

МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

Одеський національний політехнічний університет

Інститут комп'ютерних систем

Кафедра інформаційних систем

КУРСОВА РОБОТА

з дисципліни «Технології створення програмних продуктів»

Тема: «CRM система для менеджерів»

Студента III курсу групи AI-184у

спеціальності 122 – «Комп'ютерні науки»

Борисова Володимира Олександровича

Керівник:

к.т.н. доц. каф. ІС

Бабіч Микола Іванович

Національна шкала _____

Кількість балів: _____

Оцінка: ECTS _____

Члени комісії:

(підпис)

(прізвище та ініціали)

(підпис)

(прізвище та ініціали)

(підпис)

(прізвище та ініціали)

м. Одеса – 2020 рік

Одеський національний політехнічний університет

Інститут комп'ютерних систем

Кафедра інформаційних систем

ЗАВДАННЯ

НА КУРСОВУ РОБОТУ

Борисов Володимир Олександрович

AI-184y

(прізвище, ім'я, по-батькові)

(група)

1. Тема роботи: CRM система для менеджерів
2. Термін здачі студентом закінченої роботи _____
3. Мета – аналізуючи потреби можливого, створити просте і зручне сховище клієнтів табору активного туризму, а також угод з ними, для спрощення і автоматизації роботи менеджерів.

На виході готує клієнт-серверний додаток, що буде цікавий туристичній фірмам, які спеціалізуються на організації лісових походів, зимових експедицій, гірських вилазках і т.д.

4. Зміст пояснювальної записки (перелік питань, які належить розробити): зміст, вступ, основна частина (з обов'язковими посиланнями на відповідні літературні джерела), висновки та перелік використаної літератури.

Завдання видано

Завдання прийнято до виконання

АНОТАЦІЯ

Метою курсової роботи є поглиблення, узагальнення і закріплення теоретичних і практичних знань з дисципліни «Технології створення програмних продуктів». У цій роботі розглянуті принципи розробки клієнт-серверного додатка, аналізу ринку, планування процесу розробки. На підставі даних принципів і наявних технічних вимог була розроблена CRM система, яка задовольняє потребам користувачів

АННОТАЦИЯ

Целью курсовой работы является углубление, обобщение и закрепление теоретических и практических знаний по дисциплине «Технологии создания программных продуктов». В этой работе рассмотрены принципы разработки клиент-серверного приложения, анализа рынка, планирование процесса разработки. На основании данных принципов и имеющихся технических требований была разработана CRM система, которая удовлетворяет потребностям пользователей

ANNOTATION

The aim of the course work is to deepen, generalize and consolidate theoretical and practical knowledge in the discipline "Technology of software development". This paper considers the principles of client-server application development, market analysis, development process planning. Based on these principles and available technical requirements, a CRM system has been developed that meets the needs of users.

ЗМІСТ

- 1 Вимоги до програмного продукту
 - 1.1 Визначення потреб споживача
 - 1.1.1 Ієрархія потреб споживача
 - 1.1.2 Деталізація матеріальної потреби
 - 1.2 Бізнес-вимоги до програмного продукту
 - 1.2.1 Опис проблеми споживача
 - 1.2.1.1 Концептуальний опис проблеми споживача
 - 1.2.1.2 Метричний опис проблеми споживача
 - 1.2.2 Мета створення програмного продукту
 - 1.2.2.1 Проблемний аналіз існуючих програмних продуктів
 - 1.2.2.2 Мета створення програмного продукту
 - 1.2.3 Назва програмного продукту
 - 1.2.3.1 Гасло програмного продукту
 - 1.2.3.2 Логотип програмного продукту
 - 1.3 Вимоги користувача до програмного продукту
 - 1.3.1 Історія користувача програмного продукту
 - 1.3.2 Діаграма прецедентів програмного продукту
 - 1.3.3 Сценарії використання прецедентів програмного продукту
 - 1.4 Функціональні вимоги до програмного продукту -
 - 1.4.1. Багаторівнева класифікація функціональних вимог
 - 1.4.2 Функціональний аналіз існуючих програмних продуктів
 - 1.5 Нефункціональні вимоги до програмного продукту
 - 1.5.1 Опис зовнішніх інтерфейсів
 - 1.5.1.1 Опис інтерфейса користувача
 - 1.5.1.1.1 Опис INPUT-інтерфейса користувача
 - 1.5.1.1.2 Опис OUTPUT-інтерфейса користувача
 - 1.5.1.2 Опис інтерфейсу із зовнішніми пристроями

- 1.5.1.3 Опис програмних інтерфейсів
- 1.5.1.4 Опис інтерфейсів передачі інформації
- 1.5.1.5 Опис атрибутів продуктивності
- 2 Планування процесу розробки програмного продукту
 - 2.1 Планування ітерацій розробки програмного продукту
 - 2.2 Концептуальний опис архітектури програмного продукту
 - 2.3 План розробки програмного продукту
 - 2.3.1 Оцінка трудомісткості розробки програмного продукту
 - 2.3.2 Визначення дерева робіт з розробки програмного продукту
 - 2.3.3 Графік робіт з розробки програмного продукту
 - 2.3.3.1 Таблиця з графіком робіт
 - 2.3.3.2 Діаграма Ганта
- 3 Проектування програмного продукту
 - 3.1 Концептуальне та логічне проектування структур даних програмного продукту
 - 3.1.1 Концептуальне проектування на основі UML-діаграми концептуальних класів
 - 3.1.2 Логічне проектування структур даних
 - 3.2 Проектування програмних класів
 - 3.3 Проектування алгоритмів роботи методів програмних класів
 - 3.4 Проектування тестових наборів методів програмних класів
- 4 Конструювання програмного продукту
 - 4.1 Особливості конструювання структур даних
 - 4.1.1 Особливості інсталяції та роботи з СУБД
 - 4.1.2 Особливості створення структур даних
 - 4.2 Особливості конструювання програмних модулів
 - 4.2.1 Особливості роботи з інтегрованим середовищем розробки

4.2.2 Особливості створення програмної структури з урахуванням спеціалізованого Фреймворку

4.2.3 Особливості створення програмних класів

4.2.4 Особливості розробки алгоритмів методів програмних класів або процедур/функцій

4.3 Тестування програмних модулів

5 Розгортання та валідація програмного продукту

5.1 Інструкція з встановлення програмного продукту

5.2 Інструкція з використання програмного продукту

5.3 Результати валідації програмного продукту

Висновки до курсової роботи

1 Вимоги до програмного продукту

1.1 Визначення потреб споживача

1.Ієрархія потреб споживача

Відомо, що в теорії маркетингу потреби людини можуть бути представлені у вигляді ієрархії потреб ідей американського психолога Абрахама Маслоу включають рівні:

- 1 фізіологія (вода, їжа, житло, сон);
- 2 безпека (особиста, здоров'я, стабільність),
- 3 приналежність (спілкування, дружба, любов),
- 4 визнання (повага оточуючих, самооцінка),
- 5 самовираження (вдосконалення, персональний розвиток).



Рисунок 1.1 – Піраміда потреб Маслоу

Даний програмний продукт задовольняє потреби рівня пізнання в піраміді Маслоу.

1.2 Бізнес вимоги до ПП

1.2.1 Опис проблем користувача

1.2.1.1 Концептуальний опис проблеми споживача

Менеджери табору активного туризму та спорту незадоволені відсутністю єдиного середовища для зручної роботи.

1.2.1.2 Метричний опис проблеми користувача

Таблиця 1.1 – Параметри незадоволеності

№	Параметри незадоволеності
1	відсутність єдиного середовища для зручної роботи

Відсоток помилок EP (EP – Error percent) можна визначити як

$$EP = E / N,$$

де

E – кількість помилок під час роботи;

N – загальна кількість роботи

1.2.2 Мета створення ПП

1.2.2.1 Проблемний аналіз існуючих ПП

Таблиця 1.2 – Аналіз існуючих ПП

№	Назва	Вартість	Ступінь	Примітка
1	uCRM	Безкоштовно	2	Немає можливості централізованого обліку усіх клієнтів
2	CRM-tour	Безкоштовно	1	Немає можливості централізованого обліку усіх клієнтів

1.2.2.2 Мета створення ПП

Мета – аналізуючи потреби сможивача, створити просте і зручне сховище клієнтів табору активного туризму, а також угод з ними, для спрощення і автоматизації роботи менеджерів

1.2.3 Назва програмного продукту

Назва повинна бути короткою та проста для запам'ятовування. Тому командою була обрана назва ПП “Експедиція”.

1.2.3.1 Гасло програмного продукту

“Експедиция длиною в жизнь”

1.2.3.2 Логотип програмного продукту



Рисунок 1.2 – Лого ПП “Експедиція”

1.3 Вимоги користувача до ПП

1.3.1 Історія користувача ПП

У ПП задумано 2 основні ролі: керівник, менеджер

User-stories

Як менеджер я хочу:

- Логінитися в свій аккаунт
- Редагувати свій профіль
- Створювати угоди і клієнтів

- Бачити сторінку зі списком угод / батьків / дітей з основною інформацією.
- Мати сторінку-профіль з детальною інформацією угоди / батька / дитини
- Переходити від профілю батька до профілю дитини, до угоди. Мати посилання один на одного
- редагувати угоду
- Редагувати інформацію про батьку, дитині.
- видаляти угоду
- Фільтр по пошуку батьків, дітей, угод. Можливо шукати тільки прив'язаних до мене або по всій базі.

