МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение  
высшего образования

**«Южно-Уральский государственный университет**

**(национальный исследовательский университет)»**

**Высшая школа электроники и компьютерных наук**

**Кафедра системного программирования**

|  |
| --- |
| ДОПУСТИТЬ К ЗАЩИТЕ  Заведующий кафедрой, д.ф.-м.н., профессор  \_\_\_\_\_\_\_\_\_\_ Л.Б. Соколинский  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2024 г. |

**Разработка веб-приложения для транспортной компании «Луч»**

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

ЮУрГУ – 09.03.04.2024.308-350.ВКР

|  |  |
| --- | --- |
|  | Научный руководитель,  доцент кафедры СП, к.т.н.  \_\_\_\_\_\_\_\_\_\_ М.В. Сухов  Автор работы, студент группы КЭ-403  \_\_\_\_\_\_\_\_\_\_ В.А. Рявкин  Ученый секретарь  (нормоконтролер)  \_\_\_\_\_\_\_\_\_\_\_\_\_ И.Д. Володченко  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2024 г. |

Челябинск, 2024 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение  
высшего образования

**«Южно-Уральский государственный университет**

**(национальный исследовательский университет)»**

**Высшая школа электроники и компьютерных наук**

**Кафедра системного программирования**

УТВЕРЖДАЮ

Зав. кафедрой СП

\_\_\_\_\_\_\_\_\_\_ Л.Б. Соколинский

29.01.2024 г.

**ЗАДАНИЕ**

**на выполнение выпускной квалификационной работы бакалавра**

студенту группы КЭ-403

Рявкину Виталию Алексеевичу,

обучающемуся по направлению

09.03.04 «Проектирование и архитектура программных систем»

1. **Тема работы** (утверждена приказом ректора от \_\_.\_\_.2024 г. № \_\_\_)

Разработка веб-приложения для транспортной компании «Луч».

1. **Срок сдачи студентом законченной работы:** 03.06.2024 г.
2. **Исходные данные к работе2**
3. De Sanctis V. Building Web APIs with ASP.NET Core. //Manning, 2023. – 472 с.
4. Entity framework documentation. [Электронный ресурс] URL https://learn.microsoft.com/en-us/ef/core/ (дата обращения: 18.02.2024 г.).
5. ASP .NET Core web API documentation. [Электронный ресурс] URL: https://learn.microsoft.com/en-us/aspnet/core/fundamentals/apis?view=aspnetcore-7 (дата обращения: 18.02.2024 г.).
6. **Перечень подлежащих разработке вопросов**
7. Провести анализ предметной области.
8. Выполнить проектирование API и базы данных.
9. Реализовать API.
10. Провести тестирование API и сравнить полученные результаты с ожидаемыми.
11. **Дата выдачи задания:** 29.01.2024 г.

**Научный руководитель,**

доцент кафедры СП, к.т.н. М.В. Сухов

**Задание принял к исполнению** В.А. Рявкин

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 5](#_Toc159344342)

[1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 7](#_Toc159344343)

[1.1. Предметная область проекта 7](#_Toc159344344)

[1.2. Анализ аналогичных проектов 7](#_Toc159344345)

[1.2.1. Workday 7](#_Toc159344346)

[1.2.2. SAP SuccessFactors 8](#_Toc159344347)

[1.2.3. Kronos 8](#_Toc159344348)

[1.2.4. TSheets 9](#_Toc159344349)

[2. АНАЛИЗ ТРЕБОВАНИЙ К ПРОГРАММНОЙ СИСТЕМЕ 11](#_Toc159344350)

[2.1. Функциональные и нефункциональные требования к проектируемой системе 11](#_Toc159344351)

[2.2. Диаграмма вариантов использования 12](#_Toc159344352)

[ЗАКЛЮЧЕНИЕ 14](#_Toc159344353)

[ЛИТЕРАТУРА 15](#_Toc159344354)

# ВВЕДЕНИЕ

**Актуальность**

В современном мире большинство компаний стремятся к автоматизации всех бизнес-процессов, в том числе и в области управления персоналом. В сфере транспортной логистики и складского хозяйства этот вопрос особенно важен, так как необходимо точно и своевременно рассчитывать заработную плату сотрудников складов и иметь постоянный контроль над ситуацией на складах.

