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**

**Разработка бэк-энда и API для веб-приложения транспортной компании**

КУРСОВАЯ РАБОТА

по дисциплине «Программная инженерия»

ЮУрГУ – 09.03.04.2023.308-350.КР

|  |  |
| --- | --- |
| Нормоконтролер,к.т.н., доцент кафедры СП  \_\_\_\_\_\_\_\_\_\_\_\_\_\_ М.В. Сухов  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2023 г. | Научный руководитель:  к.т.н., доцент кафедры СП \_\_\_\_\_\_\_\_\_\_ М.В. Сухов  Автор работы:  студент группы КЭ-303  \_\_\_\_\_\_\_\_\_\_ В.А. Рявкин  Работа защищена  с оценкой: \_\_\_\_\_\_\_\_\_\_\_  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2023 г. |

Челябинск, 2023 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

06.02.2023 г.

**ЗАДАНИЕ**

**на выполнение курсовой работы**

по дисциплине «Программная инженерия»

студенту группы КЭ-303

Рявкину Виталию Алексеевичу,

обучающемуся по направлению

09.03.04 «Программная инженерия»

1. **Тема работы**

Разработка бэк-энда и API для веб-приложения транспортной компании.

1. **Срок сдачи студентом законченной работы:** 29.05.2023 г.
2. **Исходные данные к работе**
3. Valerio De Sanctis. Building Web APIs with ASP.NET Core //Manning, 2023.
4. Entity framework documentation. [Электронный ресурс] URL https://learn.microsoft.com/en-us/ef/core/ (дата обращения 15.02.2023 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 15.02.2023 г.).
6. **Перечень подлежащих разработке вопросов3**
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:** 06.02.2023 г.

**Научный руководитель,**

к.т.н., доцент кафедры СП М.В. Сухов **Задание принял к исполнению** В.А. Рявкин

**ОГЛАВЛЕНИЕ**

[ГЛОССАРИЙ 6](#_Toc136360343)

[ВВЕДЕНИЕ 7](#_Toc136360344)

[1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И СУЩЕСТВУЮЩИХ РАБОТ ПО ТЕМАТИКЕ КУРСОВОГО ПРОЕКТА 9](#_Toc136360345)

[1.1. Предметная область проекта 9](#_Toc136360346)

[1.2. Анализ аналогичных проектов и существующих решений для реализации проекта 9](#_Toc136360347)

[1.2.1 Workday 9](#_Toc136360348)

[1.2.2 SAP SuccessFactors 10](#_Toc136360349)

[1.2.3 Kronos 10](#_Toc136360350)

[1.2.4 TSheets 11](#_Toc136360351)

[1.3. Вывод 12](#_Toc136360352)

[2. АНАЛИЗ ТРЕБОВАНИЙ К ПРОГРАММНОЙ СИСТЕМЕ 13](#_Toc136360353)

[2.1. Функциональные требования к проектируемой системе 13](#_Toc136360354)

[2.2. Нефункциональные требования к проектируемой системе 14](#_Toc136360355)

[2.3. Диаграмма вариантов использования 14](#_Toc136360356)

[2.4. Спецификация основных прецедентов 16](#_Toc136360357)

[3. АРХИТЕКТУРА СИСТЕМЫ 20](#_Toc136360358)

[3.1. Общее описание архитектуры системы 20](#_Toc136360359)

[3.2. Описание компонентов и сервисов, составляющих систему 20](#_Toc136360360)

[3.2.1. Слой Presentation 20](#_Toc136360361)

[3.2.2. Слой Infrastructure 21](#_Toc136360362)

[3.2.3. Слой Application 22](#_Toc136360363)

[3.2.4. Слой Domain 22](#_Toc136360364)

[3.3. Модель базы данных 22](#_Toc136360365)

[3.4. Процесс работы с системой 23](#_Toc136360366)

[4. РЕАЛИЗАЦИЯ И ТЕСТИРОВАНИЕ СИСТЕМЫ 25](#_Toc136360367)

[4.1. Реализация компонентов системы 25](#_Toc136360368)

[4.1.1. Реализация слоя Presentation 25](#_Toc136360369)

[4.1.2. Реализация слоя Application 26](#_Toc136360370)

[4.1.3. Реализация слоя Infrastructure 28](#_Toc136360371)

[4.1.4. Реализация слоя Domain 29](#_Toc136360372)

[4.2. Тестирование системы 30](#_Toc136360373)

[ЗАКЛЮЧЕНИЕ 33](#_Toc136360374)

[ЛИТЕРАТУРА 34](#_Toc136360375)

[ПРИЛОЖЕНИЕ 35](#_Toc136360376)

[Приложение А. Описание таблиц базы данных 35](#_Toc136360377)

# ГЛОССАРИЙ

1. Бэк-энд – часть системы, которая выполняет основную обработку данных и не взаимодействует напрямую с пользователем. Включает в себя серверные компоненты, базы данных и другие системы.
2. API – это механизмы, которые позволяют двум программным компонентам взаимодействовать друг с другом, используя набор определений и протоколов.
3. Фреймворк – набор библиотек и инструментов, которые упрощают разработку программного обеспечения, предоставляя структуру и набор стандартных функций.
4. ORM (англ. Object-Relational Mapping) – технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных».

# 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 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 бэк-энда и API веб-приложения транспортной компании 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 39 страниц, объем списка литературы – 7 источников.