Як керівник я хочу:

- Мати всі функції ролі "Менеджер"
- Мати можливість створювати менеджера

1.3.2 Діаграма прецедентів програмного продукту

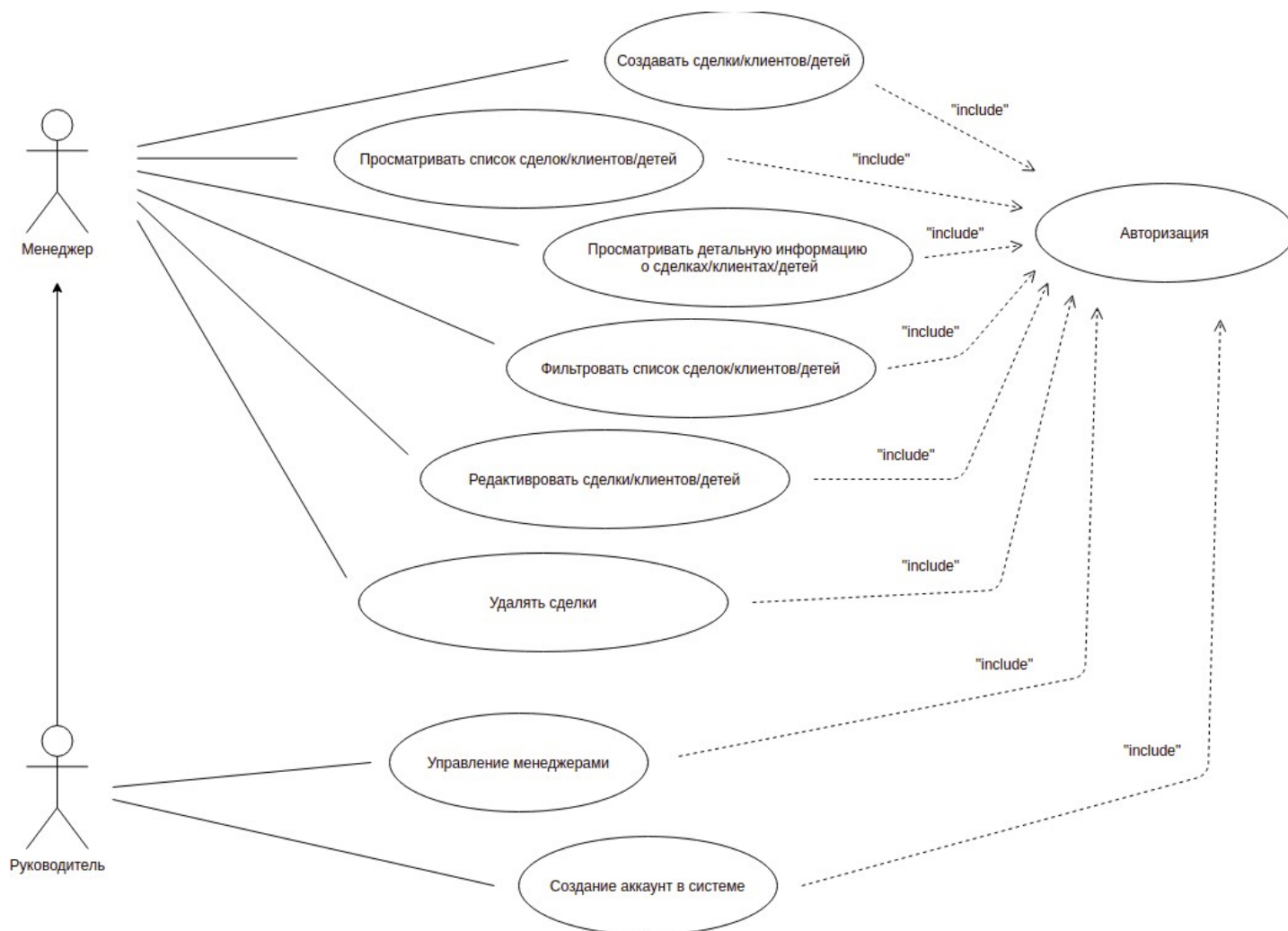


Рисунок 1.3 – Діаграма прецедентів ПП

1.3.3 Сценаріїв використання прецедентів програмного продукту

Альтернативний сценарій для всіх прецедентів:

- Користувач не має доступу до розділу системи
- ПП виводить повідомлення про помилку

Опис сценарію "Авторизація"

Умови початку прецеденту: користувач зайшов в систему НЕ авторизованим

Гарантії успіху: авторизація користувача

Успішний сценарій:

1. ПП надає форму для заповнення
2. Користувач вводить username / password
3. ПП авторизує користувача

Альтернативний сценарій:

1. ПП не отримує дані від користувача або отримує невірні дані
2. ПП виводить помилку для користувача

Опис сценарію "Створення угод / клієнтів / дітей"

Умови початку прецеденту: користувач зайшов в систему авторизованим

Гарантії успіху: створення суті угода / клієнт / дитина

Успішний сценарій:

1. ПП надає форму для заповнення полів
2. Користувач вводить дані про угоду / клієнта / дитину
3. ПП створює сутність з отриманими даними

Альтернативний сценарій:

1. ПП отримує не повні дані

2. ПП виводить помилку для користувача

Опис сценарію "Перегляд списку угод / клієнтів / дітей"

Умови початку прецеденту: користувач зайшов в систему авторизованим

Гарантії успіху: перегляд списку угод / клієнтів / дітей

Успішний сценарій:

1. ПП надає список з основною інформацією по кожній сутності
2. Користувач переглядає список

Опис сценарію "Перегляд детальної інформації про угоду / клієнта / дитину"

Умови початку прецеденту: користувач зайшов в систему авторизованим

Гарантії успіху: перегляд детальної інформації угод / клієнтів / дітей

Успішний сценарій:

1. Користувач клацає на елемент списку угод / клієнтів / дітей
2. ПП показує профайл з детальною інформацією про обраної суті.

Опис сценарію "Фільтрація списку"

Умови початку прецеденту: користувач зайшов в систему авторизованим

Гарантії успіху: користувач отримав відфільтрований список

Успішний сценарій:

1. Користувач вводить необхідні фільтри списку
2. ПП показує список відповідний параметрам фільтра

Опис сценарію "Видалення угоди"

Умови початку прецеденту: користувач зайшов в систему авторизованим

Гарантії успіху: користувач видалив угоду

Успішний сценарій:

1. Користувач клацає кнопку видалення
2. ПП показує діалогове вікно "Ви впевнені що хочете видалити угоду?"
3. Користувач дає / не дає згоду на видалення

Опис сценарію "Створення аккаунта менеджера"

Умови початку прецеденту: керівник зайшов в систему авторизованим

Актори: керівник

Гарантії успіху: керівник створив аккаунта менеджера

Успішний сценарій:

1. Керівник клацає кнопку створення менеджера
2. ПП надає поля введення необхідної інформації про менеджера
3. Керівник вводить дані
4. ПП створює сутність менеджер з отриманими даними

1.4 Функціональні вимоги до ПП

1.4.1 Багаторівнева класифікація функціональних вимог

Таблиця 1.3 – Функціональні вимоги до ПП

Ідентифікатор функції	Функція
FR 1	Авторизація
FR 1.1	Створення запиту для користувача на отримання його параметрів аутентифікації
FR 1.2	Передача параметрів аутентифікації на сервер
FR 1.3	Перевірка даних параметрів
FR 1.4	Передача інформації про помилки на стороні клієнта
FR 2	Створення сутностей менеджер / угода / клієнт / дитина
FR 2.1	Користувач вводить необхідні дані в поля введення надані графічним інтерфейсом
FR 2.2	Передача даних з полів введення на сервер
FR 2.3	Створення суті з наданими даними
FR 2.4	Відображення успішної операції в UI
FR 3	Перегляд списку менеджерів / угод / клієнтів / дітей
FR 3.1	Відображення списку даних отриманих з сервера
FR 4	Перегляд детальної інформації сутностей
FR 4.1	Клік на елемент списку

FR 4.2	Відображення детальної інформації про обраної суті
FR 5	Фільтрація списку сутностей
FR 5.1	Введення користувачем параметрів фільтрації
FR 5.2	Обробка параметрів фільтра на сервері
FR 5.3	Відображення відфільтрованого списку в UI
FR 6	Редагування суті менеджера / угоди / клієнта / дитини
FR 6.1	Клік по кнопці "редагування" на елементі списку або у вікні детальної інформації
FR 6.2	Відображення вікна з полями для введення нових даних сутності
FR 6.3	Відправлення нових даних суті на сервер
FR 6.4	Зміна даних на сервері і відображення змін в UI
FR 7	Видалення суті
FR 7.1	Клік по кнопці "видалення"
FR 7.2	Відображення діалогового вікна з можливістю видалити контакт
FR 7.3	Клік по кнопці підтвердження і видалення на сервері обраної суті

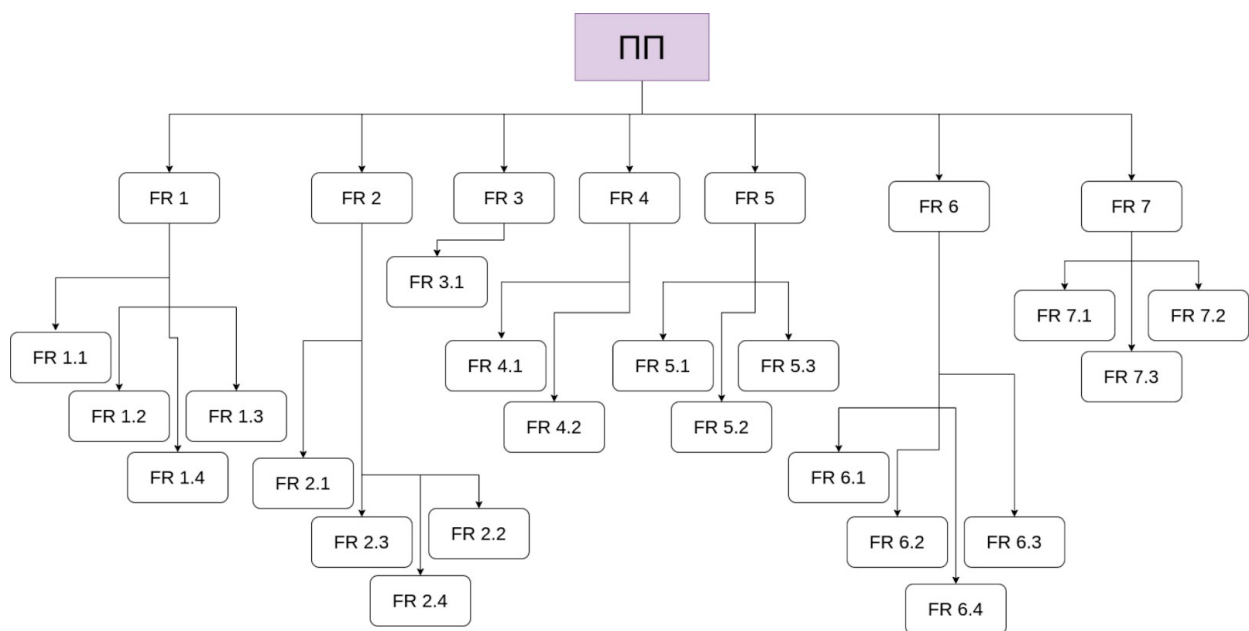


Рисунок 1.4 – WBS структура вимог

1.5 Нефункціональні вимоги до ПП

1.5.1 Опис зовнішніх інтерфейсів

1.5.1.1 Опис інтерфейса користувача

1.5.1.1.1 Опис INPUT-інтерфейса користувача

Таблиця 1.4 – INPUT-інтерфейси

Ідентифікатор функції	Спосіб INPUT	Особливості
FR 1	Стандартна клавіатура, маніпулятор типу «миша»	
FR 2	Стандартна клавіатура, маніпулятор типу «миша»	Поля введення з автодоповнення спрощують процес введення
FR 3	Маніпулятор типу «миша»	

