Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет информационных технологий и управления

Кафедра интеллектуальных информационных технологий

ОТЧЁТ

по лабораторной работе №2 по дисциплине

# ПРОЕКТИРОВНИЕ БАЗ ЗНАНИЙ

Вариант №4

|  |  |
| --- | --- |
| Студент гр. 221703 | Д.Н. Вечорко |
| Руководитель | И.А. Ерофеев |

Минск 2024

# ПОСТАНОВКА ЗАДАЧИ

1.1 Словесное описание предметной области

Технолог междугородней телефонной станции (МТС) вносит в БД информацию о новых клиентах, пользующихся услугами, это: номер телефона клиента, ФИО, адрес, дата регистрации. Также технолог изменяет цены за пользование услугами, эти данные хранятся в следующем виде: дата, название населенного пункта, с которым осуществляется связь, стоимость одной минуты разговора, льготная стоимость (с 20.00 до 6.00). Оператор МТС фиксирует дату разговора, город, с которым произошел разговор, номер абонента, длительность разговора. После этого оператор отсылает абоненту квитанцию об оплате. Он также принимает извещение об оплате и фиксирует, что те или иные разговоры оплачены.

1.2 Требования к приложению

Необходимо реализовать выполнения следующих функций:

* Добавление/редактирование/удаление информации о клиентах телефонной станции.
* Добавление/редактирование/удаление информации о стоимости услуг связи.
* Добавление/редактирование/удаление информации о телефонном звонке (функция оператора МТС).
* Расчет количества абонентов, соединившихся с выбранным городом за выбранный месяц.
* Вывести для просмотра стоимость оплаты одной минуты разговора для разных населенных пунктов на заданную дату: название организации, предоставляющей услуги связи - дата, название населенного пункта, стоимость одной минуты, льготная стоимость одной минуты.

# КОНЦЕПТУАЛЬНОЕ ПРОЕКТИРОВАНИЕ

2.1 Варианты использования

Система имеет два основных пользователя – технолог и оператор МТС.

Технолог вносит в базу данных информацию о новом клиенте (номер телефона, ФИО, адрес, дата регистрации). Также технолог обновляет данные о тарифах, включая дату, населённый пункт, стоимость одной минуты разговора, и льготную стоимость.

Оператор МТС фиксирует информацию о разговоре записывая дату, город, номер абонента и длительность разговора. Кроме того оператор может отсылать квитанцию об оплате, а также получать подтверждение оплаты от абонента и фиксировать, что разговор оплачен. (рис. 3.1).



Рисунок 2.1 – Диаграмма вариантов использования

2.2 Отношения сущностей

В базе данных определено четыре таблицы: стоимость, клиент, звонок и квитанция.

(рис. 3.2).

