МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования «Самарский национальный исследовательский университет имени академика С.П. Королева» (Самарский университет)

Институт информатики и кибернетики

Кафедра информационных систем и технологий

КУРСОВОЙ ПРОЕКТ ПО ДИСЦИПЛИНЕ

«Разработка WEB-приложений»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

«Автоматизированная информационная система заказа клининга на объект»

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е.А. Шкайдуров

*(подпись)*

Руководитель работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ И.А. Лёзин

*(подпись)*

Самара 2024

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования «Самарский национальный исследовательский университет имени академика С.П. Королева»

(Самарский университет)

Институт информатики и кибернетики

Кафедра информационных систем и технологий

**ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ**

Обучающемуся Шкайдурову Егору Александровичу группы 6402-090301D

Тема проекта: «Автоматизированная информационная система заказа клининга на объект»

|  |  |  |
| --- | --- | --- |
| Планируемые результа ты освоения образовательной программы  (компетенции) | Планируемые результаты практики | Содержание задания |
| ПК-2 Способен разрабатывать модели компонентов информационных систем, включая модели баз данных и модели интерфейсов «человек - электронно- вычислительная машина»  ПК-2.3. Разрабатывает Web-приложения, компоненты программных продуктов, в том числе системных. | **знать**: теоретические основы проектирования и создания моделей баз данных и программного обеспечения при разработке web-приложений в рамках технологии Java Enterprise Edition; **уметь**: создавать и проверять работоспособность моделей баз данных и программного обеспечения при разработке web-приложений в рамках технологии Java Enterprise Edition; **владеть**: современными программными продуктами для создания моделей баз данных и программного обеспечения при разработке web-приложений в рамках технологии Java Enterprise Edition. | 1. Разработка логической модели базы данных АИС заказа клининга 2. Разработка автоматизированной информационной системы заказа клининга. 3. Отладка и тестирование разработанной автоматизированной информационной системы заказа клининга. 4. Проведение экспериментов и анализ результатов. |

Дата выдачи задания 6 февраля 2023 г.

Срок представления на кафедру пояснительной записки 05 мая 2023 г.

Руководитель курсового проекта

И.о. зав. каф. ИСТ, к.т.н., доцент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ И.А. Лёзин

*(подпись)*

Задание принял к исполнению

обучающийся группы № 6402-090301D\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е.А. Шкайдуров

*(подпись)*

РЕФЕРАТ

Пояснительная записка к курсовому проекту:28 страниц, 21 рисунок, 10 источников, 1 приложение.

ИНФОРМАЦИОННАЯ СИСТЕМА, СУБД, POSTGRESQL, SPRING-BOOT, WEB-ИНТЕРФЕЙС, ANGULAR, КЛИНИНГ.

Цель работы – разработать автоматизированную информационную систему заказа клининга на разные объекты.

Во время курсового проекта разработана информационная система на основе Spring Boot-технологий. В качестве СУБД использован PostgreSQL.

Система предоставляет возможность заказа клининга на определённый объект в определенное время, просмотра выполненных и отменённых заказов. Вдобавок представлена панель для работы клинера. Вся работа с системой производится через Web-интерфейс, реализованный на фреймворке Angular.

Программа написана на языке Java в среде Visual Studio Code и функционирует под управлением операционной системы Windows 10.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 6](#_Toc133076847)

[1 Структура базы данных 7](#_Toc133076848)

[1.1 Логическая модель базы данных 7](#_Toc133076849)

[1.2 Физическая структура базы данных 8](#_Toc133076850)

[2 Архитектура приложения 8](#_Toc133076851)

[2.1 Архитектурная модель 8](#_Toc133076852)

[2.2 Выбор и обоснование использования технологий 10](#_Toc133076853)

[2.2.1 Выбор языка программирования и среды разработки для реализации серверного уровня web-приложения 10](#_Toc133076854)

[2.2.2 Выбор системы управления базами данных 10](#_Toc133076855)

[2.2.3 Выбор способа работы с базой данных 11](#_Toc133076856)

[2.2.4 Выбор фреймворка для разработки уровня клиента web-приложения 11](#_Toc133076857)

[2.3 Средства моделирования 12](#_Toc133076858)

[2.3.1 Диаграмма вариантов использования 12](#_Toc133076859)

[2.3.2 Диаграмма классов 13](#_Toc133076860)

[3 Интерфейс пользователя 15](#_Toc133076861)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 25](#_Toc133076862)

[ПРИЛОЖЕНИЕ А – Классы сущностей 26](#_Toc133076863)

ВВЕДЕНИЕ

Целью курсового проекта является реализация информационной системы заказа клининга на объект.