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.

В приложении А содержится описание таблиц базы данных.

# 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И СУЩЕСТВУЮЩИХ РАБОТ ПО ТЕМАТИКЕ КУРСОВОГО ПРОЕКТА

1.1. Предметная область проекта

Предметная область проекта связана с разработкой бэк-энда и API веб-приложения транспортной компании 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 и существующих решений для реализации проекта

Существует множество аналогов API для учета посещаемости сотрудников и просчета заработной платы на основе этой посещаемости и других факторов. В качестве примера возьмем некоторые из них: Workday [4], SAP SuccessFactors [5], Kronos Workforce Management [6], TSheets [7].

1.2.1 Workday

Workday [4] – 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 [5] – 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 [6] – 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 [7] – 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.

1.3. Вывод

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

ASP.NET Core Web API [3] – 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 [2] – 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.

# 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 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.

2.2. Нефункциональные требования к проектируемой системе

1. В качестве языка программирования используется C#.
2. В качестве основного фреймворка используется ASP .NET Core Web API.
3. В качестве ORM должен использоваться Entity Framework Core.
4. В качестве СУБД используется PostgreSQL.

2.3. Диаграмма вариантов использования

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

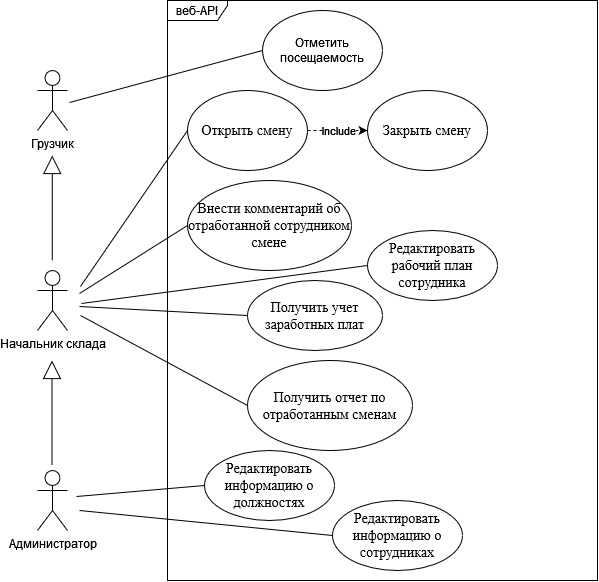


Рисунок – Диаграмма вариантов использования

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.

2.4. Спецификация основных прецедентов

Спецификация прецедентов представлена в таблицах ниже.

Таблица 1 – Спецификация прецедента «Отметить посещаемость»

|  |  |
| --- | --- |
| *Прецедент:* | Отметить посещаемость. |
| *ID:* | ID1. |
| *Аннотация:* | Пользователь системы может отметить свою посещаемость. |
| *Главные актеры:* | Грузчик, начальник склада. |
| *Второстепенные актеры:* | *-* |
| *Предусловия:* | Доступно окно для ввода пароля. |
| *Основной поток:* | 1. Сотрудник вводит свой пароль. 2. Его посещаемость записана. |
| *Постусловия:* | Новый дрон добавлен в систему и доступен для выбора. |
| *Альтернативные потоки:* | Сотрудник неверно ввел пароль – процедура повторяется. |

Таблица 2 – Спецификация прецедента «Редактировать информацию о сотрудниках»

|  |  |
| --- | --- |
| *Прецедент:* | Редактировать информацию о сотрудниках. |
| *ID:* | ID2. |
| *Аннотация:* | Пользователь системы может изменить информацию о личных данных сотрудника. |
| *Главные актеры:* | Администратор. |
| *Второстепенные актеры:* | *-* |
| *Предусловия:* | Есть доступ к окну редактирования информации о сотрудниках. |
| *Основной поток:* | 1. Администратор выбирает сотрудника. 2. Администратор меняет какую-либо личную информацию о сотруднике. |
| *Постусловия:* | Система записала новые данные сотрудника и сохранила старые в истории. |
| *Альтернативные потоки:* | Если внесенные данные некорректны, то система выдаст сообщение о соответствующей ошибке. |

Таблица 3 – Спецификация прецедента «Открыть смену»

|  |  |
| --- | --- |
| *Прецедент:* | Открыть смену. |
| *ID:* | ID3. |
| *Аннотация:* | Пользователь открывает смену. |
| *Главные актеры:* | Администратор, начальник склада. |
| *Второстепенные актеры:* | *-* |
| *Предусловия:* | Есть доступ к окну открытия смены. |
| *Основной поток:* | 1. Пользователь проверяет отмеченных сотрудников. 2. Пользователь открывает смену. |

Окончание таблицы 3

|  |  |
| --- | --- |
| *Постусловия:* | Система записала данные о начале смены и о сотрудниках на ней присутствовавших. |
| *Альтернативные потоки:* | Если на этом складе уже открыта смена, будет выдана ошибка. |

Таблица 4 – Спецификация прецедента «Закрыть смену»

|  |  |
| --- | --- |
| *Прецедент:* | Закрыть смену. |
| *ID:* | *ID*4. |
| *Аннотация:* | Пользователь закрывает смену. |
| *Главные актеры:* | Администратор, начальник склада. |
| *Второстепенные актеры:* | *-* |
| *Предусловия:* | Есть доступ к окну закрытия смены. |
| *Основной поток:* | 1. Пользователь проверяет отработанное сотрудниками время, выставленное системой с фактическим, и меняет его, если оно не сходится. 2. Пользователь закрывает смену. |
| *Постусловия:* | Система записала данные о закрытии смены и о том сколько отработал каждый сотрудник на ней. |
| *Альтернативные потоки:* | Если нет открытых смен, то не будет доступа к закрытию смены. |