Разработанная система позволит автоматизировать процесс расчета заработной платы, уменьшить количество ошибок в работе управляющего персонала, упростить контроль над исполнителями и повысить эффективность работы сотрудников складов. Это также предоставит возможность руководству быстро получать информацию о заработной плате сотрудников и проводить анализ затрат на персонал.

**Постановка задачи**

Целью данной работы является разработка веб-приложения для транспортной компании «Луч» для расчета заработной платы сотрудников складов и контроля качества выполненных работ. Для ее достижения необходимо решить следующие задачи:

1. проанализировать существующие аналоги;
2. на основе анализа спроектировать архитектуру системы и выбрать технологии, которые будут применяться для решения поставленной задачи;
3. реализовать приложение согласно спроектированной архитектуре;
4. протестировать приложение.

**Структура и содержание работы**

Работа состоит из введения, четырех глав, заключения и списка литературы. Объем работы составляет 50 страниц, объем списка литературы – 21 источник.

В первой главе проводится анализ предметной области и существующих работ, связанных с учетом заработных плат и контролем производительности персонала, что позволяет лучше понять контекст и потребности конечного пользователя.

Вторая глава посвящена анализу требований к программной системе. В ней определяются основные функциональные и нефункциональные требования, а также формируется диаграмма вариантов использования и спецификация основных прецедентов.

В третьей главе описана архитектура системы. Приведено общее описание архитектуры, описание компонентов, составляющих систему, модель базы данных и процесс работы с системой.

В четвертой главе приведена реализация ключевых компонентов системы, а также проводится тестирование системы для того, чтобы убедиться в корректности ее работы и соответствии заявленным требованиям.

**1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

**1.1. Предметная область проекта**

Предметная область проекта связана с веб-приложения для транспортной компании «Луч» для расчета заработной платы сотрудников складов и контроля качества выполненных работ. Основной задачей приложения является помощь управляющему персоналу в контроле посещаемости и качества работы подчиненных сотрудников.

**1.2. Анализ аналогичных проектов**

Существует множество аналогов систем для учета посещаемости сотрудников и просчета заработной платы на основе этой посещаемости и других факторов. В качестве примера возьмем некоторые из них: Workday, SAP SuccessFactors, Kronos Workforce Management, TSheets.

**1.2.1. Workday**

Workday – это облачный сервис управления персоналом и финансами для предприятий. Этот сервис позволяет компаниям управлять человеческими ресурсами, управлять зарплатами и выплатами, автоматизировать бухгалтерские процессы и многое другое. Он обеспечивает функции, такие как управление наймом, обучением и развитием сотрудников, управление рабочим временем и отпусками, управление производительностью, управление зарплатой и выплатами, бухгалтерские операции и аналитику.

Workday имеет обширную систему безопасности и защиты данных, которая обеспечивает защиту конфиденциальности информации компаний и сотрудников. Сервис также обеспечивает высокую масштабируемость и доступность, что позволяет компаниям масштабировать свой бизнес без проблем.

Технологии:

* используется язык программирования Java и фреймворк Spring;
* в качестве базы данных используется PostgreSQL;
* интерфейс реализован с помощью HTML, CSS и JavaScript.

**1.2.2. SAP SuccessFactors**

SAP SuccessFactors – это облачный сервис управления персоналом, предназначенный для организации HR-процессов в компаниях любого размера. Он включает в себя различные модули, такие как управление персоналом, найм, обучение и развитие, управление производительностью, компенсации и бенефиты, а также аналитику и отчетность.

Благодаря интеграции с другими системами, SAP SuccessFactors позволяет создавать единую информационную среду для управления персоналом и повышения эффективности бизнеса.

Технологии:

* использует язык программирования Java и фреймворк Spring;
* в качестве базы данных использует SAP HANA;
* интерфейс реализован с помощью HTML, CSS и JavaScript.

**1.2.3. Kronos**

Kronos – это облачный сервис управления рабочим временем, который предоставляет широкий спектр функций, включая возможность планирования графиков работы, отслеживания рабочего времени, учета отпусков и больничных, а также отчетности о рабочих часах и зарплате сотрудников, который предназначен для упрощения и автоматизации процессов учета рабочего времени в компаниях различных отраслей.

Кроме того, Kronos интегрируется с другими бизнес-приложениями, такими как системы управления персоналом, ERP-системы и системы учета рабочих часов, что позволяет создавать комплексные решения для управления бизнесом.