Автоматизированная система заказа клининга на объект является важным инструментом для упрощения жизни людей, большинство из которых не успевают выполнять домашние дела в силу своей загруженности. Также система предоставляет рабочие места для активных людей, желающих подработать в роли клинера.

Разрабатываемая информационная система должна обеспечивать полный контроль над состоянием заказа, просмотр выполненных заказов, возможность оставления отзыва по каждому заказу, что в совокупности обеспечит корректную работу системы и учёт пожелания каждой роли.

1. Структура базы данных
   1. Логическая модель базы данных

Логическая модель базы данных — это представление структуры данных в базе данных, независимое от конкретной СУБД и физической реализации. Она описывает сущности, связи между ними, их атрибуты, а также правила, ограничения и свойства данных. Логическая модель является основой для создания физической модели, которая определяет, как данные будут храниться и обрабатываться в СУБД. Кроме того, логическая модель позволяет описать все объекты базы данных и обеспечить их целостность и надежность.

Основным средством разработки логической модели данных в настоящий момент являются различные варианты ER-диаграмм. ER-диаграмма (Entity-Relationship diagram) - это визуальное представление структуры базы данных, которое позволяет понять, как сущности взаимодействуют друг с другом. ER-диаграммы используют модель типа «сущность-связь» (ER- модель), основными понятиями которой являются сущность, экземпляр сущности, атрибут и связь [1].

На рисунке 1 представлена логическая модель данных реализуемой системы.

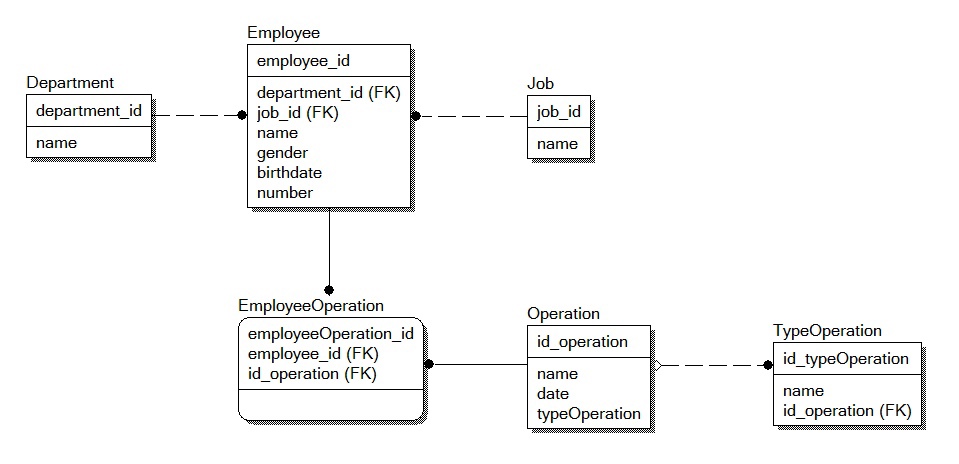


Рисунок 1 – Логическая модель базы данных

Логическая модель базы данных учёта сотрудников предприятия состоит из пяти сущностей:

* Заказ;
* Клиент;
* Клинер;
* Отзыв;
* Фотография;
  1. Физическая структура базы данных

Физическая модель базы данных – это конкретная реализация базы данных на определенной СУБД. Она включает в себя информацию о таблицах, полях, индексах и связях между таблицами. Она также определяет типы данных для каждого поля и индексы, которые используются для ускорения поиска данных [2].

Физическая модель описывает как данные будут храниться на жестком диске, как они будут индексироваться и как они будут доступны для поиска и манипуляции. Она обычно включает в себя определения типов данных, индексы, ограничения целостности и т. д. В общем смысле, физическая модель базы данных отображает структуру базы данных в соответствии с ее конкретной реализацией.

На рисунке 2 приведена физическая модель данных реализуемой системы.

Изображение выглядит как диаграмма

Автоматически созданное описание

Рисунок 2 – Физическая модель данных

1. Архитектура приложения
   1. Архитектурная модель

Архитектура программного обеспечения – это высокоуровневое описание структуры программного продукта, которое определяет его компоненты, связи между ними и их взаимодействие. Она описывает, как система будет функционировать и как ее компоненты будут работать вместе для достижения поставленных целей [3].