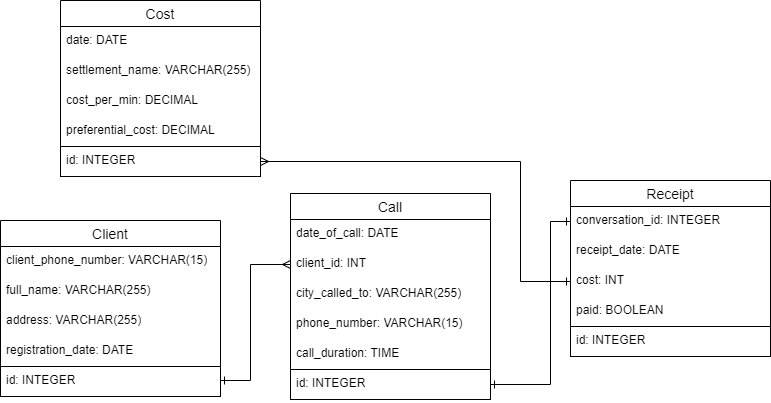


Рисунок 2.2 – ER-диаграмма

# ЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ

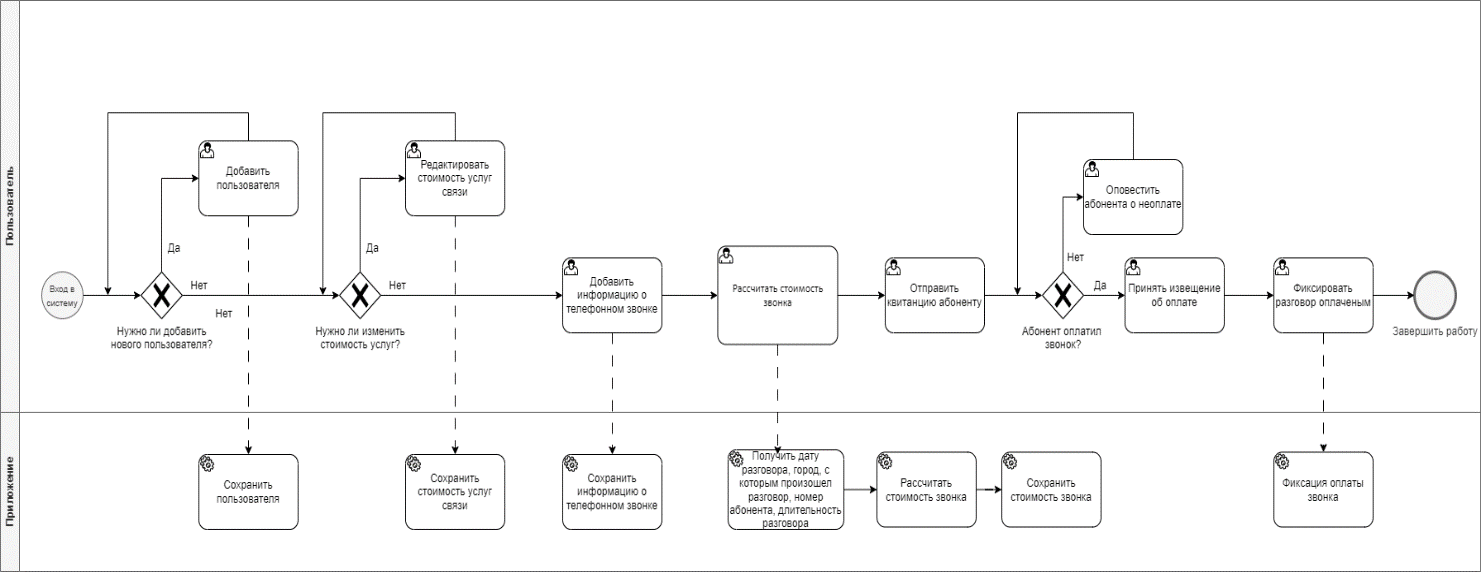


Рисунок 3.1 – BPMN-диаграмма

# ФИЗИЧЕСКОЕ ПРОЕКТИРОВАНИЕ

4.1 Схема базы данных

4.1.1 Таблица стоимости

CREATE TABLE cost( id INT PRIMARY KEY GENERATED ALWAYS AS IDENTITY,

date: DATE,

settlement\_name: VARCHAR(255) NOT NULL,

cost\_per\_min: DECIMAL NOT NULL,

preferential\_cost: DECIMAL

);

4.1.2 Таблица клиентов

CREATE TABLE client( id INT PRIMARY KEY GENERATED ALWAYS AS IDENTITY,

date: DATE,

full\_name: VARCHAR(255) NOT NULL,

address: VARCHAR(255),

registration\_date: DATE NOT NULL

);

4.1.3 Таблица звонков

CREATE TABLE call( id INT PRIMARY KEY GENERATED ALWAYS AS IDENTITY,

client\_id: INT NOT NULL,

date\_of\_call: TIMESTAMP

city\_called\_to: VARCHAR(255) NOT NULL,

phone\_number: VARCHAR(15) NOT NULL,

call\_duration: TIME NOT NULL

CONSTRAINT unique\_client\_call\_time UNIQUE (client\_id, date\_of\_call), FOREIGN KEY (client\_id) REFERENCES Client(id)

);

4.1.4 Таблица квитанций

CREATE TABLE receipt( id INT PRIMARY KEY GENERATED ALWAYS AS IDENTITY,

conversation\_id: INTEGER,

receipt\_date: DATE,

cost: INT NOT NULL,

paid: BOOLEAN NOT NULL,

FOREIGN KEY (cost\_id) REFERENCES Cost(id),

FOREIGN KEY (conversation\_id) REFERENCES Call (id) ON DELETE CASCADE

);

# РЕАЛИЗАЦИЯ

5.1 Тексты запросов

5.1.1 Получение всех клиентов

SELECT \* FROM Client;

5.1.2 Получение конкретного клиента

SELECT \* FROM Client

WHERE id = {client\_id}

5.1.3 Создание нового клиента

INSERT INTO client(full\_name, address, registration\_date)