Сервис Kronos используется в компаниях различного масштаба и отраслей, включая розничную торговлю, здравоохранение, гостинично-ресторанный бизнес, производство и транспортную логистику. Он предоставляет компаниям возможность улучшить управление персоналом, повысить производительность и эффективность работы сотрудников, а также сократить затраты на учет рабочего времени и зарплаты.

Технологии, используемые при разработке:

* язык программирования Java;
* система управления базами данных Oracle;
* фреймворк Spring Framework.

**1.2.4. TSheets**

TSheets – это онлайн-сервис для учета рабочего времени, управления задачами и графиками работы сотрудников. С помощью этого сервиса компании могут легко отслеживать время работы своих сотрудников, а также управлять их графиками и задачами.

TSheets позволяет сотрудникам быстро и легко отмечать свое время работы через мобильное приложение, веб-приложение или через специальные устройства для учета времени. Система автоматически собирает данные о времени работы и обрабатывает их для удобного отчета и анализа.

Сервис также позволяет настраивать различные права доступа и роли для пользователей, а также создавать отчеты о времени работы, задачах и графиках работы. Кроме того, TSheets интегрируется с другими приложениями, такими как QuickBooks, Xero и Gusto, что упрощает управление бизнесом и финансами.

Технологии, используемые при разработке:

* язык программирования JavaScript и фреймворк Node.js для API;
* система управления базами данных MongoDB;
* фреймворк AngularJS для фронт-енда.

**Вывод**

После изучения существующих решений было принято решение реализовать веб-приложения для транспортной компании для расчета заработной платы сотрудников складов в виде отдельного ASP.NET Core Web API-приложения с использованием Entity Framework Core и PostgreSQL на сервере и Angular на клиенте.

ASP.NET Core Web API – это кросплатформенный фреймворк для создания веб-сервисов (API) на языке программирования C# с использованием технологии ASP.NET Core. Он предназначен для разработки API, которые могут быть использованы различными клиентскими приложениями, такими как веб-приложения, мобильные приложения, настольные приложения и другие.

Entity Framework Core – это ORM (Object-Relational Mapping) фреймворк, разработанный Microsoft для работы с базами данных в приложениях .NET Core. Он позволяет разработчикам работать с данными, используя объектно-ориентированный подход, что упрощает работу с базами данных и уменьшает количество кода, необходимого для взаимодействия с ними.

PostgreSQL [1] (или Postgres) – это реляционная объектно-ориентированная система управления базами данных (СУБД), которая использует язык SQL (Structured Query Language) для управления данными. PostgreSQL является свободно распространяемой и открытой системой, которая может работать на различных операционных системах, включая Linux, Windows, macOS и другие.

Angular – это фреймворк для разработки веб-приложений, основанный на языке TypeScript. Он предоставляет мощные инструменты, такие как компонентная архитектура, двустороннее связывание данных, внедрение зависимостей, модули, роутинг и командная строка Angular CLI. TypeScript придает приложениям статическую типизацию, улучшая производительность и облегчая разработку.

**2. АНАЛИЗ ТРЕБОВАНИЙ К ПРОГРАММНОЙ СИСТЕМЕ**

В результате анализа предметной области и обзора существующих аналогов были сформированы следующие два основных типа требований:

* функциональные требования – определяют, как должна работать система и какие функции она должна выполнять;
* нефункциональные требования – определяют ограничения, свойства и критерии качества системы.

**2.1. Функциональные и нефункциональные требования к проектируемой системе**

1. Система должна предоставлять возможность изменения информации о личных данных сотрудников.
2. Система должна предоставлять разграничение уровня доступов для сотрудников с разными должностями.
3. Система должна сохранять и отслеживать любые изменения внутри себя.
4. Система должна предоставлять возможность отметки посещения сотрудников, приходящих на смену.
5. Система должна предоставлять возможность открытия и закрытия смены.
6. Система должна предоставлять возможность начальникам смены вносить комментарии о работе сотрудника на их смене.
7. Система должна предоставлять возможность редактирования информации об имеющихся должностях и позволять создавать новые.
8. Система должна предоставлять возможность получения отчета о сменах, прошедших на выбранном складе за выбранный месяц.
9. Система должна предоставлять возможность создания и изменения индивидуального рабочего плана для каждого сотрудника.
10. Система должна вести учет заработной платы сотрудника с учетом следующих факторов: количество отработанных часов из назначенных, выслуга лет в данной компании, отпускные, штрафы, премии и наставничество.
11. В качестве языка программирования используется C#.
12. В качестве основного фреймворка используется ASP .NET Core Web API.
13. В качестве ORM должен использоваться Entity Framework Core.
14. В качестве СУБД используется PostgreSQL.

