НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КПІ»

Кафедра

Обчислювальної техніки

КУРСОВА РОБОТА

|  |
| --- |
| з «ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ»  на тему: «Система пошуку та замовлення їжі FoodService» |

|  |  |  |
| --- | --- | --- |
|  | Студента 2 курсу групи ІО-31  напряму підготовки  6.050102 «Комп’ютерна інженерія»  Рожков Григорій Олегович  Керівник  Болдак Андрій Олександрович  (прізвище та ініціали)  Доцент кафедри ОТ  (посада, вчене звання, науковий ступінь)  Національна шкала\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Кількість балів: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Оцінка: ECTS \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | |
|  |  |  |
|  |  |

|  |
| --- |
| м. Київ - 2015 рік |

[РОЗДІЛ 1 3](#_Toc415530713)

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

2

*6.050102 «Комп’ютерна інженерія»*

Розроб.

Рожков Г О

Перевір.

Болдак А О

Реценз.

Н. Контр.

Затверд.

Болдак А О

Система пошуку та замовлення їжі

Літ.

Акрушів

50

*Кафедра Обчислювальної техніки*

[ЗАПИТИ ЗАЦІКАВЛЕНИХ ОСІБ 3](#_Toc415530714)

[1.1. Введення 3](#_Toc415530715)

[1.2. Короткий огляд продукту 3](#_Toc415530716)

1.3. Ділові правила..........................................................................................3

[1.4. Функціональність системи 7](#_Toc415530717)

[1.5. Надійність. 8](#_Toc415530718)

РОЗДІЛ 2 9

[РОЗРОБКА ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ 9](#_Toc415530719)

[2.1. Загальна схема прецедентів 9](#_Toc415530720)

[2.2. Прецеденти для ролі головного адміністратора 10](#_Toc415530721)

[2.3. Діаграма бізнес-сутностей 14](#_Toc415530722)

[2.4. Реляційна модель бази даних 15](#_Toc415530723)

[РОЗДІЛ 3 14](#_Toc415530725)

[РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ 15](#_Toc415530725)

[3.1. Реляційно-об’єктне відображення 16](#_Toc415530726)

[3.2. Специфікація Service класів 17](#_Toc415530727)

[3.3. Специфікація DAO-класів 18](#_Toc415530728)

[3.4. Класи контролерів та їх специфікація 19](#_Toc415530729)

[РОЗДІЛ 4 20](#_Toc415530730)

[ІЛЮСТРАЦІЯ РОБОТИ ПРОГРАМИ 20](#_Toc415530730)

[4.1. Реєстрація пошуковця в системі. 20](#_Toc415530731)

[СПИСОК ІНФОРМАЦІЙНИХ ДЖЕРЕЛ 22](#_Toc415530732)

[ДОДАТОК А 23](#_Toc415530733)

[ДОДАТОК Б 32](#_Toc415530734)



Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

3

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

ЗАПИТИ ЗАЦІКАВЛЕНИХ ОСІБ

* 1. Введення

У цьому документі описуються запити зацікавлених осіб по відношенню до розроблюваної системи «Food service»: замовник – будь-який заклад громадського харчування.

* + 1. Мета

Метою документа є визначення основних вимог до функціональності, продуктивності, зручності, доступності, а також визначення бізнес-правил і технологічних обмежень, пред’явлених предмету розробки.

* + 1. Контекст

Перелік вимог, перерахованих у цьому документі, є основою технічного завдання на розробку системи пошуку та замовлення їжі в «Food service».

* 1. Короткий огляд продукту

Система пошуків ресторанів та кафе «Food service» - база даних всіх закладів харчування. Для менеджерів закладів харчування працює адміністратор, який перевіряє і додає їх до бази. Пошуковець може знайти собі заклад харчування до смаку і зробити в ньому замовлення, а також залишити відгук про ті, в яких він вже побував. До того ж, він є користувачем соціальної мережі, в якій він зможе збирати компанії для спільних обідів, надсилати повідомлення, а також заводити нові знайомства і додавати в друзі.