VALUES ({full\_name}, {address}, {registration\_date)

5.1.4 Изменение клиента

UPDATE client SET full\_name = '{ full\_name }', address = '{ address }', registration\_date = '{ registration\_date}'

WHERE id = {client\_id}

5.1.5 Удаление клиента

DELETE FROM client WHERE id = {client\_id}

5.1.6 Получение всех стоимостей

SELECT \* FROM cost

5.1.7 Получение конкретной стоимости

SELECT \* FROM cost WHERE id = {cost\_id}

5.1.8 Добавление новой стоимости

INSERT INTO cost (date, settlement\_name, cost\_per\_min, preferential\_cost) VALUES ('{ date }', { settlement\_name }, {cost\_per\_min}', { preferential\_cost})

5.1.9 Изменение стоимости

UPDATE cost SET date = '{date}', settlement\_name = {settlement\_name}, cost\_per\_min = '{ cost\_per\_min }', preferential\_cost = { preferential\_cost } WHERE id = {cost\_id}

5.1.10 Удаление стоимости

DELETE FROM cost WHERE id = {cost\_id}

5.1.10 Получение всех телефонных звонков

SELECT ca.id, client\_id, date\_of\_call, city\_called\_to, phone\_number, call\_duration

FROM call ca JOIN client cl ON ca.client\_id = cl.id

5.1.11 Получение конкретного телефонного звонка

SELECT ca.id, client\_id, date\_of\_call, city\_called\_to, phone\_number, call\_duration

FROM call ca JOIN client cl ON ca.client\_id = cl.id

WHERE ca.id = {call\_id}

5.1.12 Создание нового телефонного звонка

INSERT INTO call (client\_id, date\_of\_call, city\_called\_to, phone\_number, call\_duration)

VALUES ({client\_id}, {date\_of\_call}, {city\_called\_to}, {phone\_number}, {call\_duration})

5.1.13 Изменение телефонного звонка

UPDATE cost SET client\_id = '{client\_id}', date\_of\_call = {date\_of\_call}, city\_called\_to = '{city\_called\_to}', call\_duration = {call\_duration} WHERE id = {call\_id}

5.1.14 Удаление телефонного звонка

DELETE FROM call WHERE id = {call\_id}

5.1.15 Расчет количества абонентов, соединившихся с выбранным городом за выбранный месяц с разбивкой по дням.

CREATE OR REPLACE FUNCTION get\_daily\_subscriber\_count(p\_city VARCHAR, p\_month DATE)

RETURNS TABLE (

call\_day DATE,

unique\_subscriber\_count INT

) AS $$

BEGIN

RETURN QUERY

SELECT

date\_of\_call AS call\_day,

COUNT(DISTINCT client\_id) AS unique\_subscriber\_count

FROM

Call

WHERE

city\_called\_to = p\_city

AND date\_of\_call >= p\_month

AND date\_of\_call < (p\_month + INTERVAL '1 month')

GROUP BY

date\_of\_call

ORDER BY

call\_day;

END;

$$

5.1.16 Вывести для просмотра стоимость оплаты одной минуты разговора для разных населенных пунктов на заданную дату: название организации, предоставляющей услуги связи - дата, название населенного пункта, стоимость одной минуты, льготная стоимость одной минуты.

CREATE OR REPLACE FUNCTION get\_costs\_by\_date(p\_date DATE)

RETURNS TABLE (

settlement\_name VARCHAR,

cost\_date DATE,

standard\_cost\_per\_min DECIMAL,

discounted\_cost\_per\_min DECIMAL

) AS $$

BEGIN

RETURN QUERY

SELECT

settlement\_name,

date AS cost\_date,

cost\_per\_min,

preferential\_cost

FROM

Cost

WHERE

date = p\_date;

END;

$$

# ИНТЕРФЕЙС ПРИЛОЖЕНИЯ

Интерфейс приложения реализован в виде оконного приложения.

6.1 Интерфейс для взаимодействия с работниками

При входе на страницу Workers, пользователю отображается таблица всех работников, включающая их ID, фамилию, имя, отчество, должность, статус нахождения в профсоюзе и разряд (рис. 6.1).