Существует несколько видов архитектур программного обеспечения:

1. Монолитная архитектура - система представляет собой единый целый модуль, который содержит всю логику приложения и функциональность;
2. Клиент-серверная архитектура - приложение разбивается на две части: клиентскую и серверную. Клиентская часть отвечает за взаимодействие с пользователем, а серверная - за хранение и обработку данных;
3. Трехслойная архитектура - система разделяет приложение на три слоя: пользовательский интерфейс, бизнес-логику и слой доступа к данным;
4. Микросервисная архитектура - приложение разбивается на отдельные микросервисы, каждый из которых выполняет отдельную функцию.

Было решено использовать трехуровневую модель архитектуры для реализации проектируемой системы, так как имеет следующие преимущества:

* Разделение ответственностей: каждый слой отвечает за свою задачу, что упрощает разработку и поддержку приложения;
* Улучшенная масштабируемость: каждый слой может быть масштабирован и изменен независимо от других;
* Уменьшение зависимостей: каждый слой имеет свои зависимости, что уменьшает связность между компонентами и повышает их независимость;

На рисунке 3 схематично отображена трехуровневая модель архитектуры.



Рисунок 3 – Трехуровневая модель архитектуры

* 1. Выбор и обоснование использования технологий
     1. Выбор языка программирования и среды разработки для реализации серверного уровня web-приложения

В качестве языка программирования выбрана Java и фреймворк Spring Boot.

Java является одним из наиболее распространенных языков программирования для разработки веб-приложений. Он широко используется для создания серверных приложений. Существует множество фреймворков и библиотек для Java, которые значительно упрощают процесс разработки веб-приложений.

Spring Boot – фреймворк для создания приложений на языке Java. Он предоставляет возможность быстрого и удобного создания веб-приложений, используя уже готовые компоненты и шаблоны проектирования. Spring Boot включает все необходимые библиотеки, что позволяет разработчику сразу же начинать работу над проектом, не тратя время на конфигурацию приложения. Он также позволяет создавать микросервисы и интегрировать различные технологии, такие как базы данных, технологии обработки сообщений, без большого количества дополнительного кода. Он также имеет интеграцию с другими популярными фреймворками и библиотеками, такими как Hibernate, Thymeleaf, Spring Data и многими другими [4].

* + 1. Выбор системы управления базами данных

СУБД (система управления базами данных) – это комплекс программно-языковых средств, позволяющих создать базы данных и управлять данными. СУБД обеспечивает доступ к данным, позволяет выполнять запросы и обеспечивает целостность и безопасность данных. Она также может обеспечивать масштабируемость и производительность базы данных. СУБД является неотъемлемой частью большинства приложений, которые требуют хранения и управления данными.

PostgreSQL – это открытая система управления базами данных, которая обладает высокой степенью надежности и устойчивости к сбоям благодаря использованию транзакций и механизмов восстановления данных. Она обладает высоким уровнем безопасности благодаря использованию механизмов аутентификации, авторизации и шифрования данных. PostgreSQL поддерживает множество стандартов SQL, что делает его совместимым с другими СУБД. Кроме того, PostgreSQL позволяет добавлять новые функции и возможности с помощью расширений, что делает его еще более гибким и мощным [5].

* + 1. Выбор способа работы с базой данных

JPA (Java Persistence API) – это спецификация Java EE для управления данными в реляционных базах данных с помощью объектно-ориентированной модели. JPA предоставляет унифицированный интерфейс для доступа к данным, а также методы для сохранения, изменения, удаления и поиска объектов в базе данных. В Java-приложениях JPA используется для сокращения рутины и повышения производительности, в том числе для уменьшения нагрузки на базу данных и улучшения управляемости кода. [6].

JPA также обеспечивает удобную работу с отношениями между объектами в базе данных, такими как связи один-ко-многим и многие-ко-многим.

* + 1. Выбор фреймворка для разработки уровня клиента web-приложения

Для реализации уровня клиента разрабатываемой системы выбран фреймворк Angular.