* 1. **Ділові правила**
     1. Призначення системи «Food service»

Система є, перш за все, зручним інструментом для осіб, шукаючих певне місце, де б вони могли провести гарно час за смачною їжою, або замовити вподобане меню з собою. Для пошуковців вона виконує функції навігатора закладів харчування. Їм також надається можливість користуватись соціальною мережею, яка функціонує як комунікативний простір для пошуковців. Соціальна мережа дозволяє їм додавати друзів, відсилати повідомлення та створювати групи однодумців (люди, які мають спільні інтереси щодо конкретних страв або закладів харчування, або ж просто особи, які знайшли будь-які інші спільні інтереси).

З іншого боку система є рекламним простором для закладів харчування. Навколо неї зав’язуються відносини між пошуковцями та адміністраторів закладів харчування, які представляють свої бізнес-точки. Для обробки замовлень потрібен додатковий вид працівників, які звуться менеджерами. Для них «FoodService» є робочим місцем.

* + 1. Політика взаємодії з клієнтом

В загальному, клієнти системи поділяються на три типи: пошуковці – особи, які зайшли на даний сервіс в цілях довідки про певний заклад, або проводять будь-яку активну діяльність в «Food service», яка в подальшому якимось чином вплине на прибуток, зареєстрованих в базі даних закладів харчування; адміністратори закладів харчування (АЗХ), які є керуючими представниками своїх установ.

Для кожного зареєстрованого закладу відводиться спеціальна сторінка з актуальними, на будь-який момент, даними про заклад, тобто меню, гарячі пропозиції, коментарі, оцінки закладу, час роботи і т.д. Відповідальність за оновлення інформації беруть на себе АЗХ. Система збирає статистику про кількість відвідувачів, замовлень та про інші корисні дані та відсилає її АЗХ. Перед тим як щось замовляти пошуковець прагне знайти для себе підходящий заклад, він заходить на веб-сайт «Food service» і тут же задає критерії для пошуку: місце, назва, рейтинг і т.д. Вибравши місце для замовлень, пошуковець може скидати бажані товари в корзину, а потім оформити замовлення. Також він може забронювати столик (при наявності), або просто залишити коментар про заклад. В соціальній мережі авторизованим користувачам надається можливість спілкуватись, запрошувати один одного на обіди і ще багато інших функцій.

* + 1. Характеристика ділового процесу

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

4

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

З точки зору ділового процесу, АЗХ надається гарна можливість для “піару” своїх закладів харчування, використовуючи дану систему. Вигодою для АЗХ є те, що він постійно бачить статистику, яка свідчить про популярність його закладів. Це дає можливість йому розуміти користь системи. Але за це МЗХ має періодично сплачувати плату. Пошуковці ж безкоштовно користуються системою, що сприяє збільшенню обсягу клієнтів «Food service». Також АЗХ має ділові стосунки з менеджерами, які обробляють замовлення пошуковців. За таку роботу АЗХ виплачують їм заробітню плату.

* + 1. Сценарій реєстрації нового закладу

**Назва:** Реєстрація закладу харчування в системі.

**Учасники:** АЗХ, адміністратор, система.

**Передумови:** АЗХ хоче розмістити заклад в системі.

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

5

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

**Результат:** АЗХ подає заявку адміністратору на розміщення в системі.

**Основний сценарій:**

1. АЗХ подає запит на розміщення закладу.
2. Система надає форму для розміщення закладу.
3. АЗХ вводить параметри закладу (назва, фото закладу, короткий опис, адреса, наявність доставки).
4. АЗХ натискає кнопку «Зареєструвати».

**Виключні ситуації:**

1. МЗХ не заповнив якесь з полів.
   * 1. Сценарій реєстрації АЗХ

**Назва:** Реєстрація нового Адміністратора закладів харчування в системі.

**Учасники:** АЗХ, система.

**Передумови:** немає.

**Результат:** система реєструє нового АЗХ.

**Основний сценарій:**

1. АЗХ наискає на кнопку Sign Up -> Shop Admin.
2. Система надає форму для реєстрації.
3. АЗХ заповнює всі поля.
4. АЗХ натискає кнопку «Register».
5. Система реєструє нового АЗХ.