Таблица 5 – Спецификация прецедента «Внести комментарий об отработанной сотрудником смене»

|  |  |
| --- | --- |
| *Прецедент:* | Внести комментарий об отработанной сотрудником смене. |
| *ID:* | ID5. |
| *Аннотация:* | Пользователь вносит комментарий о работе сотрудника на смене, если возникает такая необходимость. |
| *Главные актеры:* | Администратор, начальник склада. |
| *Второстепенные актеры:* | *-* |
| *Предусловия:* | Смена закрыта и доступно поле для добавления комментария. |
| *Основной поток:* | 1. Пользователь вводит комментарий о прошедшей смене. |
| *Постусловия:* | Система записала комментарий об отработанной сотрудником смене. |
| *Альтернативные потоки:* | *-* |

Таблица 6 – Спецификация прецедента «Редактировать информацию о должностях»

|  |  |
| --- | --- |
| *Прецедент:* | Редактировать информацию о должностях. |
| *ID:* | ID6. |
| *Аннотация:* | Администратор изменяет информацию о должности. |
| *Главные актеры:* | Администратор. |

Окончание таблицы 6

|  |  |
| --- | --- |
| *Второстепенные актеры:* | *-* |
| *Предусловия:* | Имеется доступ к интерфейсу редактирования должностей. |
| *Основной поток:* | 1. Администратор изменяет необходимые ему поля в должности. |
| *Постусловия:* | Система записала новые сведения о должности и сохранила старые в истории. |
| *Альтернативные потоки:* | Если внесенные данные некорректны, то система выдаст сообщение о соответствующей ошибке. |

Таблица 7 – Спецификация прецедента «Редактировать рабочий план сотрудника»

|  |  |
| --- | --- |
| *Прецедент:* | Редактировать рабочий план сотрудника. |
| *ID:* | ID7. |
| *Аннотация:* | Пользователь изменяет рабочий план сотрудника (сколько часов человек должен отработать за этот месяц). |
| *Главные актеры:* | Администратор, начальник склада. |
| *Второстепенные актеры:* | *-* |
| *Предусловия:* | Имеется доступ к интерфейсу редактирования рабочих планов. |
| *Основной поток:* | 1. Пользователь изменяет рабочий план выбранного сотрудника на выбранный месяц. |
| *Постусловия:* | Система записала новые сведения о рабочем плане и сохранила старые в истории. |
| *Альтернативные потоки:* | Если внесенные данные некорректны, то система выдаст сообщение о соответствующей ошибке. |

Таблица 8 – Спецификация прецедента «Получить учет заработных плат»

|  |  |
| --- | --- |
| *Прецедент:* | Получить учет заработных плат. |
| *ID:* | ID8. |
| *Аннотация:* | Пользователь формирует и получает таблицу учета заработных плат сотрудников склада за период. |
| *Главные актеры:* | Администратор, начальник склада. |
| *Второстепенные актеры:* | *-* |
| *Предусловия:* | Имеется доступ к интерфейсу получения учета заработных плат. |
| *Основной поток:* | 1. Пользователь вносит дополнительные данные, не относящиеся к отработанным часам (отпускные, премии, доплата за наставничество и др.). 2. Пользователь получает итоговую таблицу учета заработных плат. |
| *Постусловия:* | Система записала новые сведения об учете заработных плат, если были какие-то изменения, и сохраняет прошлые в истории. |
| *Альтернативные потоки:* | *-* |

Таблица 9 – Спецификация прецедента «Получить отчет по отработанным сменам»

|  |  |
| --- | --- |
| *Прецедент:* | Получить отчет по отработанным сменам. |
| *ID:* | ID9. |
| *Аннотация:* | Пользователь получает информацию о том сколько сотрудник отработал часов и возможность просмотра комментариев к сменам, если таковые имеются. |
| *Главные актеры:* | Администратор, начальник склада. |
| *Второстепенные актеры:* | *-* |
| *Предусловия:* | Имеется доступ к интерфейсу получения отчета по отработанным сменам. |
| *Основной поток:* | 1. Пользователь получает информацию о том сколько сотрудник отработал часов и возможность просмотра комментариев к сменам, если таковые имеются. |
| *Постусловия:* | Система выдала информацию об отработанных сменах на складе за заданный месяц. |
| *Альтернативные потоки:* | - |

# 3. АРХИТЕКТУРА СИСТЕМЫ

3.1. Общее описание архитектуры системы

При разработке 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).

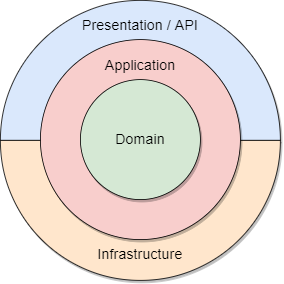


Рисунок – Схема зависимостей слоев приложения

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).

