    buttonTitleAdd = true;
    tab = 'add';
    constructor(
        private diplomService: DiplomService,

```

```

    private fb: FormBuilder,
    private toastr: ToastrService,
    private dialog: MatDialog,
    private diplomDataService: DiplomDataService
  ) {
  }
  ngOnInit(): void {
    this.getReviewerList();
    this.formInit();
  }
  private getReviewerList() {
    this.diplomService.getDiplomorders().subscribe(res => {
      this.diplomOrderList = res;
    });
  }
  private formInit() {
    this.diplomOrderForm = this.fb.group({
      Id: [0],
      Name: ['', Validators.required],
      Dateorder: [new FormControl(moment()), Validators.required],
    });
  }
  onSubmit() {
    const diplomOrder: DiplomorderModel = this.diplomOrderForm.getRawValue();
    if (diplomOrder.Dateorder === '') {
      diplomOrder.Dateorder = null;
    } else {
      diplomOrder.Dateorder = moment(diplomOrder.Dateorder).add(4, 'hours').toISOString();
    }
    if (this.buttonTitleAdd) {
      this.diplomService.createDiplomorder(diplomOrder).subscribe(res => {
        this.toastr.success('Добавлен');
        this.getReviewerList();
        this.diplomDataService.changeOrder = true;
      });
    } else {
      this.diplomService.updateDiplomorder(diplomOrder).subscribe(res => {
        this.toastr.success('Обновлен');
        this.getReviewerList();
        this.diplomDataService.changeOrder = true;
      });
    }
    this.formInit();
    this.buttonTitleAdd = true;
    this.tab = 'add';
  }
  deleteDiplomOrder(id: number) {
    const dialogRef = this.dialog.open(ConfirmDialogComponent, {
      width: '250px',
    });
  }

```

```

    dialogRef.afterClosed().subscribe(result => {
      if (result) {
        this.diplomService.deleteDiplomorder(id).subscribe(res => {
          this.toastr.success('Удален');
          this.getReviewerList();
          this.diplomDataService.changeOrder = true;
        }, error => {
          this.toastr.error('Нарушение ограничений целостности');
        });
      }
      this.formInit();
      this.buttonTitleAdd = true;
      this.tab = 'add';
    });
  }
  changeDiplomOrder(id: number) {
    const item = this.diplomOrderList.find(el => el.Id === id);
    this.diplomOrderForm.patchValue({
      ...item
    });
    this.buttonTitleAdd = false;
    this.tab = 'change';
  }
  changeTab(ev) {
    this.formInit();
    this.buttonTitleAdd = true;
    this.tab = 'add';
  }
  strToDate(str: string) {
    const date = new Date(str);
    const day = date.getDate();
    const month = date.getMonth() + 1;
    const year = date.getFullYear();
    re-
turn `${day < 10 ? `0${day}` : day}.${month < 10 ? `0${month}` : month}.${year}`;
  }
}
import {Component, OnInit} from '@angular/core';
import {DiplomService} from '../../common/services/diplom.service';
import {TeacherModel} from '../../common/models/teacher.model';
import {FormBuilder, FormGroup, Validators} from '@angular/forms';
import {ToastrService} from 'ngx-toastr';
import {DiplomModel} from '../../common/models/diplom.model';
import {MatDialog} from '@angular/material/dialog';
import {ConfirmDialogComponent} from '../dialogs/confirm-dialog/confirm-dialog.component';
import {DiplomDataService} from '../../common/services/diplom-data.service';
@Component({
  selector: 'app-pm',
  templateUrl: './pm.component.html',
  styleUrls: ['./pm.component.scss']
})

```



```

export class PmComponent implements OnInit {
  pmList: TeacherModel[];
  pmForm: FormGroup;
  buttonTitleAdd = true;
  tab = 'add';
  constructor(
    private diplomService: DiplomService,
    private fb: FormBuilder,
    private toastr: ToastrService,
    private dialog: MatDialog,
    private diplomDataService: DiplomDataService
  ) {}
  ngOnInit(): void {
    this.getPmList();
    this.formInit();
  }
  private getPmList() {
    this.diplomService.getPms().subscribe(res => {
      this.pmList = res;
    });
  }
  private formInit() {
    this.pmForm = this.fb.group({
      Id: [0],
      Fio: ['', Validators.required],
    });
  }
  onSubmit() {
    const pm: TeacherModel = this.pmForm.getRawValue();
    if (this.buttonTitleAdd) {
      this.diplomService.createPm(pm).subscribe(res => {
        this.toastr.success('Добавлен');
        this.getPmList();
        this.diplomDataService.changePm = true;
      });
    } else {
      this.diplomService.updatePm(pm).subscribe(res => {
        this.toastr.success('Обновлен');
        this.getPmList();
        this.diplomDataService.changePm = true;
      });
    }
    this.formInit();
    this.buttonTitleAdd = true;
  }
  deletePm(id: number) {
    const dialogRef = this.dialog.open(ConfirmDialogComponent, {
      width: '250px',
    });
    dialogRef.afterClosed().subscribe(result => {
      if (result) {
        this.diplomService.deletePm(id).subscribe(res => {
          this.toastr.success('Удален');
          this.getPmList();
        });
      }
    });
  }
}

```