FR 4	Маніпулятор типу «миша»	Посилання на пов'язані сутностей, що спрощує роботу функціональності
FR 5	Стандартна клавіатура, маніпулятор типу «миша»	
FR 6	Стандартна клавіатура, маніпулятор типу «миша»	
FR 7	Маніпулятор типу «миша»	

1.5.1.1.2 Опис OUTPUT-потоків

Таблиця 1.5 – Засоби OUTPUT-потоків

Ідентифікатор функції	Тип потоку
FR 1	Графічний інтерфейс

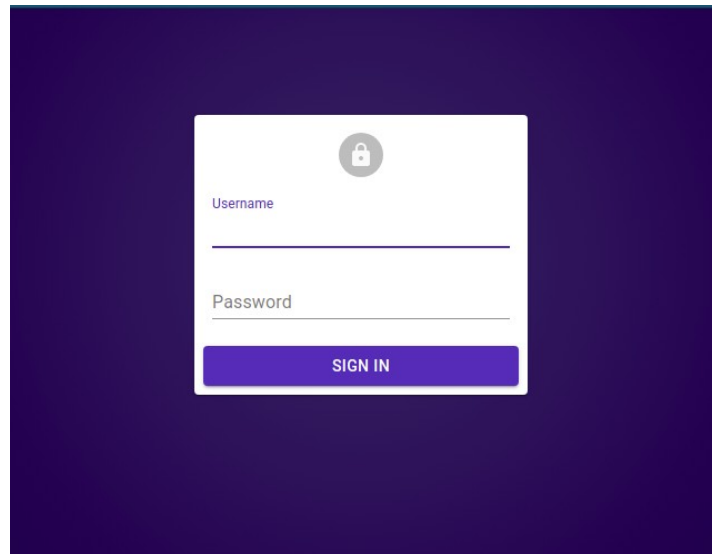


Рисунок 1.5 – OUTPUT-поток FR 1

Таблиця 1.6 – Засоби OUTPUT-потоків

Ідентифікатор функції	Тип потоку
FR 1.4	Графічний інтерфейс

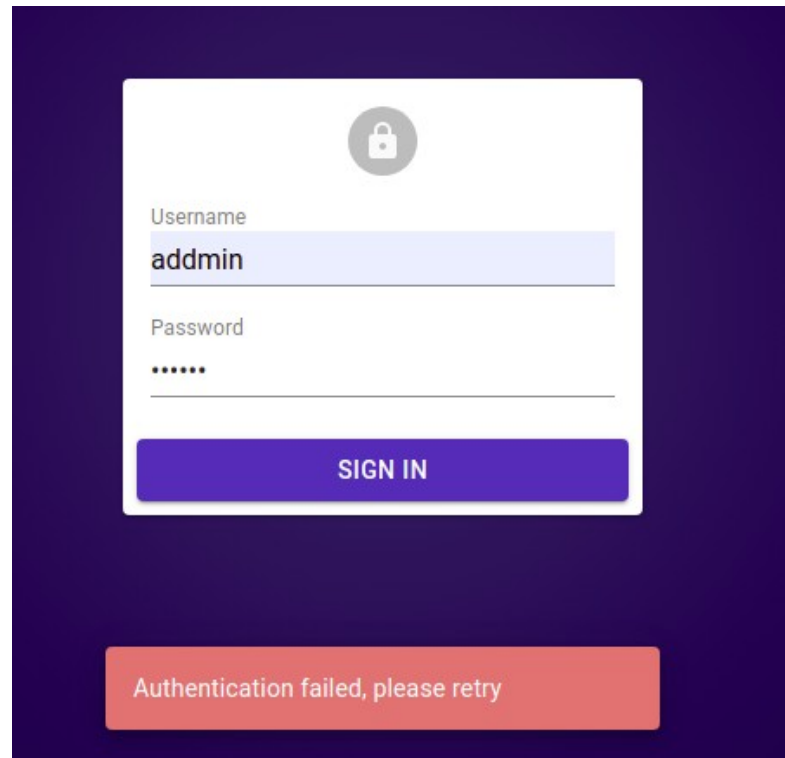


Рисунок 1.6 – OUTPUT-поток FR 1.4

Таблиця 1.7 – Засоби OUTPUT-потоків

Ідентифікатор функції	Тип потоку
FR 2	Графічний інтерфейс

Create Customer

Last name

Иванов

First name

Иван

Patronymic

Иванович

City

Одесса

Status

Parent

Exemptions

Normal

B

I

U

Комментарий-комментарий-комментарий

SAVE

Рисунок 1.7 – OUTPUT-поток FR 2

Таблица 1.8 – Засоби OUTPUT-потоків

Ідентифікатор функції	Тип потоку
FR 3	Графічний інтерфейс

Children

Dashboard

Employees

Customers

Children

Deals

Expeditions

Payments

Search

ADD FILTER

CREATE

EXPORT

<input type="checkbox"/>	Id ↑	Full name	Birthday	Parent	Actions
<input type="checkbox"/>	1	Струнов Олег Максимович	01.08.2017	Борисов Владимир Александрович	<div>EDITDELETE</div>
<input type="checkbox"/>	2	Лапченко Сергей Владимирович	10.04.2009	Борисов Владимир Александрович	<div>EDITDELETE</div>
<input type="checkbox"/>	3	Иванов Иван Иванович	09.04.2008		<div>EDITDELETE</div>

Rows per page: 10 1-3 of 3

Рисунок 1.8 – OUTPUT-поток FR 3

Таблиця 1.9 – Засоби OUTPUT-потоків

Ідентифікатор функції	Тип потоку
FR 4	Графічний інтерфейс

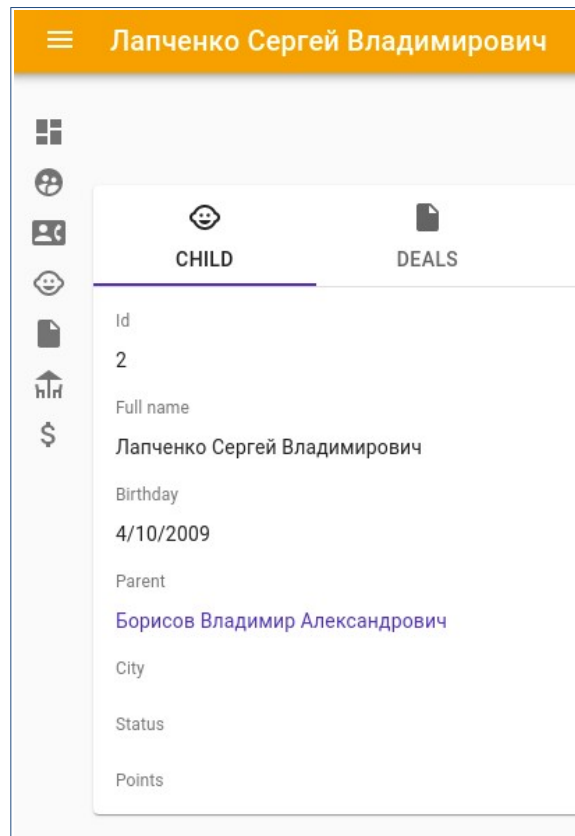


Рисунок 1.9 – OUTPUT-поток FR 4

Таблиця 1.10 – Засоби OUTPUT-потоків

Ідентифікатор функції	Тип потоку
FR 5	Графічний інтерфейс

Children

Search

Age between (6-13)

Parent: Борисов Владимир Александр

<input type="checkbox"/>	Id ↑	Full name	Birthday	Parent	Actions
<input type="checkbox"/>	2	Лапченко Сергей Владимирович	10.04.2009	Борисов Владимир Александрович	EDIT DELETE

Rows per page: 10 1-1 of 1

Рисунок 1.10 – OUTPUT-поток FR 5

Таблиця 1.11 – Засоби OUTPUT-потоків

Ідентифікатор функції	Тип потоку
FR 6	Графічний інтерфейс

Employee #3

Last name: Борисов

First name: Александр

Patronymic: Владимирович

Birthday: 2020-04-09

Phone: 0985925572

Salary: 1000

Email: vovbor

SAVE

Рисунок 1.11 – OUTPUT-поток FR 6

Таблиця 1.12 – Засоби OUTPUT-потоків

Ідентифікатор функції	Тип потоку
FR 7	Графічний інтерфейс

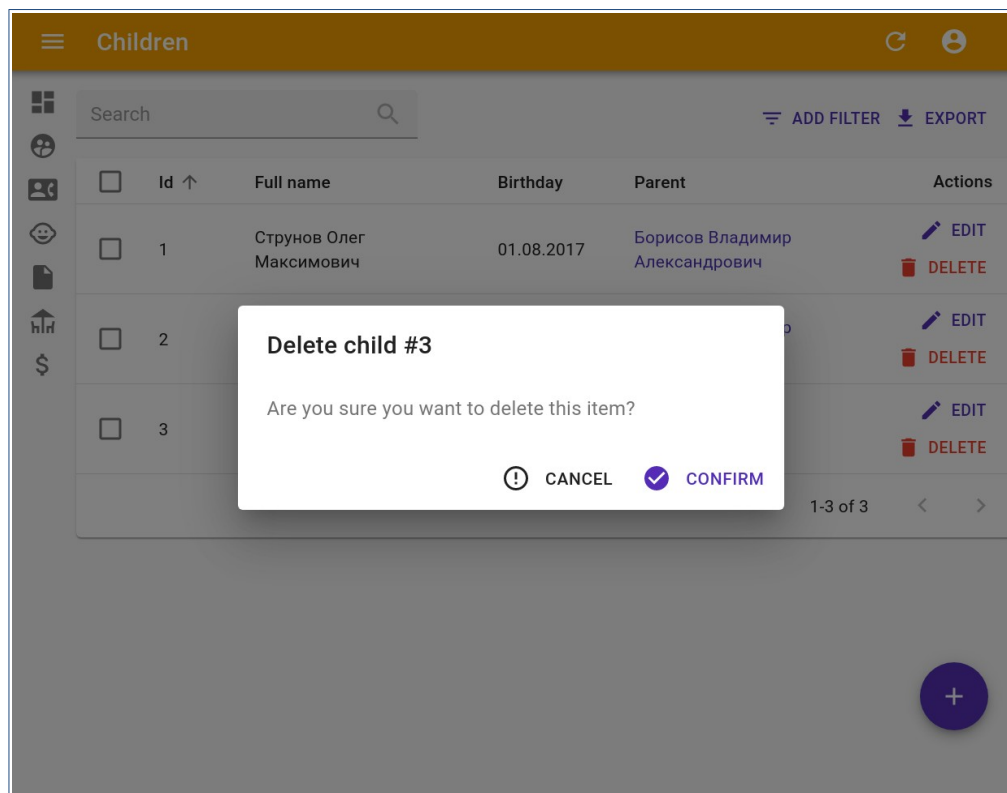


Рисунок 1.12 – OUTPUT-поток FR 7

1.5.1.2 Опис інтерфейсу з зовнішніми пристроями

Таблиця 1.13 – Опис інтерфейсу з зовнішніми пристроями

Ідентифікатор функції	Зовнішній пристрій
FR 1.1 – FR 1.4	Смартфон, планшет, Desktop-персональний комп'ютер, Notebook;
FR 2.1	
FR 3.1 – FR 3.4	
FR 4.1	
FR 5.1 – 5.5	
FR 6.1 – 6.3	

1.5.1.3 Опис програмних інтерфейсів

Версії операційних систем та програмних бібліотек, які знадобляться при реалізації більшості функцій ПП.