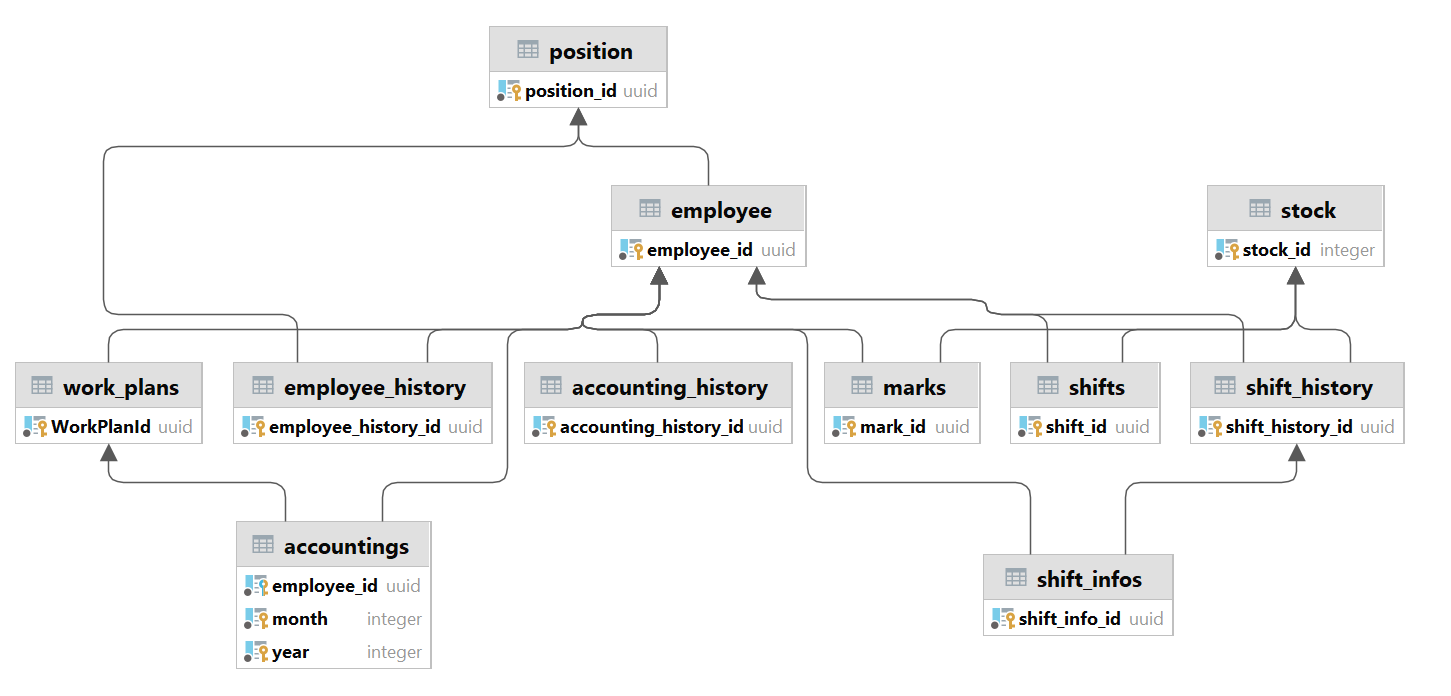


Рисунок – Диаграмма базы данных проекта

Подробная информация о таблицах предоставлена в приложении А.

3.4. Процесс работы с системой

На рисунке 4 приведена диаграмма деятельности, которая подробно описывает процесс работы с API для веб-приложения транспортной компании для расчета заработной платы и контроля работы сотрудников складов.

Процесс начинается с получения запроса с клиентского приложения. Если выполнение данного запроса требует авторизации, то система отправляет ошибку с сообщением о том, что необходимо авторизоваться. В случае если пришел токен доступа, система проверяет его валидность и либо направляет запрос на необходимый контроллер, если токен валиден, либо отправляет на клиент ошибку с сообщением о том, что необходимо авторизоваться повторно. Если авторизация не требуется, то запрос направляется на необходимый контроллер.

После того, как запрос пришел на контроллер он обрабатывается системой и отправляет ответ с результатом на клиент.

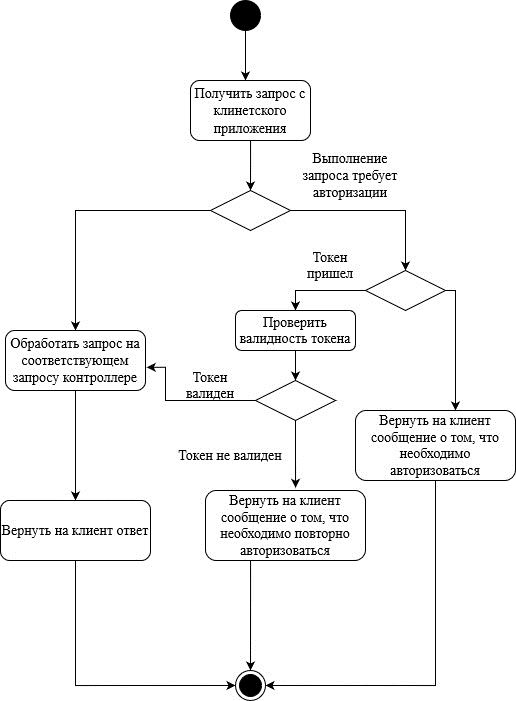


Рисунок – Диаграмма деятельности

# 4. РЕАЛИЗАЦИЯ И ТЕСТИРОВАНИЕ СИСТЕМЫ

4.1. Реализация компонентов системы

4.1.1. Реализация слоя Presentation

Слой Presentation является приложением типа Web API. Является точкой запуска всего приложения и содержит в себе набор контроллеров, которые обрабатывают входящие из сети запросы.