```

        this.diplomDataService.changePm = true;
    }, error => {
        this.toastr.error('Нарушение ограничений целостности');
    });
}
this.formInit();
this.buttonTitleAdd = true;
this.tab = 'add';
});
}
changePm(id: number) {
    const item = this.pmList.find(el => el.Id === id);
    this.pmForm.patchValue({
        ...item
    });
    this.buttonTitleAdd = false;
    this.tab = 'change';
}
changeTab(ev) {
    this.formInit();
    this.buttonTitleAdd = true;
    this.tab = 'add';
}
}
import {Component, Input, OnInit, DoCheck} from '@angular/core';
import {DiplomService} from '../../../common/services/diplom.service';
import {DiplomModel} from '../../../common/models/diplom.model';
import {DiplomDataService} from '../../../common/services/diplom-data.service';
import {DiplomInfoModel} from '../../../common/models/diplomInfo.model';
import {Router} from '@angular/router';
import {PasswordDialogComponent} from '../../../dialogs/password-dialog/password-dialog.component';
import {ConfirmDialogComponent} from '../../../dialogs/confirm-dialog/confirm-dialog.component';
import {MatDialog} from '@angular/material/dialog';
import {ToastrService} from "ngx-toastr";
@Component({
    selector: 'app-diplom-list',
    templateUrl: './diplom-list.component.html',
    styleUrls: ['./diplom-list.component.scss']
})
export class DiplomListComponent implements OnInit, DoCheck {
    @Input() diplomsList: DiplomModel[];
    @Input() infoAboutDiplom: DiplomInfoModel;
    isAdmin: boolean;
    constructor(
        private diplomDataService: DiplomDataService,
        private diplomService: DiplomService,
        private router: Router,
        private toastr: ToastrService,
        public dialog: MatDialog
    ) {

```

```

) {
}
ngOnInit(): void {
  const filterItem = JSON.parse(localStorage.getItem('filter'));
  if (filterItem) {
    Object.keys(filterItem).forEach(key => {
      const keyObj = `${key}Check`;
      filterItem[keyObj] = true;
    });
  }
}
ngDoCheck() {
  this.isAdmin = this.diplomDataService.isAdmin;
}
changeDiplom(e, id: number) {
  e.preventDefault();
  e.stopPropagation();
  this.diplomDataService.selectDiplomId = id;
  this.diplomDataService.isDiplomSelect = true;
}
createTitleString(diplom: DiplomModel) {
  const diplomOrder = this.infoAboutDiplom.diplomorder-
List.find((el) => el.Id === diplom.DiplomorderId);
  const specialty = this.infoAboutDiplom.specialtyL-
ist.find((el) => el.Id === diplom.SpecialtyId);
  const pm = this.in-
foAboutDiplom.pmList.find((el) => el.Id === diplom.PmId);
  return `[${diplomOrder.Name}, ${this.strToDate(diplo-
mOrder.Dateorder)}]-${diplom.QueueNumber}-${specialty.Name}-
${diplom.Fio}/${pm.Fio}`;
}
getNormcontroller(normcontrollerId: number) {
  const normcontroller = this.infoAboutDiplom.normcontrol-
lerList.find((el) => el.Id === normcontrollerId);
  return normcontroller.Fio;
}
getReviewer(reviewerId: number) {
  const reviewer = this.infoAboutDiplom.reviewer-
List.find((el) => el.Id === reviewerId);
  return reviewer.Fio;
}
getCommission(commissionId: number) {
  const commission = this.infoAboutDiplom.commission-
List.find((el) => el.Id === commissionId);
  return commission.Fio;
}
getChairman(chairmanId: number) {
  const chairman = this.infoAboutDiplom.chairman-
List.find((el) => el.Id === chairmanId);
  return chairman.Fio;
}
deleteDiplom(id: number) {
  const dialogRef = this.dialog.open(ConfirmDialogComponent, {
    width: '250px',

```

```

    });

    dialogRef.afterClosed().subscribe(result => {
      if (result) {
        this.diplomService.deleteDiplom(id).subscribe(res => {
          this.toastr.success('Удален');
          this.diplomDataService.isDiplomsUpdate = true;
        }, error => {
          this.toastr.error('Нарушение ограничений целостности');
        });
      }
    });
  }
  openDiplom(id: number) {
    this.router.navigate(['/diplom-components-info/' + id]);
  }
  strToDate(str: string) {
    const date = new Date(str);
    const day = date.getDate();
    const month = date.getMonth() + 1;
    const year = date.getFullYear();
    re-
turn `${day < 10 ? `0${day}` : day}.${month < 10 ? `0${month}` : mont
h}.${year}`;
  }
}
import {Component, OnDestroy, OnInit} from '@angular/core';
import {ActivatedRoute, Router} from '@angular/router';
import {DiplomService} from '../../../common/services/diplom.ser-
vice';
import {DiplomModel} from '../../../common/models/diplom.model';
import {zip} from 'rxjs';
import { timer } from 'rxjs';
@Component({
  selector: 'app-diplom-id',
  templateUrl: './diplom-id.component.html',
  styleUrls: ['./diplom-id.component.scss']
})
export class DiplomIdComponent implements OnInit, OnDestroy {

  constructor(
    private router: Router,
    private route: ActivatedRoute,
    private diplomService: DiplomService
  ) {
  }
  diplomId: number;
  pm = '';
  normcontroller = '';
  reviewer = '';
  chairman = '';
  diplomOrder = '';
  specialty = '';
  commission = '';

```