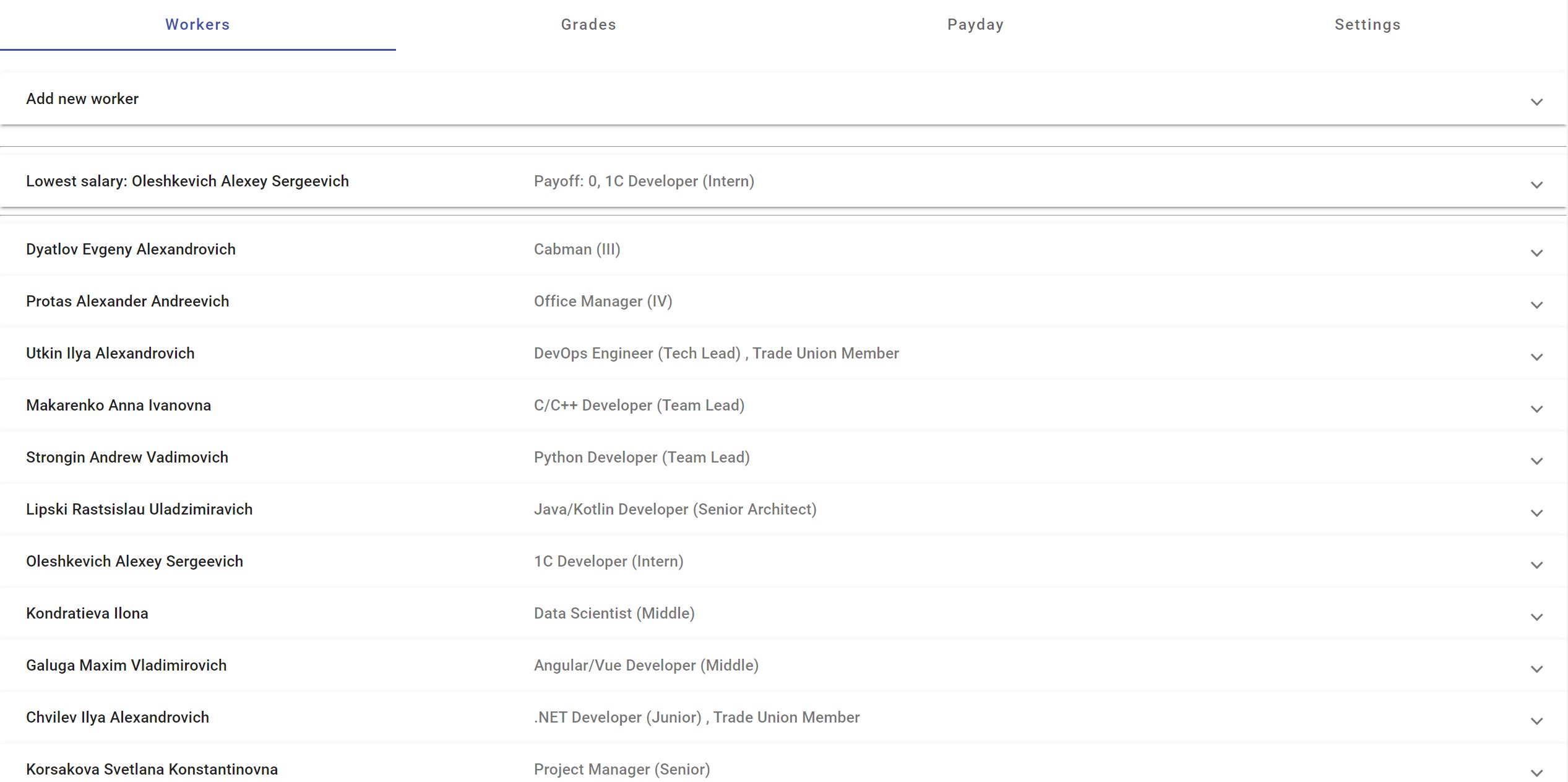


Рисунок 6.1 – Получение всех работников

При нажатии на кнопку Add new worker, у пользователя запрашиваются фамилия, имя, отчество, должность, статус нахождения в профсоюзе и разряд нового работника, после чего новый работник сохраняется в базу данных (рис. 6.2).



Рисунок 6.2 – Добавление нового работника

При нажатии на работника, у пользователя запрашиваются новые фамилия, имя, отчество, должность, статус нахождения в профсоюзе и разряд работника. Также можно удалить работника. После ввода всех данных, изменения сохраняются в базу данных (рис. 6.3).