AOWH.API/Program.cs - основной файл, который отвечает за инициализацию и конфигурацию хоста приложения. Он содержит точку входа в приложение и используется для настройки веб-хоста, который будет запускать Web API. Ниже предоставлен листинг кода.

Листинг 1 – Program.cs

|  |
| --- |
| var builder = WebApplication.CreateBuilder(args);  {  builder.Services.AddControllers();  builder.Services.AddEndpointsApiExplorer();  builder.Services.AddSwaggerGen();  builder.Services.AddHttpClient();  builder.Services.AddApplication()  .AddInfrastructure(builder.Configuration);  builder.Services.AddControllersWithViews()  .AddNewtonsoftJson(options =>  options.SerializerSettings.ReferenceLoopHandling = Newtonsoft.Json.ReferenceLoopHandling.Ignore);  builder.Services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)  .AddJwtBearer(options =>  {  var key = Encoding.UTF8.GetBytes(builder.Configuration["JWT:Key"]!);  options.TokenValidationParameters = new TokenValidationParameters  {  ValidateIssuer = true,  ValidateAudience = true,  ValidateLifetime = true,  ValidateIssuerSigningKey = true,  ValidIssuer = builder.Configuration["JWT:Issuer"],  ValidAudience = builder.Configuration["JWT:Audience"],  IssuerSigningKey = new SymmetricSecurityKey(key)  };  });  var mapperConfig = new MapperConfiguration(mc =>  {  mc.AddProfile(new AccountingMap());  mc.AddProfile(new EmployeeMap());  mc.AddProfile(new ShiftMap());  mc.AddProfile(new WorkPlanMap()); |

Окончание листинга 1

|  |
| --- |
| });  IMapper mapper = mapperConfig.CreateMapper();  builder.Services.AddSingleton(mapper);  }  var app = builder.Build();  {  app.UseHttpsRedirection();  app.MapControllers();  app.UseAuthentication();  app.UseAuthorization();  app.Run();  } |

В качестве примера контроллера приведу файл «AccountingController.cs».

Листинг 2 - AccountingController

|  |
| --- |
| [ApiController]  [Route("api/[controller]")]  [Authorize]  public class AccountingController : ControllerBase  {  // Регистрация зависимостей  private readonly IAccountingService \_accountingService;  // Внедрение зависимостей  public AccountingController(IAccountingService accountingService)  // Получение учета заработных плат  [HttpGet]  public async Task<IActionResult> GetAccounting([FromQuery] int year, [FromQuery] int month, [FromQuery] int stockId)  // Обновление учета заработных плат  [HttpPatch]  public async Task<IActionResult> UpdateAccountings([FromBody] List<UpdateAccountingRequest> request)  } |

4.1.2. Реализация слоя Application

Слой Application реализован в виде библиотеки классов. Отвечает за реализацию бизнес-логики приложения, а также предоставляет интерфейсы для взаимодействия со слоем Infrastructure.

Список интерфейсов, предоставляемых слоем Application:

* «ITokenBuilder» ­- предоставляет интерфейс для работы с JWT токенами.
* «IAccountingRepository» - предоставляет интерфейс для работы с репозиторием моделей «Accounting» и «AccountingHistory».
* «IEmployeeRepository» - предоставляет интерфейс для работы с репозиторием моделей «Employee» и «EmployeeHistory».
* «IPositionRepository» - предоставляет интерфейс для работы с репозиторием модели «Position».
* «IShiftRepository» - предоставляет интерфейс для работы с репозиторием моделей «Shift», «ShiftInfo» и «ShoftHistory».
* «IStockRepository» - предоставляет интерфейс для работы с репозиторием модели «Stock».
* «IWorkPlanRepository» - предоставляет интерфейс для работы с репозиторием модели «WorkPlan».

Реализация бизнес-логики представлена в виде сервисов. У каждого из которых присутствует описание того, как должен выглядеть входящий и результирующий объект.

Листинг - Пример входящего объекта

|  |
| --- |
| public class UpdateAccountingRequest  {  public Guid EmployeeId { get; set; }  public int Year { get; set; }  public int Month { get; set; }  public decimal Mentoring { get; set; }  public decimal Teaching { get; set; }  public decimal Bonus { get; set; }  public decimal Vacation { get; set; }  public decimal Advance { get; set; }  } |

Листинг - Пример результирующего объекта

|  |
| --- |
| public class GetAccountingResponse  {  public Guid EmployeeId { get; set; }  public int OvertimeDay { get; set; }  public int OvertimeNight { get; set; }  public decimal SalaryForShift { get; set; } |

Окончание листинга 4

|  |
| --- |
| public decimal SalaryForHour { get; set; }  public decimal Mentoring { get; set; }  public decimal Seniority { get; set; }  public decimal Teaching { get; set; }  public decimal Bonus { get; set; }  public decimal Vacation { get; set; }  public decimal Earned { get; set; }  public decimal Advance { get; set; }  public decimal Penalties { get; set; }  public decimal Sends { get; set; }  public decimal Payment { get; set; }  } |

4.1.3. Реализация слоя Infrastructure

Слой Infrastructure реализует предоставленные слоем Application интерфейсы.