Angular – это платформа для разработки, написанная на TypeScript. Он предоставляет множество инструментов и функций для создания мощных и динамических веб-приложений [7]. Angular обладает следующими преимуществами:

* Модульность и компонентный подход: Angular разбивает приложение на отдельные компоненты, что упрощает его разработку и поддержку.
* Мощный набор инструментов: Angular предоставляет разработчикам множество инструментов и функций для создания мощных и динамических веб-приложений.
* Поддержка TypeScript: TypeScript является строго типизированным языком программирования, что упрощает процесс разработки и улучшает качество кода.
* Встроенная поддержка для маршрутизации, обработки форм и анимации: Angular предоставляет встроенную поддержку для многих функций, что значительно упрощает процесс разработки веб-приложений.
  1. Средства моделирования

Средства моделирования UML представляют собой программные инструменты, которые позволяют создавать диаграммы и модели на основе стандарта UML. Эти инструменты обычно включают в себя графический интерфейс для создания и редактирования диаграмм, а также функциональность для экспорта и импорта моделей.

UML (Unified Modeling Language) – это стандартный язык моделирования, который используется для описания, визуализации, проектирования и документирования программных систем [8].

Средства моделирования UML могут быть использованы для создания различных типов диаграмм, таких как диаграммы классов, диаграммы прецедентов, диаграммы последовательностей и диаграммы состояний. Каждый тип диаграммы предназначен для описания определенных аспектов программной системы.

* + 1. Диаграмма вариантов использования

Диаграмма вариантов использования – специальный тип диаграммы, который используется для анализа и описания функций и возможностей системы или приложения. Она позволяет описать, как система будет использоваться в реальном мире и какие взаимодействия будут происходить между ее компонентами и пользователями [9].

Диаграмма вариантов использования системы изображена на рисунке 4

Изображение выглядит как диаграмма

Автоматически созданное описание  
Рисунок 4 – Диаграмма вариантов использования

* + 1. Диаграмма классов

Диаграмма классов — это графическое представление классов и связей между ними в системе. Она используется для описания структуры системы и ее классов. Она позволяет определить классы, их атрибуты и методы, а также отношения между классами [10].

На рисунке 5 изображена диаграмма сущностных классов системы, на рисунке 6 полная диаграмма классов системы.

Изображение выглядит как диаграмма

Автоматически созданное описание

Рисунок 5 – Диаграмма сущностных классов

Изображение выглядит как диаграмма, схематичный

Автоматически созданное описание

Рисунок 6 – Диаграмма классов системы

1. Интерфейс пользователя

Пользовательский интерфейс – визуальная часть программы или устройства, предназначенная для взаимодействия с пользователем. UI включает в себя все элементы, которые пользователь видит на экране, а также способы, которыми пользователь может взаимодействовать с ними [10].

Хороший пользовательский интерфейс должен быть интуитивно понятным, удобным и привлекательным для пользователя. UI должен быть легким в использовании и удобным для работы. Это может быть достигнуто путем использования простых и понятных элементов управления, логической организации информации и удобной навигации.

Элементы пользовательского интерфейса могут включать в себя кнопки, поля ввода, меню, диалоговые окна, иконки и т. д. Каждый элемент имеет свою функцию и предназначен для определенного действия.

При открытии приложения пользователь попадает на страницу списка департаментов. В верхней части экрана находится меню навигации, которое позволяет перейти на страницы списка департаментов, сотрудников, операций и должностей. На рисунке 7 представлено изображение панели навигации.



Рисунок 7 – Панель навигации

На странице департаментов представлен список всех департаментов предприятия. Пользователь может добавить новый департамент, перейти на страницу департамента, изменить его или удалить. На рисунке 8 представлена страница департаментов.