```

diplom = new DiplomModel();
subscribeTimer: any;
timeLeft = 20;
timeEnd = 20;
timerText: string;
color = '#212529';
timer = false;
interval;
checkInterval;
ngOnInit(): void {
    this.diplomId = this.route.snapshot.params.id;
    this.getDiplom();
    this.startTimer();
}
ngOnDestroy() {
    clearInterval(this.interval);
    clearInterval(this.checkInterval);
}
getDiplom() {
    this.diplomService.getDiplomById(this.diplomId).sub-
scribe(res => {
    if (res) {
        this.diplom = res;
        if (this.diplom.Time !== 0) {
            this.timeLeft = this.diplom.Time;
            this.start();
        }
        this.getDiplomInfo(res);
    }
    });
}
getDiplomInfo(diplom: DiplomModel) {
    zip(
        this.getPmList(),
        this.getNormcontrollerList(),
        this.getReviewerList(),
        this.getChairmanList(),
        this.getDiplomorderList(),
        this.getSpecialtyList(),
        this.getCommissionList(),
    ).subscribe(([pm, normcontroller, reviewer, chairman, diplo-
morder, specialty, commission]) => {
        this.pm = pm.find(el => el.Id === diplom.PmId).Fio;
        this.normcontroller = normcontrol-
ler.find(el => el.Id === diplom.NormcontrollerId).Fio;
        this.reviewer = reviewer.find(el => el.Id === diplom.Review-
erId).Fio;
        this.chairman = chairman.find(el => el.Id === diplom.Chair-
manId).Fio;
        const diplomOrderModel = diplo-
morder.find(el => el.Id === diplom.DiplomorderId);
        this.diplomOrder = `${diplomOrderModel.Name} ${this.strTo-
Date(diplomOrderModel.Dateorder)}`;
    });
}

```

```

        this.specialty = specialty.find(el => el.Id === diplom.Special-
tyId).Name;
        this.commission = commission.find(el => el.Id === diplom.Com-
missionId).Fio;
    });
}
private getPmList() {
    return this.diplomService.getPms();
}
private getNormcontrollerList() {
    return this.diplomService.getNormcontrollers();
}
private getReviewerList() {
    return this.diplomService.getReviewers();
}
private getChairmanList() {
    return this.diplomService.getChairmans();
}
private getDiplomorderList() {
    return this.diplomService.getDiplomorders();
}
private getSpecialtyList() {
    return this.diplomService.getSpecialtys();
}
private getCommissionList() {
    return this.diplomService.getCommissions();
}
private updateDiplom(diplom: DiplomModel) {
    if (this.timer) {
        this.diplomService.updateDiplom(diplom).subscribe(res => {});
    }
}
onMain() {
    this.router.navigate(['/']);
}
stop() {
    clearInterval(this.interval);
    this.diplom.Time = 0;
    this.updateDiplom(this.diplom);
    this.timerText = '';
    this.color = '#212529';
    this.timer = false;
}
start() {
    if (!this.interval) {
        this.interval = setInterval(() => {
            this.diplom.Time = this.timeLeft;
            this.timer = true;
            this.updateDiplom(this.diplom);
            console.log('1')
            if (this.timeLeft > 0) {
                this.timeLeft--;
                this.timerText = this.secToMin(this.timeLeft.toString());
            } else {

```

```

        this.timeLeft--;
        this.color = 'red';
        this.timerText = this.secToMin((this.timeLeft * -
1).toString());
    }
    }, 1000);
}
}
startTimer() {
    this.checkInterval = setInterval(() => {
        console.log('2')
        this.diplomService.getDiplomById(this.diplomId).sub-
scribe(res => {
            if (res.Time !== 0) {
                clearInterval(this.checkInterval);
                this.start();
            }
        });
    }, 2000);
}
secToMin(sec: string) {
    const min = Math.floor(+sec / 60);
    let seconds = (+sec - min * 60).toString();
    if (+seconds < 10) {
        seconds = `0${seconds}`;
    }
    return `${min}:${seconds}`;
}
strToDate(str: string) {
    const date = new Date(str);
    const day = date.getDate();
    const month = date.getMonth() + 1;
    const year = date.getFullYear();
    re-
turn `${day < 10 ? `0${day}` : day}.${month < 10 ? `0${month}` : mont
h}.${year}`;
}
}
}

```

ПРИЛОЖЕНИЕ Е
Таблица экономических показателей