Листинг 5 - Пример реализации интерфейса ITokenBuilder

|  |
| --- |
| public class TokenBuilder : ITokenBuilder  {  private readonly IConfiguration \_config;  public TokenBuilder(IConfiguration config)  {  \_config = config;  }  public string BuildToken(Guid employeeId)  {  var credentials = new SigningCredentials(  new SymmetricSecurityKey(Encoding.UTF8.GetBytes(\_config["JWT:Key"]!)),  SecurityAlgorithms.HmacSha256);  var claims = new[]  {  new Claim("EmployeeId", employeeId.ToString())  };  var token = new JwtSecurityToken(  \_config["Jwt:Issuer"]!,  \_config["Jwt:Audience"]!,  claims,  expires: DateTime.Now.AddMinutes(Convert.ToDouble(\_config["Jwt:ExpireTime"]!)),  signingCredentials: credentials);    return new JwtSecurityTokenHandler().WriteToken(token);  }  public RefreshToken GenerateRefreshToken() |

Окончание листинга 5

|  |
| --- |
| {  return new RefreshToken  {  Token = Convert.ToBase64String(RandomNumberGenerator.GetBytes(64)),  Expires = DateTime.UtcNow.AddMinutes(30)  };  }  public ClaimsPrincipal? GetPrincipalForExpiredToken(string token, IConfiguration config)  {  var tokenValidationParameters = new TokenValidationParameters  {  ValidateAudience = false,  ValidateIssuer = false,  ValidateIssuerSigningKey = true,  IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(config["JWT:Key"]!)),  ValidateLifetime = false  };  var tokenHandler = new JwtSecurityTokenHandler();  var principal = tokenHandler.ValidateToken(token, tokenValidationParameters, out SecurityToken securityToken);  if (securityToken is not JwtSecurityToken jwtSecurityToken || !jwtSecurityToken.Header.Alg.Equals(SecurityAlgorithms.HmacSha256, StringComparison.InvariantCultureIgnoreCase))  throw new SecurityTokenException("Invalid token");  return principal;  }  } |

4.1.4. Реализация слоя Domain

В слое Domain находится описание ключевых сущностей предметной области: «Accounting», «AccountingHistory», «Employee», «EmployeeHistory», «InterfaceAccesses», «Mark», «Position», «Shift», «ShiftHistory», «ShiftInfo», «Stock», «WorkPlan».

Листинг 6 - Пример модели «WorkPlan»

|  |
| --- |
| public class WorkPlan  {  public Guid WorkPlanId { get; set; }  public int Month { get; set; }  public int Year { get; set; }  public Guid EmployeeId { get; set; }  public int NumberOfDayShifts { get; set; }  public int NumberOfHoursPerDayShift { get; set; }  public int NumberOfNightShifts { get; set; }  public int NumberOfHoursPerNightShift { get; set; }  [JsonIgnore]  public Employee? Employee { get; set; }  [JsonIgnore] |

Окончание листинга 6

|  |
| --- |
| public virtual ICollection<Accounting> Accountings { get; } = new List<Accounting>();  } |

4.2. Тестирование системы

В ходе разработки API были разработаны и проведены комплексные тесты для проверки функциональности и стабильности работы системы. Проведенные тесты относятся к виду функционального тестирования.

В качестве основного инструментом тестирования было выбрано приложение Postman.

Тестирование проводилось в соответствии с предложенной методологией, при которой каждый из тестовых сценариев включал в себя конкретные шаги для воспроизведения действий пользователя, а также определенный ожидаемый результат. В процессе тестирования внимание было уделено как отдельным функциям, так и взаимодействию между различными компонентами системы.

Процесс тестирования осуществлялся с использованием различных входных данных, включая граничные значения и невалидные данные, чтобы оценить устойчивость и надежность системы в различных условиях, а также чтобы выявить и устранить возможные недостатки. Примеры тестов приведены в таблице 10.

Таблица 10 – Тестирование системы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Название теста** | **Шаги** | **Ожидаемый результат** | **Тест пройден?** |
| 1 | Авторизация | 1. Отправить запрос с методом POST на uri /api/login с паролем сотрудника в теле | Пользователю должен вернуться JWT токен, а в базу данных и cookie браузера занестись refresh-токен и время его истечения. | да |

Продолжение таблицы 10

| **№** | **Название теста** | **Шаги** | **Ожидаемый результат** | **Тест пройден?** |
| --- | --- | --- | --- | --- |
| 2 | Добавление сотрудника в систему | 1. Отправить запрос с методом POST на uri /api/Employee со всеми необходимыми данными о сотруднике в теле | Клиенту возвращается ответ со статусом 200, в теле которого содержится информация о сотруднике, которую только что внесли в систему. В базе данных появился новый сотрудник | да |
| 3 | Получение информации о всех сотрудниках, зарегистрированных в системе | 1. Отправить запрос с методом GET на uri /api/Employee | Клиенту возвращается ответ со статусом 200, в теле которого содержится краткая информация о всех сотрудниках, зарегистрированных в системе. | Да |
| 4 | Некорректная попытка обновления информации о сотруднике | 1. Отправить запрос с методом PUT на uri /api/Employee/{employeeId} со всеми необходимыми данными о сотруднике в теле и несуществующим id сотрудника | Клиенту возвращается ответ со статусом 404 и сообщением о том, что заданного сотрудника не существует. | Да |
| 5 | Создание рабочего плана для сотрудника | 1. Отправить запрос с методом POST на uri /api/workplan с необходимыми параметрами и данными о сменах и количестве часов в них в теле. | Клиенту возвращается ответ со статусом 200. В базе данных появляется запись о новом рабочем плане за заданный месяц. | Да |
| 6 | Открытие смены | 1. Отправить запрос с методом POST на uri /api/shift/open с данными о том на каком складе открыта смен, дневная она или ночная и списком сотрудников, которые должны заступить на смену. | Клиенту возвращается ответ со статусом 200 и id открытой смены. В базе данных появляется запись о новой открытой смене. | Да |