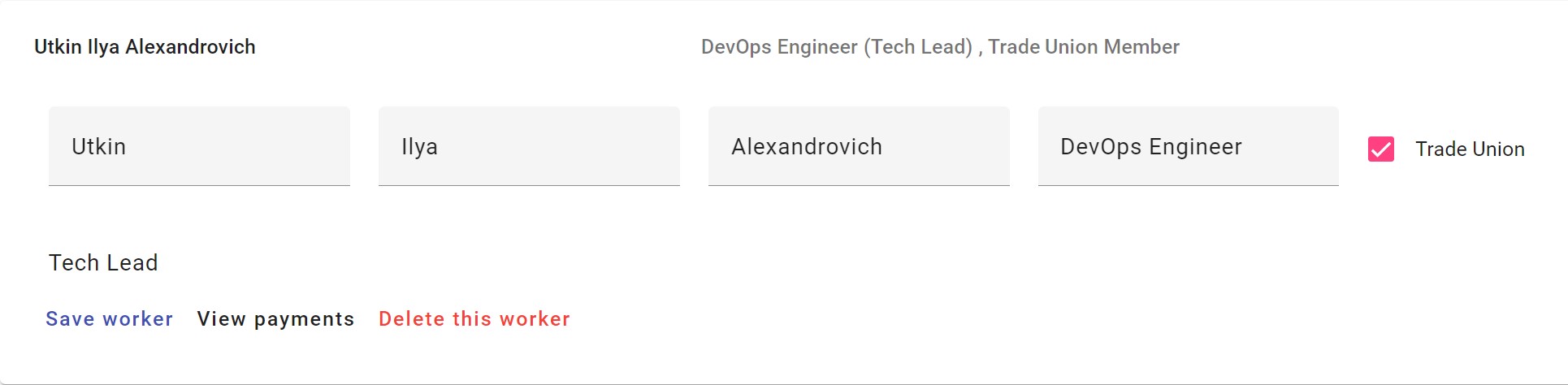


Рисунок 6.3 – Изменение работника

6.2 Интерфейс для взаимодействия с разрядами

При входе на страницу Grades, пользователю отображается таблица всех разрядов, включающая в себя название и коэффициент разряда (рис.

1. 4).