- Linux
- Windows
- Android
- IOS
- Підтримка браузерів з html 5 та JS
- Heroku server
- JAVA

Таблиця 1.14 – Опис атрибутів продуктивності

Ідентифікатор функції	Максимальний час реакції, с
FR 1	3
FR 1.1	0.2
FR 1.2	0.5
FR 1.3	0.5
FR 1.4	0.2
FR 2	1.5
FR 2.1	0.3
FR 2.2	0.7
FR 2.3	0.4
FR 2.4	0.3
FR 3	1
FR 3.1	0.4
FR 4	1.2
FR 4.1	0.8
FR 4.2	0.2
FR 5	2
FR 5.1	0.9
FR 5.2	0.5
FR 5.3	0.4
FR 6	1

FR 6.1	0.3
FR 6.2	0.5
FR 6.3	0.6
FR 6.4	0.6
FR 7	1.1
FR 7.1	0.9
FR 7.2	0.7
FR 7.3	0.1

2 Планування процесу розробки програмного продукту

2.1 Планування ітерацій розробки програмного продукту

Таблиця 2.1 – Планування ітерацій розробки

Ідентифікатор функції	функціональна залежність	Вплив на досягнення мети, %	Приоритет
FR 1	-	75	М
FR 1.1	-	75	М
FR 1.2	-	75	М
FR 1.3	-	75	М
FR 1.4	-	75	М
FR 2	FR 1	50	М

FR 2.1	FR 1	50	M
FR 2.2	FR 1	50	M
FR 2.3	FR 1	50	M
FR 2.4	FR 1	50	M
FR 3	FR 2	60	M
FR 4	FR 2	55	S
FR 4.1	FR 2	55	S
FR 4.2	FR 2	55	S
FR 5	FR 4	30	C
FR 5.1	FR 4	30	C
FR 5.2	FR 4	30	C
FR 5.3	FR 4	30	C
FR 6	FR 3	80	S
FR 6.1	FR 3	80	S
FR 6.2	FR 3	80	S
FR 6.3	FR 3	80	S
FR 6.4	FR 3	80	S
FR 7	FR 3	80	M
FR 7.1	FR 3	80	M
FR 7.2	FR 3	80	M
FR 7.3	FR 3	80	M

2.2 Концептуальний опис архітектури програмного продукту

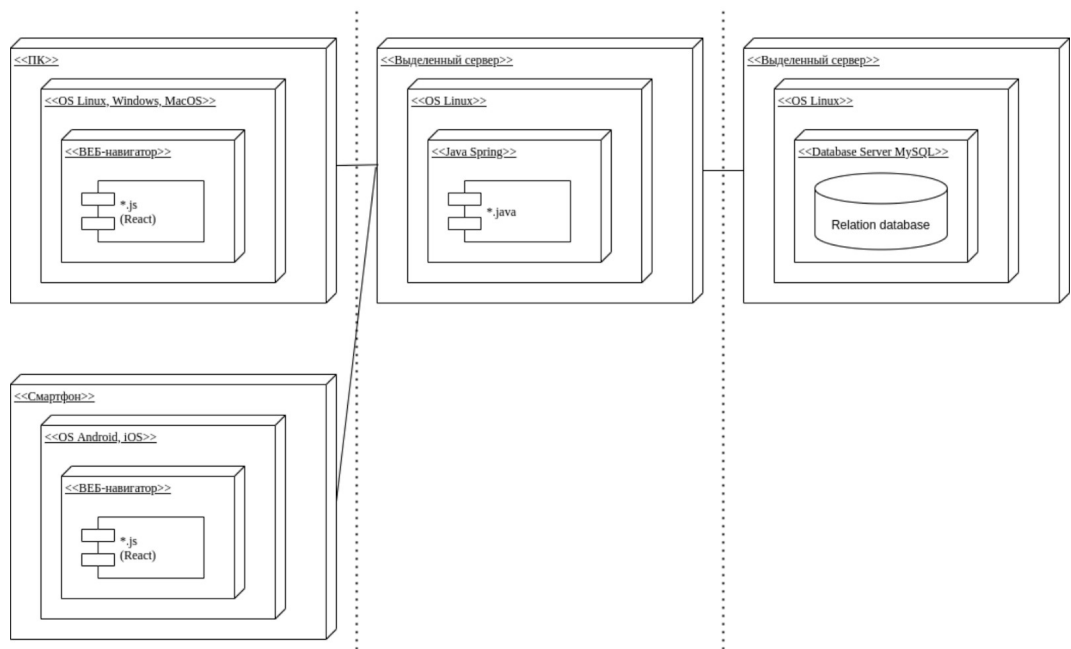


Рисунок 2.1 – Концептуальний опис архітектури ПП

2.3 План розробки ПП

2.3.1 Оцінка трудомісткості розробки ПП

1. Визначення вагових показників акторів (Таблиця 2.1)

Таблиця 2.2 – Вагові коефіцієнти акторів

Актор	Весовой коэффициент
Руководитель	3
Менеджер	2

$A = 3 + 2 = 5$

2. Визначення вагових показників прецедентів UC

Таблица 2.3 – Вагові показники прецедентів

Прецедент	Количество шагов	Весовой коэффициент
Авторизация	4-7	10
Создание сущностей	4-7	10
Список сущностей	≤ 3	5
Детальная информация сущности	≤ 3	5
Фильтр списка	≤ 3	5
Редактирование сущности	4-7	10
Удаление сущности	≤ 3	5

$$UC = 4 * 5 + 3 * 10 = 50$$

3. Визначення UUCP

$$UUCP = A + UC = 5 + 50 = 60$$

4. Визначення технічної складності проекту

Таблица 2.4 – Визначення технічної складності проекту

Показатель	Описание	Вес	ST
T1	Распределённая система	2	3

T2	Высокая производительность	1	2
T3	Работа пользователей онлайн	1	5
T4	Сложная обработка данных	-1	2
T5	Повторное использование кода	1	4
T6	Простота установка	0.5	3
T7	Простота использования	0.5	3
T8	Переносимость	2	2
T9	Простота внесения изменений	1	4
T10	Параллелизм	1	1
T11	Специальные требования к безопасности	1	1
T12	Непосредственный доступ к системе со стороны внешних пользователей	1	2
T13	Специальные требования к обучению пользователей	1	1

$$TCF = 0.6 + (0,01 * (2 * 3 + 1 * 2 + 1 * 5 + (-1) * 2 + 1 * 4 + 0.5 * 3 + 0.5 * 3 + 2 * 2 + 1 * 4 + 1 * 1 + 1 * 1 + 1 * 2 + 1 * 1) = 0.91$$

5. Визначення рівня кваліфікації розробників

Таблица 2.5 – Визначення рівня кваліфікації розробників

Показатель	Описание	Вес	ST
F1	Знание технологий	1.5	3
F2	Опыт разработки	0.5	3
F3	Опыт использования ООП	1	5
F4	Наличие аналитика	0.5	3
F5	Мотивация	1	5
F6	Стабильность требований	2	5
F7	Частичная занятость	-1	5
F8	Сложные языки разработки	-1	3