Окончание таблицы 10

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Название теста** | **Шаги** | **Ожидаемый результат** | **Тест пройден?** |
| 7 | Закрытие смены | 1. Отправить запрос с методом POST на uri /api/shift/close с данными о том сколько часов отработали сотрудники на смене. | Клиенту возвращается ответ со статусом 200. Смена закрывается и сохраняется в таблицу с историями. | да |
| 8 | Закрытие смены с неотмеченным сотрудником | 1. Отправить запрос с методом POST на uri /api/shift/close с данными о том сколько часов отработали сотрудники на смене, но не отмечать присутствие сотрудника перед этим. | Клиенту возвращается ответ со статусом 500 и сообщением, что сотрудник из списка не был отмечен. | да |
| 9 | Получение посещаемости склада за выбранный месяц | 1. Отправить запрос с методом  GET на uri /api/attendance с данными о том за какой месяц, и на каком складу мы хотим узнать посещаемость. | Клиенту возвращается ответ со статусом 200 и тело, в котором содержится краткая информация о сотруднике и отработанные им смены. | да |
| 10 | Получение таблицы учета заработных плат сотрудников склада за выбранный месяц | 1. Отправить запрос с методом  GET на uri api/accounting с данными о том за какой месяц, и о каком складе мы хотим получить таблицу учета заработных плат. | Клиенту возвращается ответ со статусом 200 и тело, в котором содержится краткая информация о том сколько полагается выплатить сотрудникам выбранного склада. | да |

# ЗАКЛЮЧЕНИЕ

В рамках данной работы был разработан бэк-энд и API для веб-приложения транспортной компании для учета заработной платы и контроля качества работы сотрудников складов.

Был проведен анализ предметной области и изучены существующие решения, связанные с учетом заработной платы и контролем качества работы сотрудников.

* + - 1. Была разработана архитектура приложения, обеспечивающая его масштабируемость и простоту внесения изменений в будущем.
      2. Был спроектирован и разработан API для возможности взаимодействия со сторонними системами.
      3. Было проведено тестирование приложения, включая проверку корректности расчетов и работы пользовательского интерфейса.

# ЛИТЕРАТУРА

1. Официальная документация PostgreSQL. [Электронный ресурс] URL: https://www.postgresql.org/docs/ (дата обращения 15.02.2023 г.).
2. Официальная документация по Entity Framework. [Электронный ресурс] URL: https://learn.microsoft.com/en-us/aspnet/core/fundamentals/apis?view=aspnetcore-7 (дата обращения 15.02.2023 г.).
3. Официальная документация по C#. [Электронный ресурс] URL: https://learn.microsoft.com/en-us/dotnet/csharp/ (дата обращения 15.02.2023 г.).
4. Официальный сайт проекта Workday. [Электронный ресурс] URL: https://www.workday.com/ (дата обращения 22.03.2023 г.).
5. Официальный сайт проекта SAP SuccessFactors. [Электронный ресурс] URL: https://www.sap.com/cis/products/hcm/talent-management.html (дата обращения 22.03.2023 г.).
6. Официальный сайт проекта Kronos. [Электронный ресурс] URL: https://www.ukg.com/ (дата обращения 22.03.2023 г.).
7. Официальный сайт проекта TSheets. [Электронный ресурс] URL: https://quickbooks.intuit.com/time-tracking/ (дата обращения 22.03.2023 г.).

# ПРИЛОЖЕНИЕ

Приложение А. Описание таблиц базы данных

Таблица 11 – Поля таблицы «positon»

|  |  |
| --- | --- |
| name | varchar(255) |
| salary | numeric(10, 2) |
| quarterly\_bonus | numeric(10, 2) |
| interface\_accesses | jsonb |
| interface\_accesses | jsonb |
| position\_id | uuid, pk |

Таблица 12 – Поля таблицы «stock»

|  |  |
| --- | --- |
| name | varchar(255) |
| links | jsonb |
| stock\_id | integer, pk |

Таблица 13 – Поля таблицы «employee»

|  |  |
| --- | --- |
| Password | integer |
| name | varchar(255) |
| surname | varchar(255) |
| patronymic | varchar(255) |
| birthday | date |
| passport\_number | varchar(255) |
| passport\_issuer | varchar(255) |
| passport\_issue\_date | date |
| start\_of\_total\_seniority | date |
| start\_of\_luch\_seniority | date |
| date\_of\_termination | date |
| position\_id | uuid, fk |
| date\_of\_start\_in\_the\_current\_position | date |
| salary | numeric(10, 2) |
| quarterly\_bonus | numeric(10, 2) |