**2.2. Диаграмма вариантов использования**

На основе требований, предъявляемых к разрабатываемому приложению, были разработаны варианты его использования, которые представлены на рисунке 1.

Изображение выглядит как текст, диаграмма, линия

Автоматически созданное описание

Рисунок 1 – Диаграмма вариантов использования

Список актеров:

* грузчик – сотрудник, выполняющий функции грузчика, являющийся сотрудником склада;
* начальник склада – сотрудник склада, отвечающий за открытие и закрытие смен, контроль выполненной работы и поддержку в актуальном состоянии рабочих планов;
* администратор – сотрудник, отвечающий за поддержание данных системы в актуальном состоянии.

Список прецедентов:

* отметить посещаемость – зафиксировать время своего прибытия на смену;
* редактировать информацию о сотрудниках – изменение информации о сотрудниках (ФИО, паспортные данные, дата начала стажа и т.д.);
* открыть смену – внести запись о том, что смена начата и утвердить список присутствующих на ней сотрудников;
* закрыть смену – внести запись о том, что смена закончена;
* внести комментарий об отработанной сотрудником смене – внесение какой-либо информации об отработанной смене;
* редактировать информацию о должностях – изменение информации о должностях (название, оклад и т.д.);
* редактировать рабочий план сотрудника – внесение изменений о том сколько смен и сколько часов должен отработать сотрудник за месяц;
* получить учет заработных плат – получение информации о том сколько сотрудник заработал за выбранный месяц;
* получить отчет по отработанным сменам – получение информации о том сколько сотрудник отработал часов и просмотр комментариев к сменам, если таковые имеются.

**Вывод**

После изучения существующих решений было принято решение реализовать веб-приложения для транспортной компании для расчета заработной платы сотрудников складов в виде отдельного ASP.NET Core Web API-приложения с использованием Entity Framework Core и PostgreSQL на сервере и Angular на клиенте.

ASP.NET Core Web API – это кросплатформенный фреймворк для создания веб-сервисов (API) на языке программирования C# с использованием технологии ASP.NET Core. Он предназначен для разработки API, которые могут быть использованы различными клиентскими приложениями, такими как веб-приложения, мобильные приложения, настольные приложения и другие.

Entity Framework Core – это ORM (Object-Relational Mapping) фреймворк, разработанный Microsoft для работы с базами данных в приложениях .NET Core. Он позволяет разработчикам работать с данными, используя объектно-ориентированный подход, что упрощает работу с базами данных и уменьшает количество кода, необходимого для взаимодействия с ними.

PostgreSQL [1] (или Postgres) – это реляционная объектно-ориентированная система управления базами данных (СУБД), которая использует язык SQL (Structured Query Language) для управления данными. PostgreSQL является свободно распространяемой и открытой системой, которая может работать на различных операционных системах, включая Linux, Windows, macOS и другие.

Angular – это фреймворк для разработки веб-приложений, основанный на языке TypeScript. Он предоставляет мощные инструменты, такие как компонентная архитектура, двустороннее связывание данных, внедрение зависимостей, модули, роутинг и командная строка Angular CLI. TypeScript придает приложениям статическую типизацию, улучшая производительность и облегчая разработку.

**3. АРХИТЕКТУРА СИСТЕМЫ**

3.1. Общее описание архитектуры системы и шаблонов экранов

При разработке серверной части данного приложения было принято решение придерживаться подхода Domain-driven design. Domain-driven design (DDD) – это подход к проектированию программного обеспечения, который центрирован вокруг модели предметной области (domain model). Это означает, что в DDD мы стремимся создать четкое понимание того, что представляет собой предметная область, которую мы моделируем, и выражаем это понимание в виде явно определенных классов, объектов, связей и операций в нашем коде.

Разработанное в приложение состоит из четырех слоев: Infrustructure, Application, Domain и Presentation (Рисунок 2).