$$EF = 1.4 + (-0.03 * (1.5 * 3 + 0.5 * 3 + 1 * 5 + 0.5 * 3 + 1 * 5 + 2 * 5 - 1 * 5 - 1 * 3)) = 0.815$$

6. Визначення UCP

$$UCP = UUCP * TCF * EF = 60 * 0.91 * 0.815 = 44.499$$

7. Оцінка трудомісткості проекту

$$\text{Трудоемкость} = UCP * 20 = 889.98 \text{ чел/час}$$

2.3.2 Визначення дерева робіт з розробки ПП

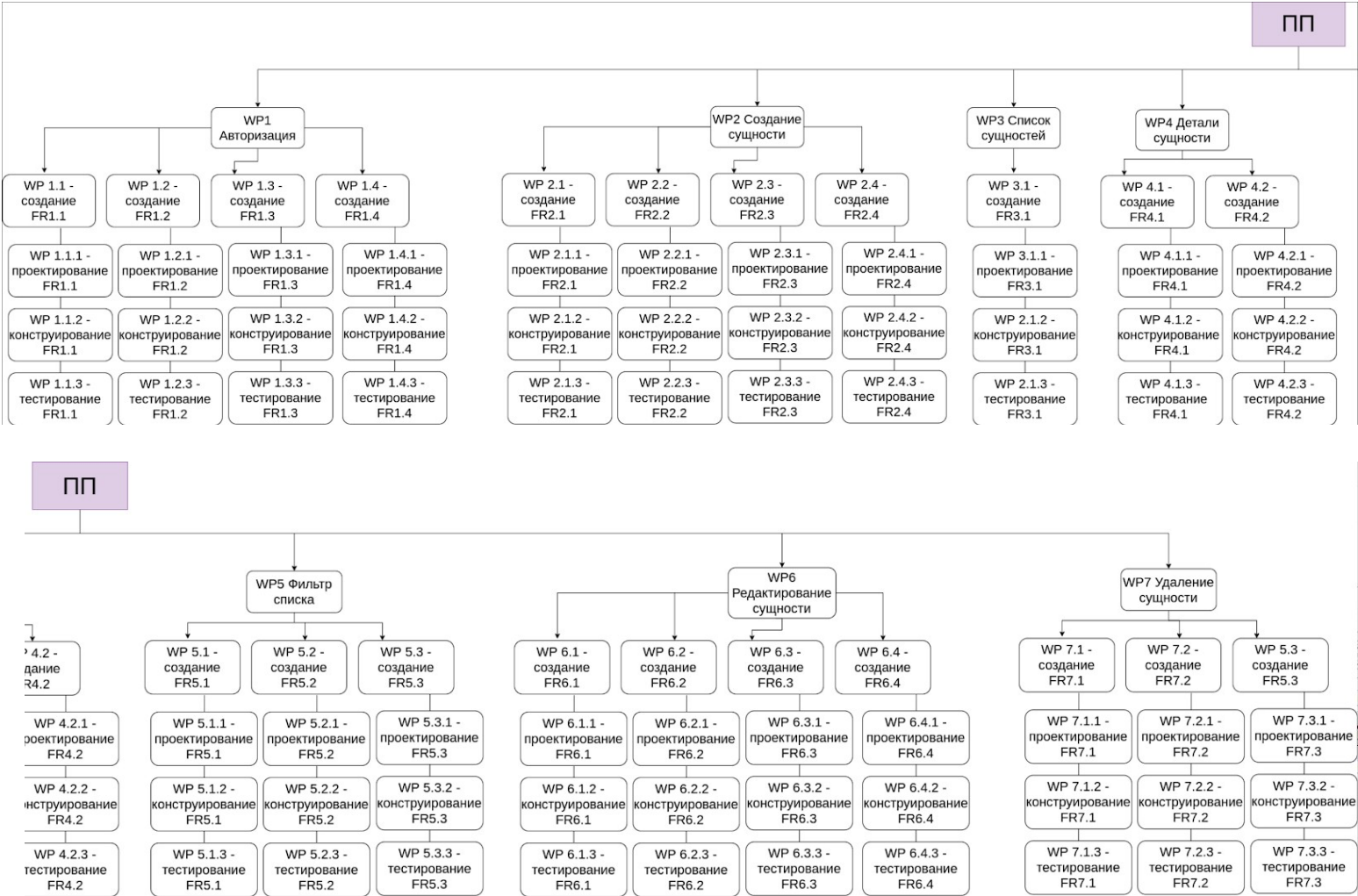


Рисунок 2.2 – WBS дерево робіт

Таблица 2.6 – Опис підзадач з виконавцями

Подзадача	Исполнитель
WST1.*.*	Борисов В.О.
WST2.*.*	Попов С.О.
WST4.*.*	Борисов В.О.
WST4.*.*	Попов С.О.

WST5.*.*	Борисов В.О.
WST6.*.*	Попов С.О.
WST7.*.*	Борисов В.О.

2.3.3 Графік робіт з розробки ПП

2.3.3.2 Діаграма Ганта

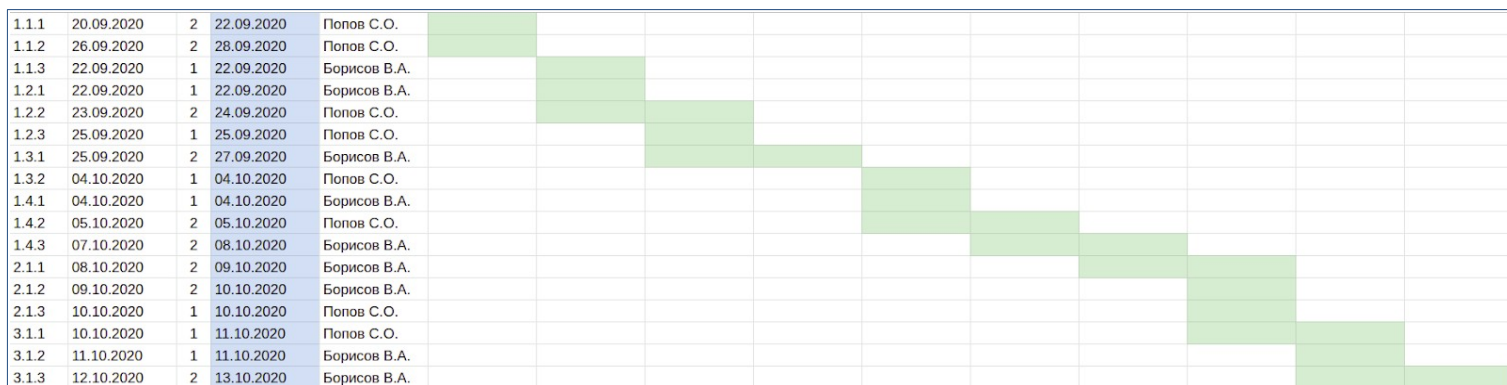


Рисунок 2.4 – Діаграма Ганта

3 Проектування ПП

3.1 Концептуальне та логічне проектування структур даних ПП

3.1.1 Концептуальне проектування на основі UML-діаграми концептуальних класів

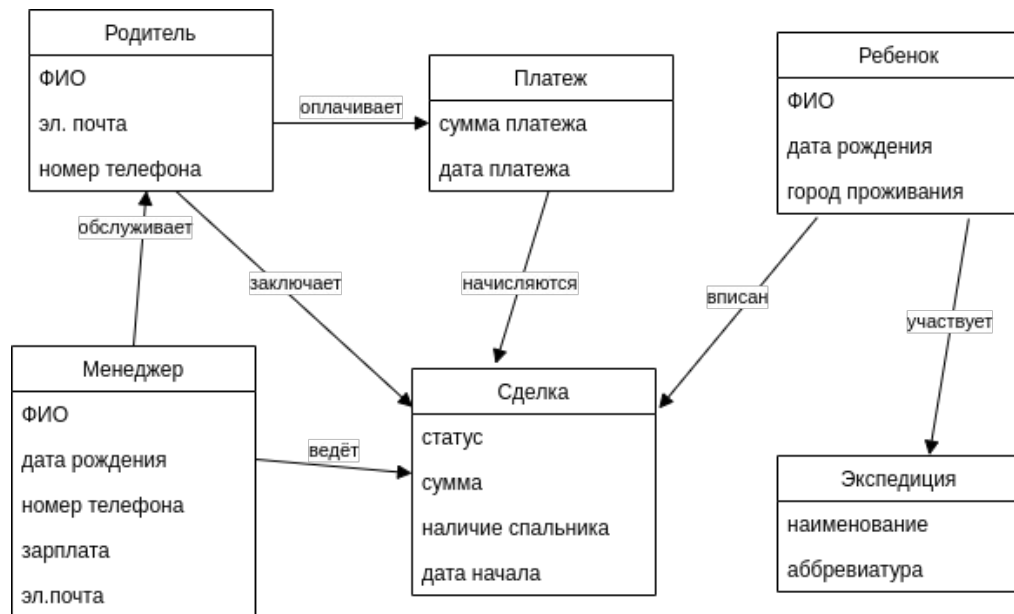


Рисунок 3.1 – UML-діаграма концептуальних класів

3.1.2 Логічне проектування структур даних

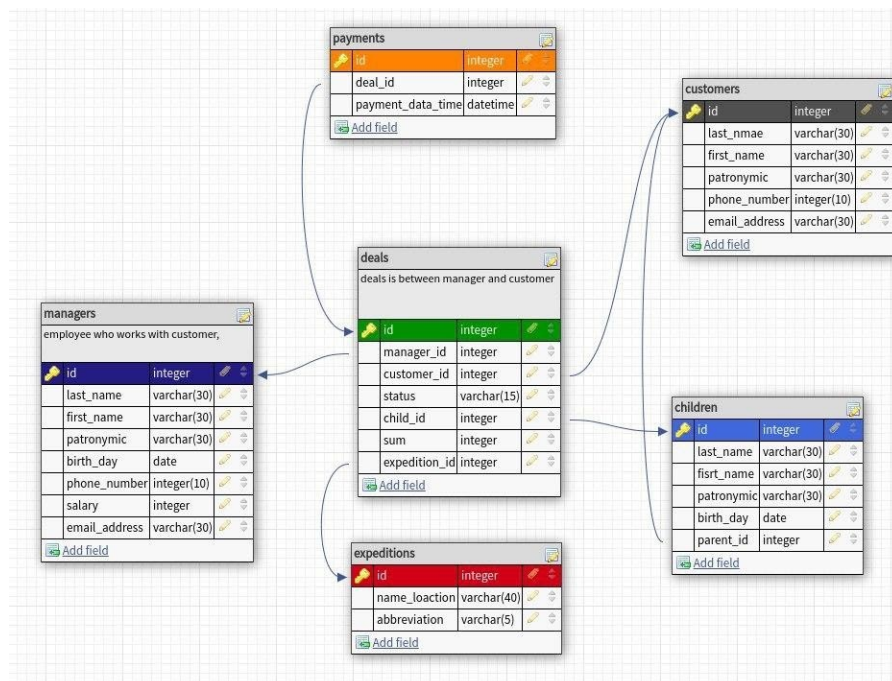


Рисунок 3.2 – Структура БД

3.2 Проектування програмних класів

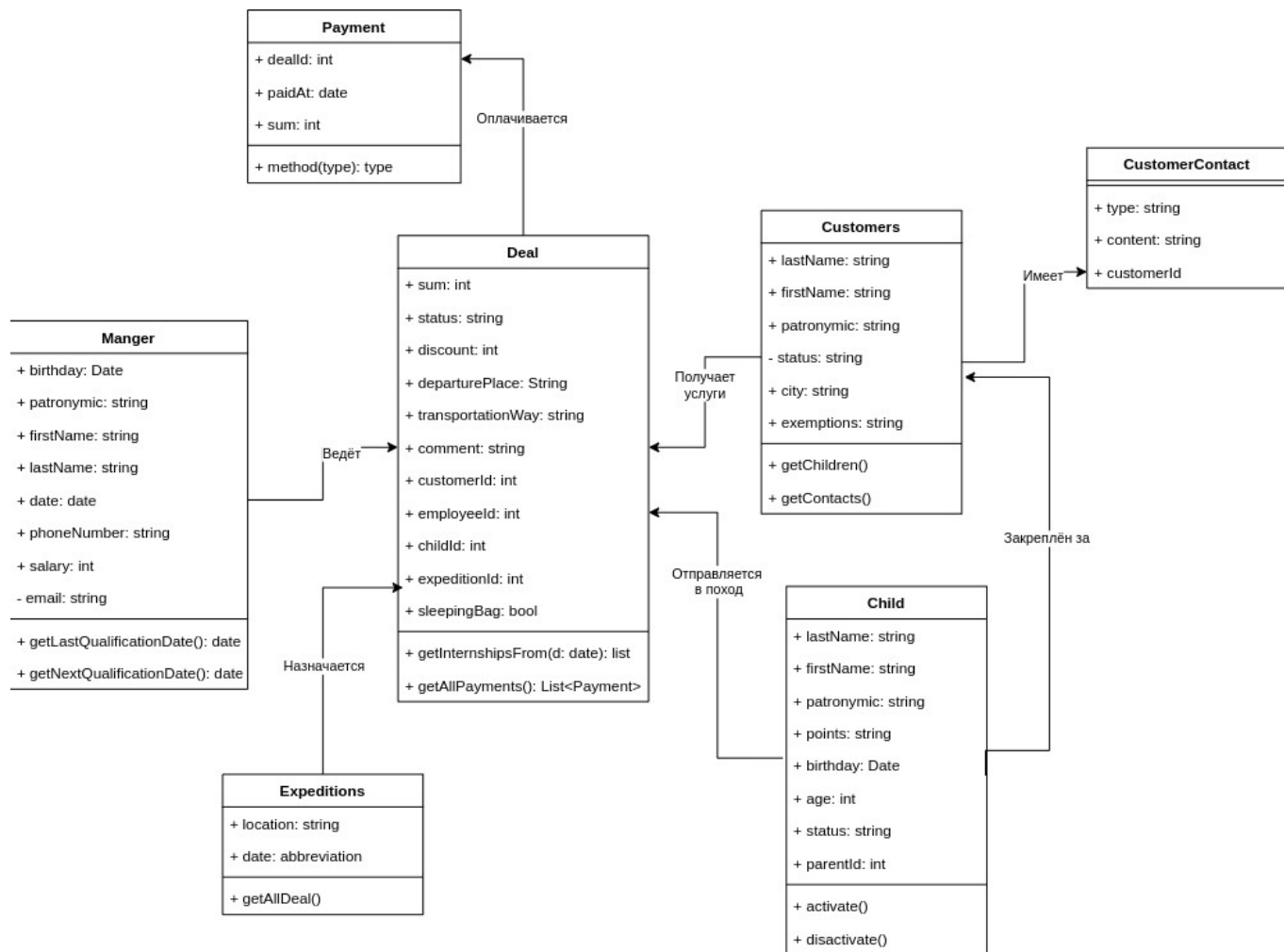


Рисунок 3.3 – Проектування програмних класів

3.3 Проектування алгоритмів роботи методів програмних класів

AddEmployee.puml

@startuml

start

:Выводится форма добавления менеджера;