Окончание таблицы 13

|  |  |
| --- | --- |
| percentage\_of\_salary\_in\_advance | Integer |
| Link | varchar(255) |
| DateOfStartInTheCurrentLink | Date |
| stocks | Jsonb |
| date\_of\_start\_in\_the\_current\_stock | Date |
| forklift\_control | Boolean |
| rolleys\_control | Boolean |
| refresh\_token | Text |
| refresh\_token\_expires | timestamp with time zone |
| employee\_id | uuid, pk |

Таблица 14 ­­­­­­– Поля таблицы «employee\_history»

|  |  |
| --- | --- |
| employee\_id | uuid, fk |
| Name | varchar(255) |
| Surname | varchar(255) |
| Patronymic | varchar(255) |
| Birthday | date |
| PassportNumber | varchar(255) |
| PassportIssuer | varchar(255) |
| PassportIssueDate | date |
| StartOfTotalSeniority | date |
| StartOfLuchSeniority | date |
| DateOfTermination | date |
| position\_id | uuid, fk |
| start\_date\_of\_work\_in\_current\_position | date |
| end\_date\_of\_work\_in\_current\_position | date |
| salary | numeric(10, 2) |
| quarterly\_bonus | numeric(10, 2) |
| percentage\_of\_salary\_in\_advance | integer |
| link | varchar(255) |

Окончание таблицы 14

|  |  |
| --- | --- |
| StartDateOfWorkInCurrentLink | date |
| EndDateOfWorkInCurrentLink | date |
| stocks | jsonb |
| start\_date\_of\_work\_in\_stock | date |
| end\_date\_of\_work\_in\_stock | date |
| ForkliftControl | boolean |
| RolleysControl | boolean |
| DateOfCreation | timestamp with time zone |
| employee\_history\_id | uuid, pk |

Таблица 15 – Поля таблицы «accountings»

|  |  |
| --- | --- |
| work\_paln\_id | uuid |
| overtime\_day | integer |
| overtime\_night | integer |
| salary\_for\_shift | numeric(10, 2) |
| salary\_for\_hour | numeric(10, 2) |
| mentoring | numeric(10, 2) |
| seniority | numeric(10, 2) |
| teaching | numeric(10, 2) |
| bonus | numeric(10, 2) |
| vacation | numeric(10, 2) |
| earned | numeric(10, 2) |
| advance | numeric(10, 2) |
| penalties | numeric(10, 2) |
| sends | numeric(10, 2) |
| payment | numeric(10, 2) |
| employee\_id | uuid, pk |
| month | intger, pk |
| year | integer, pk |

Таблица 16 – Поля таблицы «accounting\_history»

|  |  |
| --- | --- |
| employee\_id | uuid, fk |
| year | integer |
| month | integer |
| mentoring | numeric(10, 2) |
| teaching | numeric(10, 2) |
| bonus | numeric(10, 2) |
| vacation | numeric(10, 2) |
| advance | numeric(10, 2) |
| mentoring\_prev | numeric(10, 2) |
| teaching\_prev | numeric(10, 2) |
| bonus\_prev | numeric(10, 2) |
| vacation\_prev | numeric(10, 2) |
| advance\_prev | numeric(10, 2) |
| update\_time | timestamp with time zone |
| accounting\_history\_id | uuid, pk |

Таблица 17 – Поля таблицы «work\_plans»

|  |  |
| --- | --- |
| month | integer |
| year | integer |
| employee\_id | uuid, fk |
| number\_of\_day\_shifts | integer |
| number\_of\_hours\_per\_day\_shift | integer |
| number\_of\_night \_shifts | integer |
| number\_of\_hours\_per\_night\_shift | integer |
| work\_plan\_id | uuid |

Таблица 18 – Поля таблицы «shifts»

|  |  |
| --- | --- |
| stock\_id | integer, fk |
| employee\_who\_posted\_the\_shift\_id | uuid, fk |
| day\_or\_night | varchar(255) |
| opening\_date | timestamp with time zone |
| employees | jsonb |

Окончание таблицы 18

|  |  |
| --- | --- |
| closing\_date\_and\_time | timestamp with time zone |
| last\_update | timestamp with time zone |
| shift\_id | uuid, pk |

Таблица 19 – Поля таблицы «shift\_history»

|  |  |
| --- | --- |
| stock\_id | integer, fk |
| employee\_who\_posted\_the\_shift\_id | uuid, fk |
| day\_or\_night | varchar(255) |
| opening\_date | timestamp with time zone |
| employees | jsonb |
| closing\_date\_and\_time | timestamp with time zone |
| last\_update | timestamp with time zone |
| shift\_id | uuid, pk |

Таблица 20 – Поля таблицы «shift\_infos»

|  |  |
| --- | --- |
| shift\_history\_id | uuid, fk |
| employee\_id | uuid, fk |
| date\_and\_time\_of\_arrival | timestamp with time zone |
| day\_or\_night | varchar(255) |
| number\_of\_hours\_worked | integer |
| penalty | numeric |
| penalty\_comment | varchar(255) |
| send | numeric |
| send\_comment | varchar(255) |
| shift\_info\_id | uuid, pk |

Таблица 21 – Поля таблицы «marks»

|  |  |
| --- | --- |
| stock\_id | integer, fk |
| employee\_id | uuid, fk |
| mark\_date | timestamp with time zone |
| mark\_id | uuid, pk |