Изображение выглядит как текст, круг, снимок экрана

Автоматически созданное описание

Рисунок 2 – Схема зависимостей слоев приложения

В процессе встреч с product owner’ом был согласован единый шаблон для всех страниц приложения (Рисунок 3).

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 3 – Шаблон рабочего экрана

3.2. Описание компонентов и сервисов, составляющих систему

3.2.1. Слой Presentation

Слой Presentation разработанного приложения отвечает, за взаимодействие пользователей с системой. Данный слой работает с интерфейсом пользователя, который позволяет взаимодействовать с приложением через различные способы, такие как веб-браузер, мобильное приложение или настольное приложение.

В нашем случае взаимодействие с системой будет доступно с помощью следующих наборов конечных точек:

* Набор «Log in», отвечающий за аутентификацию и авторизацию пользователей.
* Набор «Employee», отвечающий за изменение информации о сотрудниках, их увольнение и регистрацию, а также за получение информации о них.
* Набор «Position», отвечающий за изменение информации о должностях, их удаление и создание, а также за получение информации о них.
* Набор «Stock», отвечающий за получение списка складов с основного API компании и за изменение информации о звеньях складов.
* Набор «Shift», отвечающий за открытие и закрытие смен, редактирование и получение информации о них.
* Набор «Attendance», отвечающий за получение табеля посещений.
* Набор «Workplan», отвечающий за создание и редактирование рабочих планов сотрудников.
* Набор «Accounting», отвечающий за получение и редактирование таблицы учета рабочих часов.

3.2.2. Слой Infrastructure

Слой Infrastructure отвечает за надежность, безопасность и производительность приложения во время его взаимодействия с внешними системами и ресурсами.

В нашем приложении данный слой отвечает за генерацию JWT токенов для авторизации пользователей и за обслуживание всех необходимых операций для работы с базой данных проекта.

3.2.3. Слой Application

Слой Application отвечает за реализацию бизнес-логики приложения. Этот слой работает с моделями данных, предоставляет интерфейс для управления бизнес-процессами и обрабатывает запросы от пользователей или других систем.

Основными задачами слоя Application в разработанном нами приложении являются:

* Реализация бизнес-логики: слой Application отвечает за обработку бизнес-правил и реализацию логики приложения, которая описывает, как приложение должно работать с данными и какие действия следует предпринимать при определенных событиях или условиях. Этот слой использует модели данных, которые предоставляются слоем Domain.
* Обеспечение доступа к данным: слой Application обеспечивает доступ к данным, предоставляемым слоем Infrastructure. Он использует слой Infrastructure для работы с базами данных, файловыми системами и другими внешними системами.
* Обработка запросов: слой Application обрабатывает запросы от пользователей или других систем и предоставляет им необходимую информацию. Он использует слой Presentation для отображения данных пользователю.

3.2.4. Слой Domain

Слой Domain отвечает за представление бизнес-логики приложения. Он содержит модели данных, описывающие ключевые сущности и объекты бизнес-процессов, которые используются в приложении.

3.3. Модель базы данных

Ниже представлена диаграмма базы данных проекта, содержащая только ключи (Рисунок 3).

Изображение выглядит как текст, снимок экрана, число

Автоматически созданное описание

Рисунок 3 – Диаграмма базы данных проекта

# ЗАКЛЮЧЕНИЕ

**ЛИТЕРАТУРА**

1. Официальная документация PostgreSQL. [Электронный ресурс] URL: https://www.postgresql.org/docs/ (дата обращения 15.02.2024 г.).
2. Официальная документация по C#. [Электронный ресурс] URL: https://learn.microsoft.com/en-us/dotnet/csharp/ (дата обращения 15.02.2024 г.).
3. Официальный сайт проекта Workday. [Электронный ресурс] URL: https://www.workday.com/ (дата обращения 22.03.2024 г.).
4. Официальный сайт проекта SAP SuccessFactors. [Электронный ресурс] URL: https://www.sap.com/cis/products/hcm/talent-management.html (дата обращения 22.03.2024 г.).
5. Официальный сайт проекта Kronos. [Электронный ресурс] URL: https://www.ukg.com/ (дата обращения 22.03.2024 г.).
6. Официальный сайт проекта TSheets. [Электронный ресурс] URL: https://quickbooks.intuit.com/time-tracking/ (дата обращения 22.03.2024 г.).