:Админ вводит информацию о менеджере(ФИО, дату рождения, номер телефона, зарплату, ФИО, email, пароль);

if(Данные корректны) then (yes)

floating note left: Поиск в таблице employees

if(Существует менеджер с таким email?) then (yes)

floating note left: менеджер с таким email уже существует

#pink:Вывод ошибок;

kill

else (no)

:Создается менеджер;

endif

else (no)

#pink:Вывод ошибок;

kill

endif

stop

@enduml

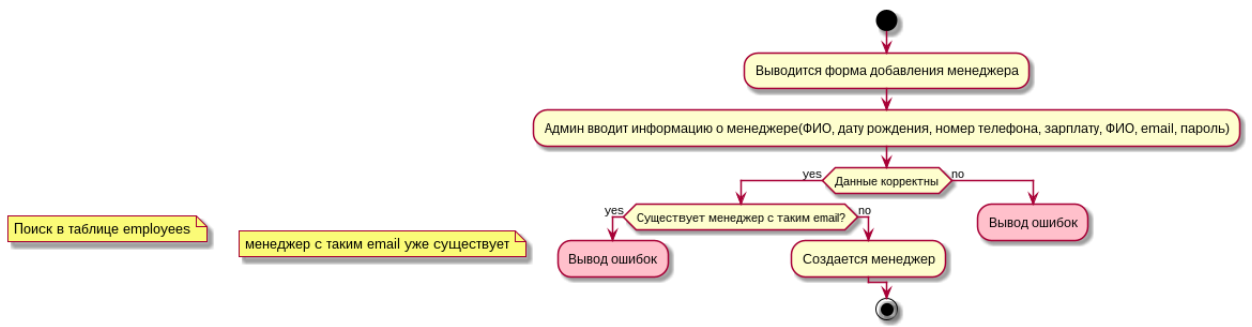


Рисунок 3.4 – Діаграма активностей для «Додати менеджера»

EditProfile.puml

@startuml

start

:Выводится форма редактирования профиля;

:Пользователь вводит новые данные о себе(ФИО,email, пароль, дата рождения, телефон);

if(Данные корректны (Правильный формат даты, email и пароль)) then (yes)

floating note left: SELECT * FROM `employees` WHERE `employee_id` =
__employee_id__

if(Существует пользователь с таким email?) then (yes)

#pink:Вывод ошибок;

kill

else

:Сохраняем информацию;

endif

else

#pink:Вывод ошибок;

kill

endif

stop

@enduml

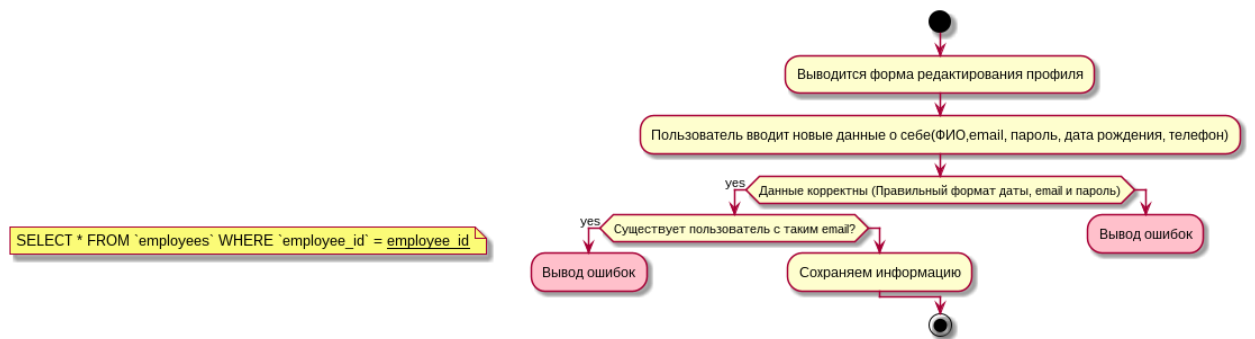


Рисунок 3.5 – Диаграмма активностей для «Редагувати менеджера»

Login.puml

@startuml

start

:Выводится форма логина;

:Пользователь вводит информацию(email и пароль);

if(Валидация данных(корректный email, пароль)) then (yes)

floating note left: `SELECT * FROM `employees` \nWHERE `email` = __email__ AND
`password` = __password__`

if(Поиск пользователя в базе) then (exists)

:Авторизировать пользователя;

else (not exists)

floating note right: Пользователь не существует

#pink:Вывод ошибки;

kill

endif

else (no)

floating note right: Некорректные данные

#pink:Вывод ошибки;

kill

endif

stop

@enduml

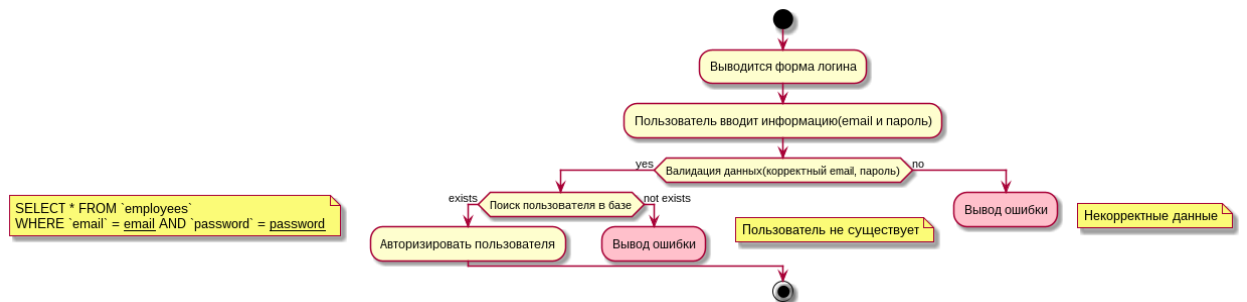


Рисунок 3.6 – Діаграма активностей для «Авторизація»

ViewProfile.puml

@startuml

start

floating note left: SELECT * FROM `users` WHERE `token` = __token__;

if(Пользователь авторизован?) then (yes)

floating note left: SELECT * FROM `employee` WHERE `employee_id` = __employee_id__;
SELECT `deals`.`*` FROM `deals` INNER JOIN `deals` ON id = employee_id WHERE employee_id = __employee_id__;
SELECT * FROM `customers` WHERE `employee_id` = __employee_id__;

:Показать информацию о пользователе\n(ФИО, номер телефона, закреплённые за ним сделки, клиенты);

else (no)

#pink:Показать 403 ошибку;

kill

endif

stop

@enduml

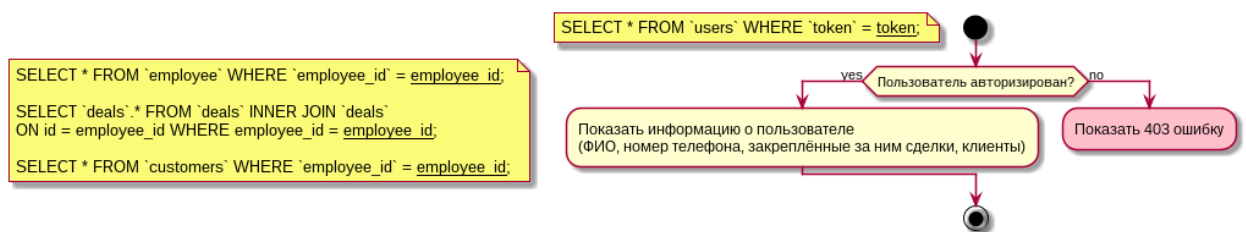


Рисунок 3.7 – Диаграмма активностей для «Перегляд профілю»

3.4 Проектування тестових наборів методів програмних класів

Таблиця 3.1 – Тестові набори

Назва функції	№ тесту	Вхідні значення	Опис очікуваних результатів
Авторизация	1	Ввод валидных данных существующих в БД. Пример: Email: admin Пароль: admin	Токен пользователя
	2	Ввод неправильного email Пример: bademail, aaa, man	Сообщения об ошибке
	3	Ввести слишком короткий пароль Пример: 1 132444	Сообщение об ошибке «Пароль должен быть длиной от 8 до 32 символов»
	4	Ввод валидных данных для несуществующего пользователя в БД	Сообщение об ошибке «Неправильный email или пароль»
Редактировать профиль	1	Ввод валидных данных авторизованным пользователем	Сообщение «Пользователь изменен» и информация о новом пользователе
	2	Не заполнение данных обязательных полей (ФИО, email)	Сообщение об ошибке «Поле обязательно для заполнения»
	3	Не валидный email Пример: Bademail, test@gmail, test@332434	Сообщение об ошибке «Email имеет неправильный формат»
	4	Ввод неправильного телефона	Сообщение об ошибке «Неправильный формат»

		Пример: А34ку2, 323, +380023	телефона»
	5	Попробовать изменить данные неавторизованным пользователем	403 ошибка
Просмотреть список записей	1	Кликнув на элемент списка перейти на профиль с правами	Информация о пользователе
	2	Генерация фильтров	Отфильтрованный список элементов
Создание сделки с клиентом	1	Ввод всех валидных данных	Успешное создание сделки
	2	Добавление в сделку несуществующего клиента	Сообщение об ошибке
	3	Добавление в сделку несуществующего ребёнка	Сообщение об ошибке

4 Конструювання ПП

4.1 Особливості конструювання структур даних

4.1.1 Особливості інсталяції та роботи з СУБД

Використовуючи результати логічного проектування структур даних, опишіть особливості:

Інсталяції та роботи з СУБД (тип СУБД, версія)

В данном ПП использована реляционная СУБД MySQL версии 12. Существуют реализации для множества UNIX-подобных платформ, включая AIX, различные BSD-системы, Linux, macOS, Solaris/ OpenSolaris , Microsoft Windows, а также в контейнеризаторах как Docker.