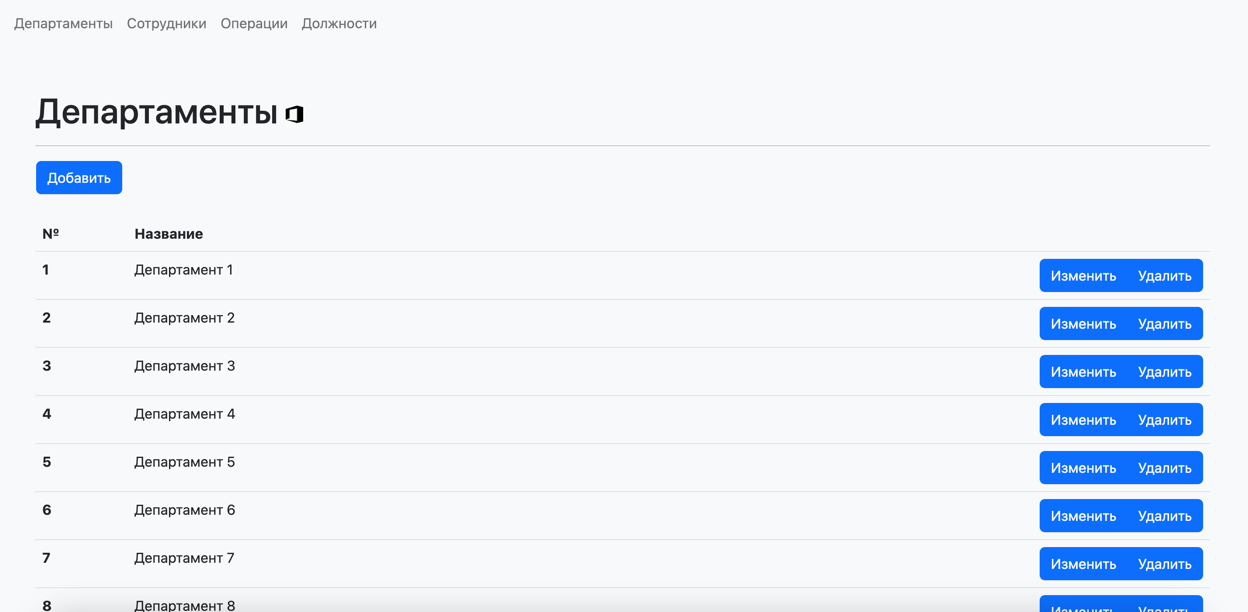


Рисунок 8 – Страница департамента

При нажатии на кнопку «Добавить» пользователю открывается модальное окно «Добавить департамент». На рисунке 9 представлено модальное окно добавления департамента.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 9 – Модальное окно добавления департамента

При нажатии на кнопку «Изменить» пользователю открывается модальное окно, аналогичное добавлению, предыдущее название заполняется автоматически. На рисунке 10 представлено модальное окно изменения департамента.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 10 – Модальное окно изменения департамента

При двойном нажатии на департамент открывается страница этого департамента. На рисунке 11 представлена страница департамента.

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 11 – Страница департамента

При нажатии на кнопку «Добавить» пользователю открывается модальное окно «Добавить сотрудника». На рисунке 12 представлено модальное окно добавления сотрудника.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 12 – Модальное окно добавления сотрудника в департамент

При нажатии на поле «Должность» открывается модальное окно выбора значений данного объекта. На рисунке 13 представлено модальное окно выбора значений.

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 13 – Модальное окно справочника

Для того чтобы выбрать значение из списка необходимо нажать на него.

Чтобы сохранить сотрудника необходимо нажать кнопку «Сохранить».

При нажатии на кнопку «Изменить» - откроется модальное окно «Изменение сотрудника». Данное окно аналогично окну «Создания сотрудника», значения полей будут автоматически подставлены. После внесения изменений необходимо нажать кнопку «Сохранить». Нажатие на кнопку «Удалить» удаляет сотрудника.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 14 – Модальное окно изменения сотрудника

На странице Сотрудники представлена информация о сотрудниках, пользователь может добавить нового сотрудника, перейти на страницу сотрудника, изменить его или удалить. Кнопки работают аналогично добавлению/изменению и удалению департамента. На рисунке 15 представлена страница с сотрудниками.

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 15 – Страница сотрудники

Если двойным кликом нажать на конкретного сотрудника, то откроется страница с информацией об этом сотруднике, также там будут представлены операции, которые выполняются этим сотрудником.

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 16 – Страница сотрудник

При нажатии на кнопку добавить операцию, откроется модальное окно для добавления операции. В данном окне необходимо заполнить все поля и нажать сохранить, затем на странице сотрудника отобразится добавленная операция.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 17 – Модальное окно создания операции

На странице Операции представлены типы операций и все операции. При нажатии на кнопку «Добавить тип операции» откроется модальное окно, в котором необходимо заполнить все поля и нажать «Сохранить». Затем в таблице с типами операций отобразится добавленный тип. Модальное окно с типами операций представлено на рисунке 19.