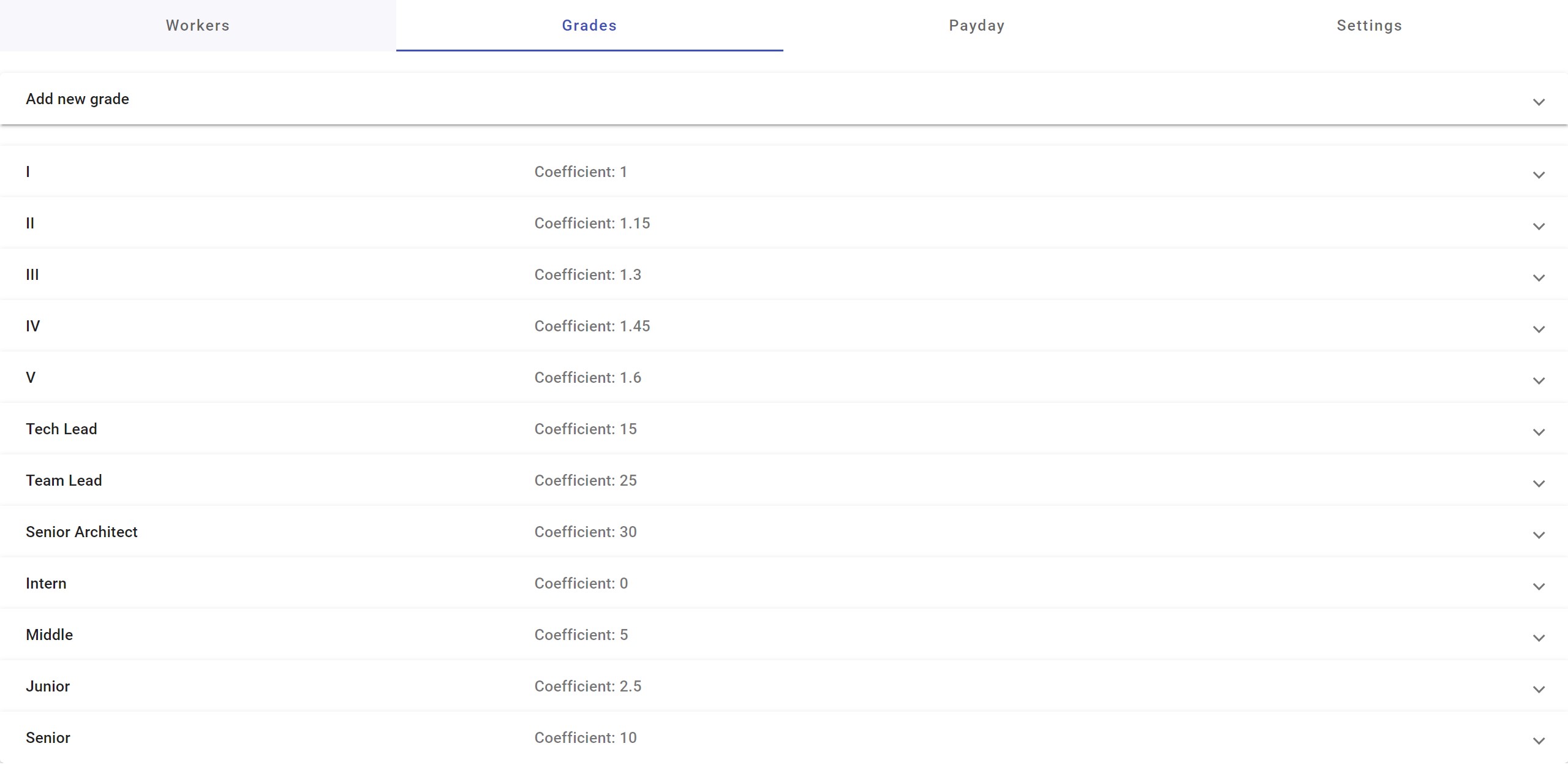


Рисунок 6.4 – Получение всех разрядов

При нажатии на кнопку Add new grade, у пользователя запрашивается название разряда и его коэффициент для расчёта заработной платы. После ввода данных, новый разряд сохраняется в базу данных (рис. 6.5).

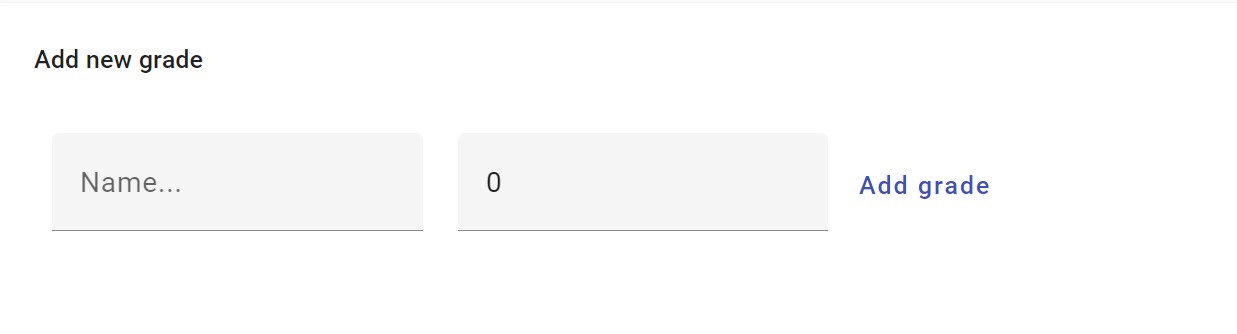


Рисунок 6.5 – Добавление нового разряда

При нажатии на разряд, у пользователя запрашиваются новое название и коэффициент для расчёта заработной платы. Также есть возможность удалить разряд. После ввода всех даннных, изменения сохраняются в базу данных (рис. 6.6).