4.1.2 Особливості створення структур даних

Создание таблиц:

```
CREATE TABLE `managers`  
  
(  
    `id` INT NOT NULL AUTO_INCREMENT,  
    `last_name` varchar(40) NOT NULL,  
    `first_name` varchar(40) NOT NULL,  
    `patronymic` varchar(40),  
    `birth_day` DATE NOT NULL,  
    `phone_number` varchar(14) NOT NULL,  
    `salary` INT NOT NULL,  
    `email_address` varchar(70) NOT NULL,  
    PRIMARY KEY (`id`)  
);  
  
CREATE TABLE `customers`  
  
(  
    `id` INT NOT NULL AUTO_INCREMENT,  
    `last_name` varchar(40) NOT NULL,  
    `first_name` varchar(40) NOT NULL,  
    `patronymic` varchar(40),  
    `email_address` varchar(70),  
    PRIMARY KEY (`id`)
```

```
);
```

```
CREATE TABLE `children`  
  
(  
  
  `id` INT NOT NULL AUTO_INCREMENT,  
  `last_name` varchar(40) NOT NULL,  
  `first_name` varchar(40) NOT NULL,  
  `patronymic` varchar(40),  
  `birth_day` DATE NOT NULL,  
  `parent_id` INT NOT NULL,  
  
  PRIMARY KEY (`id`)  
  
);
```

```
CREATE TABLE `deals`  
  
(  
  
  `id` INT NOT NULL AUTO_INCREMENT,  
  `manager_id` INT NOT NULL,  
  `customer_id` INT NOT NULL,  
  `status` varchar(255) NOT NULL,  
  `child_id` INT NOT NULL,  
  `sum` INT NOT NULL,  
  `sleeping_bag` BOOLEAN,  
  `comment` varchar(255),  
  `expedition_id` INT NOT NULL,  
  
  PRIMARY KEY (`id`)  
  
);
```

```
CREATE TABLE `payments`  
  
(  
  
  `id` INT NOT NULL AUTO_INCREMENT,  
  
  `deal_id` INT NOT NULL,  
  
  `paid_at` TIMESTAMP NOT NULL,  
  
  PRIMARY KEY (`id`)  
  
);
```

```
CREATE TABLE `expeditions`  
  
(  
  
  `id` INT NOT NULL AUTO_INCREMENT,  
  
  `name_loaction` varchar(100) NOT NULL,  
  `abbreviation` varchar(7) NOT NULL  
  UNIQUE,  
  
  PRIMARY KEY (`id`)  
  
);
```

```
CREATE TABLE `customer_phone_numbers`  
  
(  
  
  `id` INT NOT NULL AUTO_INCREMENT,  
  
  `customers_id` INT NOT NULL,  
  
  `phone_number` varchar(14) NOT NULL,  
  
  PRIMARY KEY (`id`)  
  
);
```

4.2 Особливості конструювання програмних модулів

4.2.1 Особливості роботи з інтегрованим середовищем розробки

Данный ПП был реализован в IDE [IntelliJ IDEA](#) — коммерческая кросс-платформенная [интегрированная среда разработки](#) для [Java](#). Разрабатывается компанией [JetBrains](#).

IntelliJ IDEA представляет собой интеллектуальный редактор специальный для [Java](#), однако его можно использовать для множества других языков [HTML](#) и [JavaScript](#), PHP, Haskell и т.д. с возможностями анализа кода на лету, предотвращения ошибок в коде и автоматизированными средствами [рефакторинга](#) для Java и JavaScript. [Автодополнение](#) кода в IntelliJ IDEA поддерживает спецификацию основные JAVA 8, 11, 12, 14 ранних. Имеется полноценный [SQL](#)-редактор с возможностью редактирования полученных результатов запросов.

4.2.2 Особливості створення програмної структури з урахуванням спеціалізованого фреймворку

Разработка данного ПП велась на React(фронт) и Java Spring(бэк)

Spring Framework, или просто **Spring** — один из самых популярных фреймворков для создания веб-приложений на Java. Фреймворками предоставляет ряд инструментариев, где вы просто пишете какие-то свои

классы, прописываете там часть логики, а создает объекты ваших классов и вызывает методы за вас уже сам фреймворк. Чаще всего, ваши классы имплементируют какие-то интерфейсы из фреймворка или наследуют какие-то классы из него, таким образом получая часть уже написанной за вас функциональности.

React (иногда React.js или ReactJS) — JavaScript- библиотека с открытым исходным кодом для разработки пользовательских интерфейсов.

React разрабатывается и поддерживается Facebook, Instagram и сообществом отдельных разработчиков и корпораций.

React может использоваться для разработки одностраничных и мобильных приложений. Его цель — предоставить высокую скорость, простоту и масштабируемость. В качестве библиотеки для разработки пользовательских интерфейсов React часто используется с другими библиотеками, такими как MobX, Redux и GraphQL.

4.2.3 Особливості створення програмних класів

Модель Child.java

@Getter

@Setter

@NoArgsConstructor

@AllArgsConstructor

@Entity


```
@Table(name = "children")
@Builder
public class Child {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private Integer id;

    @Column(name = "last_name", length = 40)
    private String lastName;

    @Column(name = "first_name", length = 40)
    private String firstName;

    @Column(name = "patronymic", length = 40)
    private String patronymic;

    @Column(name = "birth_day")
    @JsonFormat(pattern = GlobalConfig.DATE_FORMAT_PATTERN)
    private LocalDate birthday;

    @Column(name = "points")
    private String points;

    @ManyToOne(fetch = FetchType.LAZY,
        cascade = CascadeType.REFRESH)
    @JsonIgnore
    private Customer parent;
```

```

@Column(name = "city")
private String city;

@Column(name = "status")
@Enumerated(EnumType.STRING)
private ChildStatus status;

@OneToMany(
    mappedBy = "child",
    cascade = {CascadeType.REFRESH},
    fetch = FetchType.LAZY
)
@JsonIgnore
private List<Deal> deals;
}

```

Модель Customer.java

```

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Entity
@Table(name = "customers")
@Builder
public class Customer {

    @Id

```

```
@GeneratedValue(strategy = GenerationType.IDENTITY)
@Column(name = "id")
private Integer id;

@Column(name = "last_name", length = 40)
private String lastName;

@Column(name = "first_name", length = 40)
private String firstName;

@Column(name = "patronymic", length = 40)
private String patronymic;

@Column(name = "city")
private String city;

@Column(name = "status")
@Enumerated(EnumType.STRING)
private CustomerStatus status;

@Column(name = "exemptions")
private String exemptions;

@OneToMany(
    cascade = CascadeType.REFRESH,
    fetch = FetchType.LAZY
)
@JoinColumn(name = "parent_id")
private List<Child> children;
```

```

    @OneToMany(
        mappedBy = "customer",
        cascade = {CascadeType.REMOVE, CascadeType.REFRESH},
        fetch = FetchType.LAZY,
        orphanRemoval = true
    )
    private List<Deal> deals;

    @OneToMany(
        mappedBy = "customer",
        cascade = {CascadeType.REMOVE, CascadeType.REFRESH},
        fetch = FetchType.LAZY,
        orphanRemoval = true
    )
    private List<CustomerContact> contacts;
}

```

Модель CustomerContact.java

```

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Entity
@Table(name = "customer_contacts")
@Builder
public class CustomerContact {

```

```

@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
@Column(name = "id")
private Integer id;

@Column(name = "type")
@Enumerated(EnumType.STRING)
private ContactType type;

@Column(name = "content", length = 150)
private String content;

@ManyToOne(fetch = FetchType.LAZY,
           cascade = CascadeType.REFRESH)
@JoinColumn(name = "customer_id")
private Customer customer;
}

```

Модель Deal.java

```

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Entity
@Table(name = "deals")
@Builder
public class Deal {

```

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "id")

private Integer id;

@Column(name = "sum")

private Integer sum;

@Column(name = "status")

@Enumerated(EnumType.STRING)

private DealStatus status;

@Column(name = "discount")

private String discount;

@Column(name = "departure_place")

private String departurePlace;

@Column(name = "transportation_way")

private String transportationWay;

@Column(name = "comment")

private String comment;

@ManyToOne(fetch = FetchType.LAZY,

 cascade = CascadeType.REFRESH)

@JoinColumn(name = "employee_id")

private Employee employee;

```

    @ManyToOne(fetch = FetchType.LAZY,
        cascade = CascadeType.REFRESH)
    @JoinColumn(name = "customer_id")
    private Customer customer;

    @ManyToOne(fetch = FetchType.LAZY,
        cascade = CascadeType.REFRESH)
    @JoinColumn(name = "child_id")
    private Child child;

    @ManyToOne(fetch = FetchType.LAZY,
        cascade = CascadeType.REFRESH)
    @JoinColumn(name = "expedition_id")
    private Expedition expedition;

    @OneToMany(
        mappedBy = "deal",
        cascade = {CascadeType.REMOVE, CascadeType.REFRESH,
        CascadeType.PERSIST},
        fetch = FetchType.LAZY,
        orphanRemoval = true
    )
    private List<Payment> payments;

    @Column(name = "sleeping_bag")
    private Boolean sleepingBag;
}

```

Модель Employee.java

@Getter

@Setter

@NoArgsConstructor

@AllArgsConstructor

@Entity

@Table(name = "employees")

@Builder

public class Employee {

 @Id

 @GeneratedValue(strategy = GenerationType.IDENTITY)

 @Column(name = "id")

 private Integer id;

 @Column(name = "last_name", length = 40)

 private String lastName;

 @Column(name = "first_name", length = 40)

 private String firstName;

 @Column(name = "patronymic", length = 40)

 private String patronymic;

 @Column(name = "birth_day")

 @JsonFormat(pattern = GlobalConfig.DATE_FORMAT_PATTERN)

 private LocalDate birthday;

 @Column(name = "phone_number", length = 14)

 private String phone;


```

@Column(name = "salary")
private int salary;

@Column(name = "email_address", length = 70)
private String email;

@OneToMany(
    mappedBy = "employee",
    cascade = {CascadeType.REFRESH, CascadeType.PERSIST},
    fetch = FetchType.LAZY
)
private List<Deal> deals;
}

```

Модель Expedition.java

```

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Entity
@Table(name = "expeditions")
@Builder
public class Expedition {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")