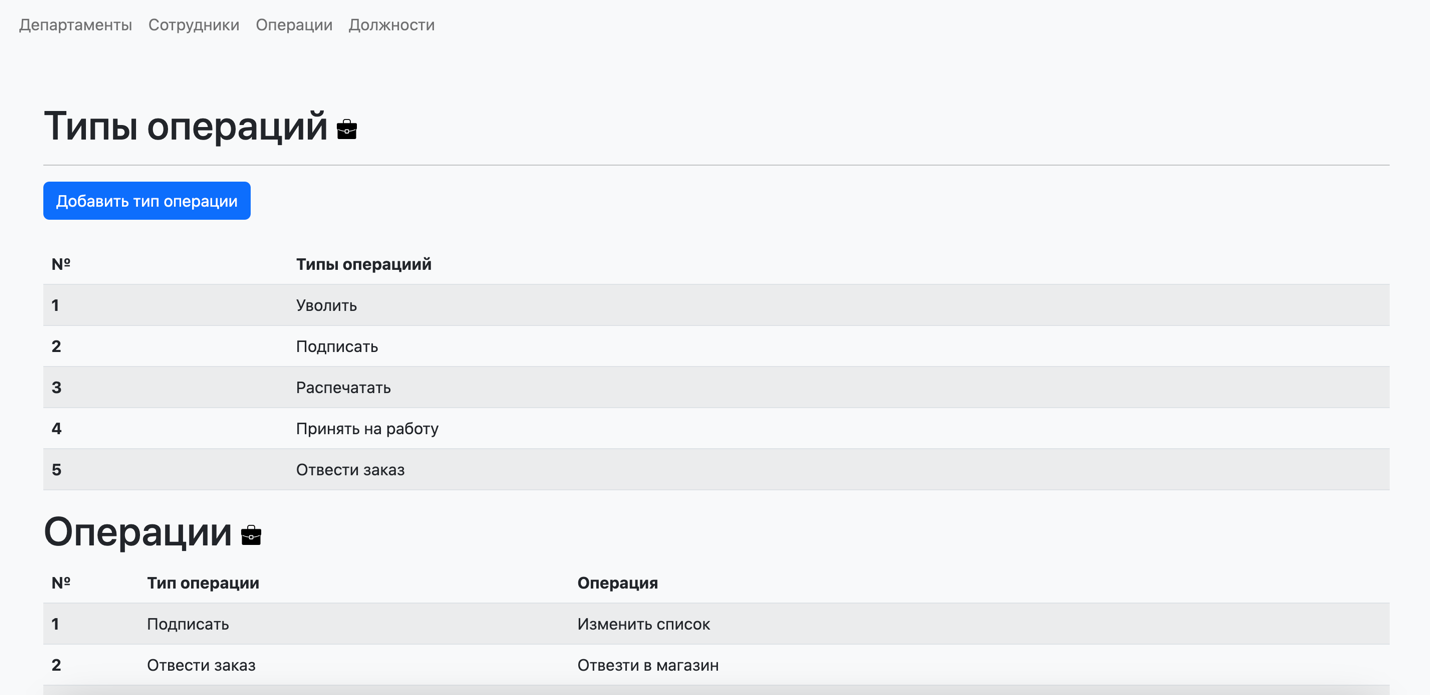


Рисунок 18 – Страница Операции

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 19 – Модальное окно типы операций

На странице Должности представлены все должности на предприятии, она находится на рисунке 20.

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 20 – Страница Должности

При нажатии на кнопку «Добавить» откроется модальное окно, в котором необходимо заполнить название должности и нажать «Сохранить», а при нажатии на кнопку «Изменить», в модальное окно подставится текущее название должности, которое можно изменить, либо не изменять. Окно изменения аналогично добавлению.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 21 – Модальное окно добавление должности

ЗАКЛЮЧЕНИЕ

В ходе курсового проекта была разработана и реализована информационная система учёта сотрудников предприятия на основе Spring Boot и фреймворка Angular.

Реализован web-интерфейс для взаимодействия пользователя с данными и выполнять основные операции: добавление, изменение и удаление в базе данных.