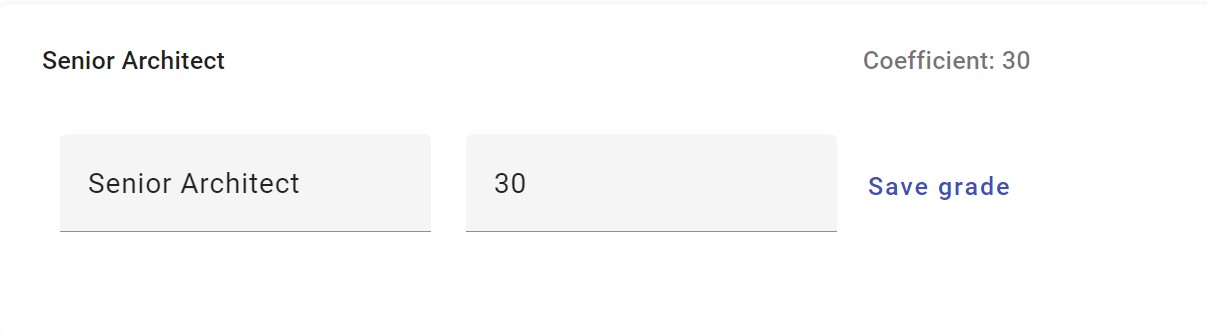


Рисунок 6.6 – Редактирование разряда

6.3 Интерфейс для взаимодействия с выплатами

При переходе на вкладку Payday, у пользователя запрашивается месяц, для которого необходимо рассчитать зарплату, после чего рассчитываются выплаты для всех работников в базе данных (рис. 6.7).

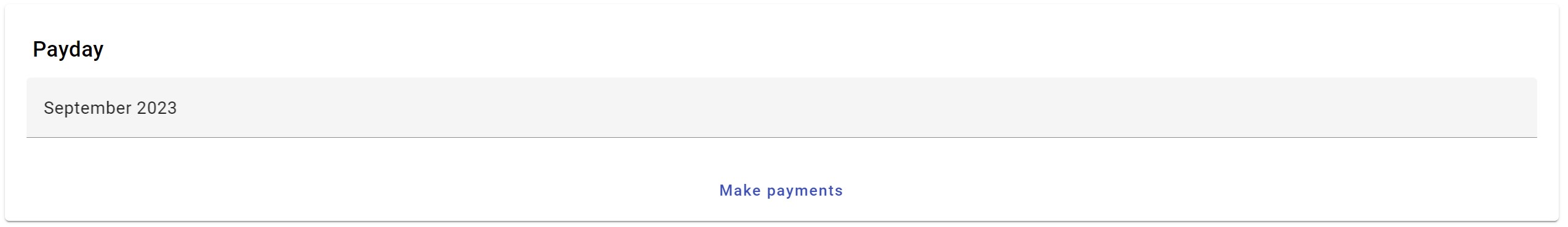


Рисунок 6.7 – Расчёт выплат

При нажатии кнопки View Payments отображается таблица с фамилией, именем, должностью, статусом нахождения в профсоюзе, ID платежа, месяцем платежа, зарплатой, надбавкой, подоходным налогом, вычетом в пенсионный фонд, вычетом в пользу профсоюза и конечной выплатой (рис.

6.8).