```

```

private Integer id;

@Column(name = "name_location", length = 100)
private String location;

@Column(name = "abbreviation", length = 7)
private String abbreviation;

@OneToMany(
    mappedBy = "expedition",
    cascade = {CascadeType.REMOVE, CascadeType.REFRESH},
    fetch = FetchType.LAZY,
    orphanRemoval = true
)
private List<Deal> deals;
}

```

Модель Payment.java

```

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Entity
@Table(name = "payments")
@Builder
@EntityListeners(AuditingEntityListener.class)
public class Payment {

```

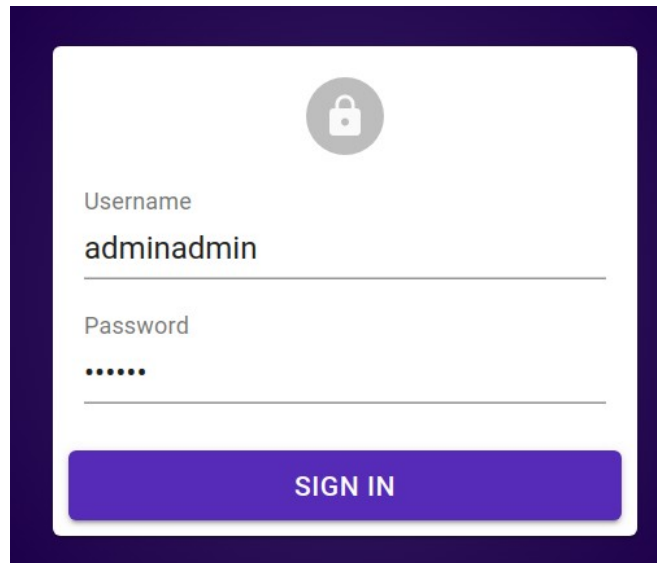
```
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
@Column(name = "id")
private Integer id;

@Column(name = "paid_at")
@JsonFormat(pattern = GlobalConfig.DATE_FORMAT_PATTERN_WITH_TIME)
@CreatedDate
private LocalDateTime paidAt;

@ManyToOne(fetch = FetchType.LAZY,
    cascade = {CascadeType.REFRESH, CascadeType.PERSIST})
@JoinColumn(name = "deal_id")
private Deal deal;

@Column(name = "sum")
private Integer sum;
}
```

4.3 Тестування програмних модулів



A login form with a purple border. At the top center is a grey circular icon with a white padlock. Below it, the text 'Username' is followed by a text input field containing 'adminadmin'. Below that, the text 'Password' is followed by a text input field containing seven dots. At the bottom is a large purple button with the text 'SIGN IN' in white.

Рисунок 4.1 – Невалідні дані

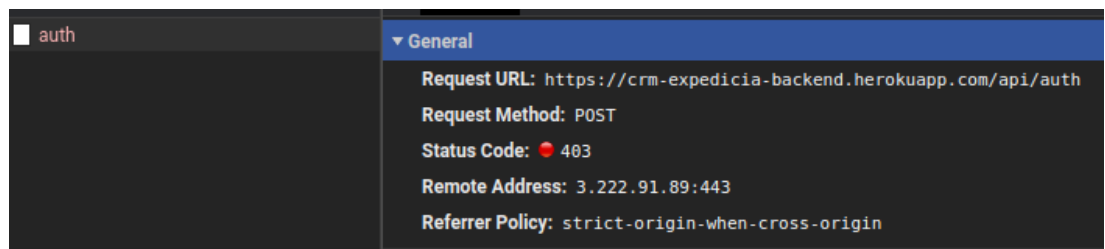


Рисунок 4.2 – Невалідні дані

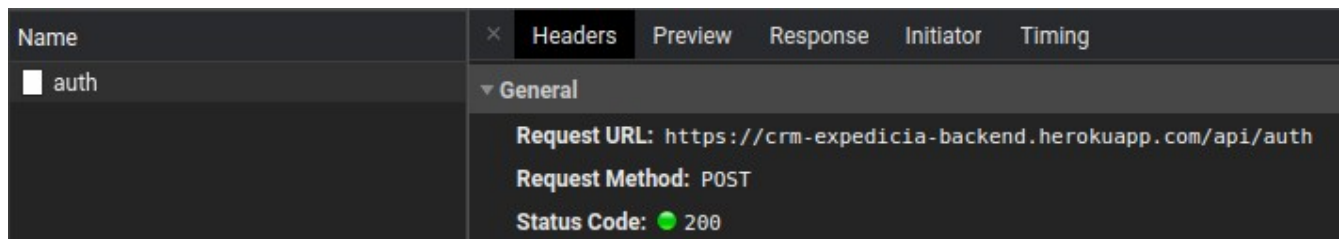


Рисунок 4.3 – Успішна авторизація

Name	× Headers Preview Response Initiator Timing
auth	<pre> { "jwt": "eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJhZG1pb1IsImV4cCI6MTYwNzE5MTE0MywibWF0IjoxNjA3MTYyMzQzL18ge9w7zQVvTmkhaaXmIVkpw70FJNrdmzhN9DYDP0uf4sV9NMLp-42ud0Hv1IsXX_IHMYXJacCCDIay8sE-CvA" } </pre>

Рисунок 4.4 – У відповідь отримуємо JWT-токен

Create Employee

Last name

Пупкин

First name

Василий

Patronymic

Васильевич

Birthday

1998-12-15

Phone

+380505050505

Salary

2000

Email

pupkin@mail.io

SAVE

Рисунок 4.5 – Створення суті і перегляд списку сутностей

Employees

Search

ADD FILTER

CREATE

EXPORT

<input type="checkbox"/>	Id ↑	Full name	Birthday	Phone	Salary	Email	Actions
<input type="checkbox"/>	1	Семёнов Алексей Александрович	02.02.2020	0978887755	0	ho@ho.com	EDIT DELETE
<input type="checkbox"/>	3	Борисов Александр Владимирович	09.04.2020	0985925572	1000	vovbor@gmail.com	EDIT DELETE
<input type="checkbox"/>	5	Пупкин Василий Васильевич	15.12.1998	+380505050505	2000	pupkin@mail.io	EDIT DELETE

Rows per page: 10 1-3 of 3

☰

Employee #3

Last name
Иванов

First name
Иван

Patronymic
Иванович

Birthday
2020-04-09

Phone
0985925572

Salary
1000

Email
vovbor@gmail.com

SAVE

Рисунок 4.5 – Редагування

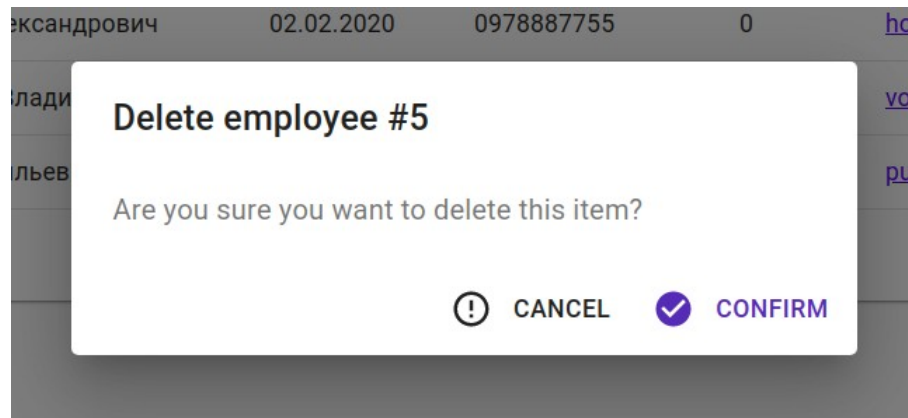


Рисунок 4.6 – Видалення

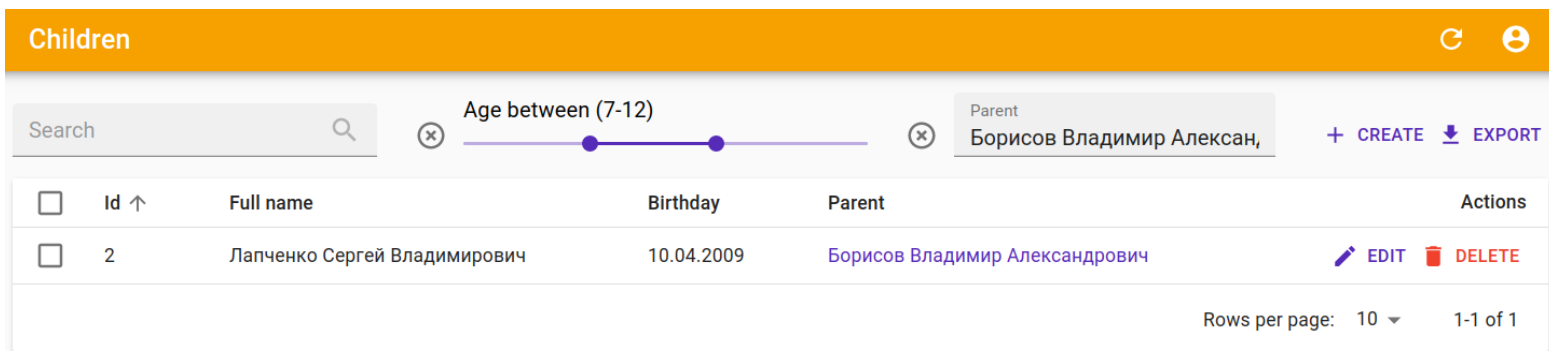


Рисунок 4.6 – Фільтрація списку

5 Розгортання та валідація ПП

5.1 Інструкція з встановлення ПП

Для встановлення ПП треба клонувати репозиторії з гитхабу командою
git clone <link>

бек: <https://github.com/vovBorya/crm-expedicia-backend.git>

фронт: <https://github.com/vovBorya/crm-expedicia-frontend.git>

Кожний репозиторій має файл README.md з детальною інструкцією для інсталяції

5.3 Результати валідації ПП

Метою створення ПП було зменшення відсотка помилок під час роботи відділу кадрів.

Відсоток помилок EP (EP – Error percent) можна визначити як

$$EP = E / N,$$

де

E – кількість помилок під час роботи;

N – загальна кількість роботи

До введення ПП цей показник складав ~15%. Після введення ПП цей показник становить ~3%. На основі цієї можна сказати, що ПП пройшов валідацію й виконує свою мету.

Висновки

В результаті створення програмного продукту була досягнута наступна мета його споживача: зменшення відсотку помилок під час роботи відділу кадрів.

Доказом цього є значення метрики ЕР, яка зменшилась приблизно в 5 разів.

В процесі створення програмного продукту виникли такі труднощі:

- 1) обмеженість в часі;
- 2) складність розробки ПП;
- 3) недостатні знання де-яких інструментів.

Не зважаючи на це всі прецеденти, які були заплановані було реалізовано.