Так же были спроектированы и реализованы логическая и физическая модели базы данных, диаграммы вариантов использования и диаграмма классов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Понятие ER-модели. Понятие сущности (entity). Атрибуты. Виды атрибутов [Электронный ресурс] URL: https://www.bestprog.net/ru/2019/01/24/the-concept-of-er-model-the-concept-of-essence-and-communication-attributes-attribute-types-ru/ (дата обращения 24.04.2023).
2. Системы баз данных. Уровни моделирования ИС [Электронный ресурс] URL:https://dl.nure.ua/pluginfile.php/550/mod\_resource/content/2/content/  
   content4.html - (дата обращения 24.04.2023).
3. Архитектура информационной системы, типы архитектур [Электронный ресурс] URL: https://studbooks.net/2213447/informatika/  
   arhitektura\_informatsionnoy\_sistemy\_tipy\_arhitektur - (дата обращения: 24.04.2023).
4. Введение в Spring Boot: создание простого REST API на Java [Электронный ресурс] URL: https://habr.com/ru/articles/435144 (дата обращения 24.04.2023).
5. PostgreSQL — объектно-реляционная система управления базами данных [Электронный ресурс] URL: https://web-creator.ru/articles/postgresql (дата обращения 24.04.2023).
6. JPA: Знакомство с технологией [Электронный ресурс] URL: https://javarush.com/groups/posts/2259-jpa--znakomstvo-s-tekhnologiey - (дата обращения: 24.04.2023).
7. Введение в Angular [Электронный ресурс] URL:  [https://metanit.com/web/angular2/1.1.php](http://www.bseu.by/it/tohod/lekcii5.htm) (дата обращения: 24.04.2023).
8. Унифицированный язык моделирования UML [Электронный ресурс] URL: https://samara.mgpu.ru/~dzhadzha/dis/15/200.html (дата обращения: 24.04.2023).
9. Чигарина, Е.И. Базы данных [Текст] / Е.И. Чигарина. – Самара: СГАУ, 2015. – 208 с
10. Пользовательский интерфейс и его виды [Электронный ресурс]. URL: https://intuit.ru/studies/courses/3609/851/lecture/31652?page=3 (дата обращения: 24.04.2021).

ПРИЛОЖЕНИЕ А – Классы сущностей

package com.ssau.employeeAccounting.entity;  
  
import jakarta.persistence.Entity;  
import jakarta.persistence.GeneratedValue;  
import jakarta.persistence.GenerationType;  
import jakarta.persistence.Id;  
import lombok.Getter;  
import lombok.Setter;  
  
@Entity  
@Getter  
@Setter  
public class Department {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private long id;  
  
 private String name;  
}

package com.ssau.employeeAccounting.entity;  
  
import jakarta.persistence.\*;  
import lombok.Getter;  
import lombok.Setter;  
  
import java.util.Date;  
  
@Entity  
@Getter  
@Setter  
public class Employee {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private long id;  
  
 private String name;  
  
 private String gender;  
  
 private Date birthdate;  
  
 private int number;  
  
 @ManyToOne  
 @JoinColumn(name = "department\_id")  
 private Department department;  
  
 @ManyToOne  
 @JoinColumn(name = "job\_id")  
 private Job job;  
}

package com.ssau.employeeAccounting.entity;  
  
import jakarta.persistence.\*;  
import lombok.Getter;  
import lombok.Setter;  
  
@Entity  
@Getter  
@Setter  
public class EmployeeOperation {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private long id;  
  
 @ManyToOne  
 @JoinColumn(name = "employee\_id")  
 private Employee employee;  
  
 @ManyToOne  
 @JoinColumn(name = "operation\_id")  
 private Operation operation;  
}

package com.ssau.employeeAccounting.entity;  
  
  
import jakarta.persistence.Entity;  
import jakarta.persistence.GeneratedValue;  
import jakarta.persistence.GenerationType;  
import jakarta.persistence.Id;  
import lombok.Getter;  
import lombok.Setter;  
  
@Entity  
@Getter  
@Setter  
public class Job {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private long id;  
  
 private String name;  
}

package com.ssau.employeeAccounting.entity;  
  
  
import jakarta.persistence.\*;  
import lombok.Getter;  
import lombok.Setter;  
  
import java.util.Date;  
  
@Entity  
@Getter  
@Setter  
public class Operation {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private long id;  
  
 private String name;  
  
 private Date date;  
  
 @ManyToOne  
 @JoinColumn(name = "type\_operation\_id")  
 private TypeOperation typeOperation;  
}

package com.ssau.employeeAccounting.entity;  
  
  
import jakarta.persistence.Entity;  
import jakarta.persistence.GeneratedValue;  
import jakarta.persistence.GenerationType;  
import jakarta.persistence.Id;  
import lombok.Getter;  
import lombok.Setter;  
  
@Entity  
@Getter  
@Setter  
public class TypeOperation {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private long id;  
  
 private String name;  
}