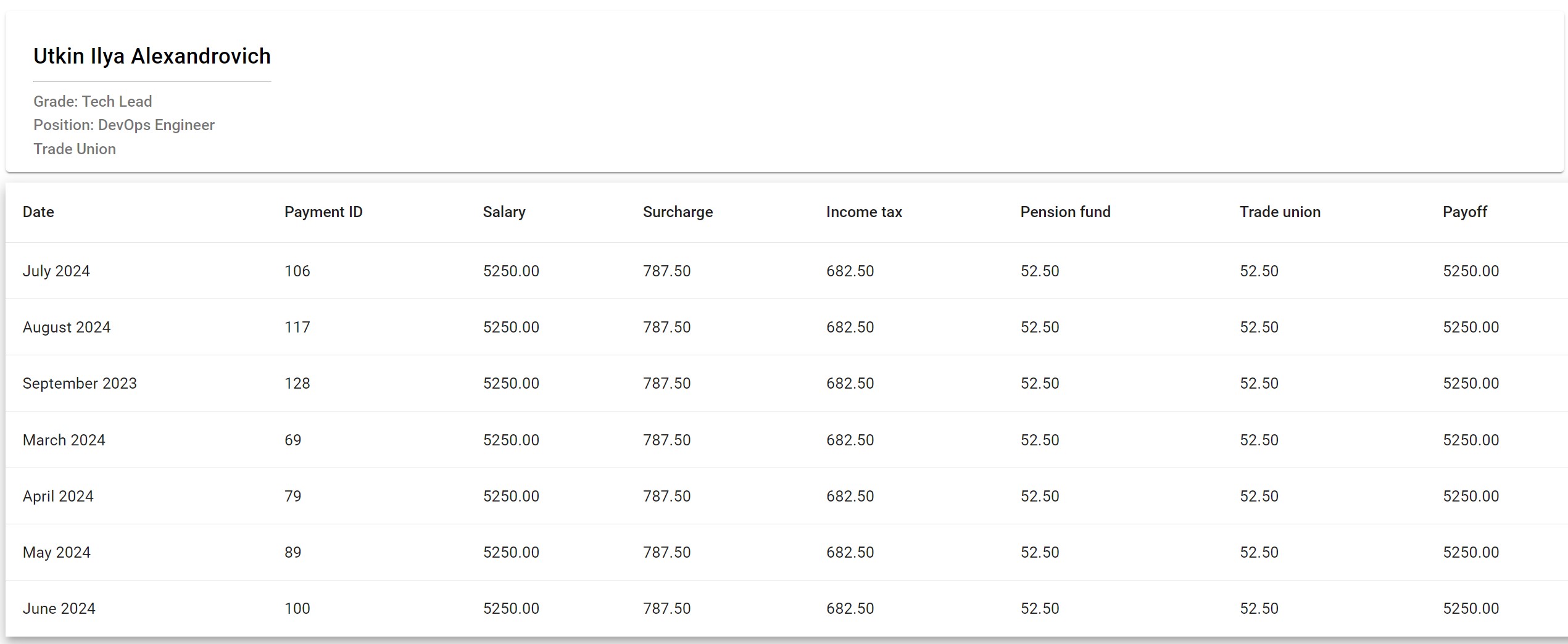


Рисунок 6.8 – Получение выплат работника

При нажатии на кнопку Lowest salary, у пользователя запрашивается месяц, для которого необходимо найти работников, после чего отображаются работники с наименьшей выплатой в этот месяц, включающая фамилию, имя, отчество, статус нахождения в профсоюзе, разряд и выплату.

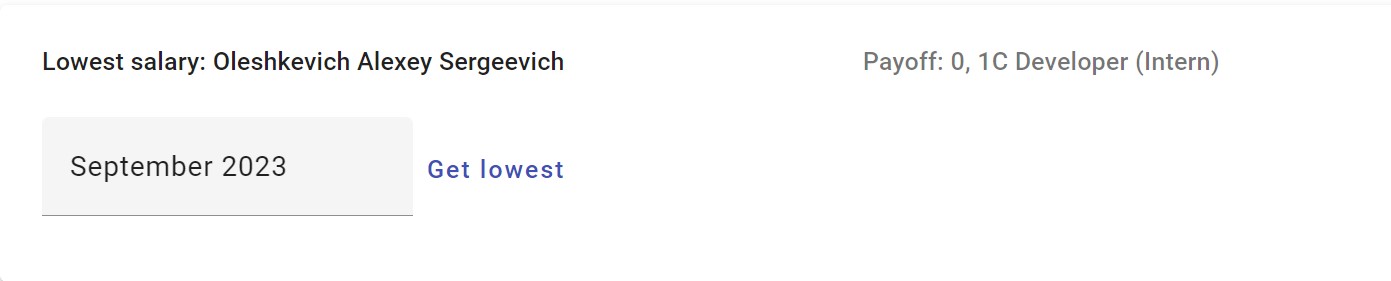


Рисунок 6.9 – Получение работников с наименьшей выплатой в месяце

# ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА, ИСПОЛЬЗОВАННЫЕ ПРИ РАЗРАБОТКЕ

7.1 PostgreSQL

В качестве базы данных для реализации приложения была выбрана PostgreSQL - одна из наиболее популярных реляционных баз данных на данный момент. Её преимуществами являются открытый исходный код, большое сообщество, возможность интеграции с большинством современных языков программирования, а также наличие качественной документации.

7.2 Docker, Docker Compose

Для удобства запуска и конфигурации PostgreSQL использовалась технология контейнеризации Docker, которая позволяет быстро и надежно запускать любые программные системы в любых условиях. Преимуществами Docker является кросс-платформенность и большое количество обучающих материалов. При помощи Docker возможно быстро установить и настроить PostgreSQL (или любое иное программное обеспечение).

7.3 Java

Для разработки интерфейса приложения использовался язык Python версии 3.11. Выбор данного языка программирования обусловен его высокоуровневостью, удобством в разработке и поддержке, наличии большого количества обучающих материалов по нему в сети Интернет.

Внимания также заслуживает огромная коллекция библиотек для Python: например, psycopg2, которая позволяет удобно интегрировать код на Python с базой данных PostgreSQL, а также tabulate, предоставляющая инструменты для отображения таблиц в режиме командой строки.