**Виключні ситуації:**

1. АЗХ не заповнив якесь з полів або поле не пройшло етап валідації.
   * 1. Сценарій реєстрації АЗХ

**Назва:** Реєстрація нового Пошуковця.

**Учасники:** Пошуковець, система.

**Передумови:** немає.

**Результат:** система реєструє нового Пошуковця.

**Основний сценарій:**

1. Пошуковець наискає на кнопку Sign Up ->Simple User.
2. Система надає форму для реєстрації.
3. Пошуковець заповнює всі поля.
4. Пошуковець натискає кнопку «Register».
5. Система реєструє нового Пошуковеця.

**Виключні ситуації:**

1. Пошуковець не заповнив якесь з полів або поле не пройшло етап валідації.

* + 1. Сценарій реєстрації Менеждера

**Назва:** Реєстрація нового Менеджера.

**Учасники:** Менеджер, система.

**Передумови:** немає.

**Результат:** система реєструє нового Менеджера.

**Основний сценарій:**

1. Менеджер наискає на кнопку Sign Up ->Manager.
2. Система надає форму для реєстрації.
3. Менеджер заповнює всі поля, включаючи емейл АЗХ.
4. Пошуковець натискає кнопку «Register».
5. Система відправляє заявку менеджера на підтвердження реєстрації Менеджера до АЗХ.

**Виключні ситуації:**

1. Пошуковець не заповнив якесь з полів або поле не пройшло етап валідації.

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

6

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

* + 1. Сценарій відправлення запиту на додання в друзі

**Назва**: Створення запиту на додання в друзі.

**Учасники**: Пошуковець, користувач.

**Попередні умови**: Пошуковець не має у друзях іншого пошуковця і не являється його підписчиком.

**Результат**: видалення користувача.

**Основний сценарій:**

1. Пошуковець знаходить профіль іншого пошуковця;

2. Пошуковець заходить на сторінку профіля іншого пошуковця;

3. Пошуковець відправляє заявку, натискаючи на кнопку «Add to friends»

* + 1. Сценарій знаходження закладу харчування

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

7

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

**Назва**: Знаходження закладу харчування.

**Учасники**: Будь-який з трьох типів користувачів.

**Попередні умови**: немає.

**Результат**: Переходження на сторінку попоредньо вибраного закладу харчування.

**Основний сценарій:**

1. Користувач заходить на сторінку сервісу, натискаючи на кнопку «Service».

2. Система перенаправлює користувача на сторінку з фільтром для пошуку закладів.

3. Користувач фільтрує результат за назвою, рейтингом та розташуванням шуканого закладу.

4. Система відображає список відфільтрованих закладів.

5. Користувач натискає на назву певного закладу.

6. Система перенаправлює користувача на сторінку закладу.

* 1. Функціональність системи

Основні вимоги до функціональності, пропоновані зацікавленими особами до предмета розробки, відносяться до трьох категоріях:

• Адмінісратор закладів харчування (АЗХ).

• Менеджер.

• Пошуковець.

* + 1. Можливості АЗХ:

• реєстрація та видалення своїх закладів харчування;

• редагування контенту закладів харчування;

• редагування власного профілю;

• підтвердження реєстрації нових менеджерів, як співробітників;

• вибір стратегії обробки замовлень менеджерами;

• перегляд рейтингу інших закладів харчування.

* + 1. Можливості Менеджера:

• обробка замовлень пошуковців;

• редагування власного профілю;

• підтвердження бронювання на столикспевним пошуковцем;

• призначення групам користувачів прав доступу до даних;

• перегляд рейтингу інших закладів харчування.

* + 1. Можливості Пошуковця

• додання в друзі;

• відправлення повідомлень;

• редагування власного профілю;

• пошук інших пошуковців, АЗХ, менеджерів;

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

8

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

• пошук закладів харчування;

• пошук страв;

• користування корзиною замовлень;

• створення замовлень;

* 1. Надійність.
     1. Резервне копіювання і відновлення даних.

Має здійснюватися резервне копіювання баз даних.

* + 1. Реєстрація дій користувачів в системі.

В цілях безпеки та для забезпечення контролю використання даних і внесень змін, дії всіх користувачів в системі реєструються.

* + 1. Інтерфейс для користувача.

Інтерфейс подання електронного пошуку закладів швидкого харчування повинен відповідати наступним вимогам :

* Бути зрозумілим і не допускати двозначного тлумачення.
* Бути інтуїтивно зрозумілим.

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

9

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

РОЗРОБКА ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ

* 1. Загальна схема прецедентів

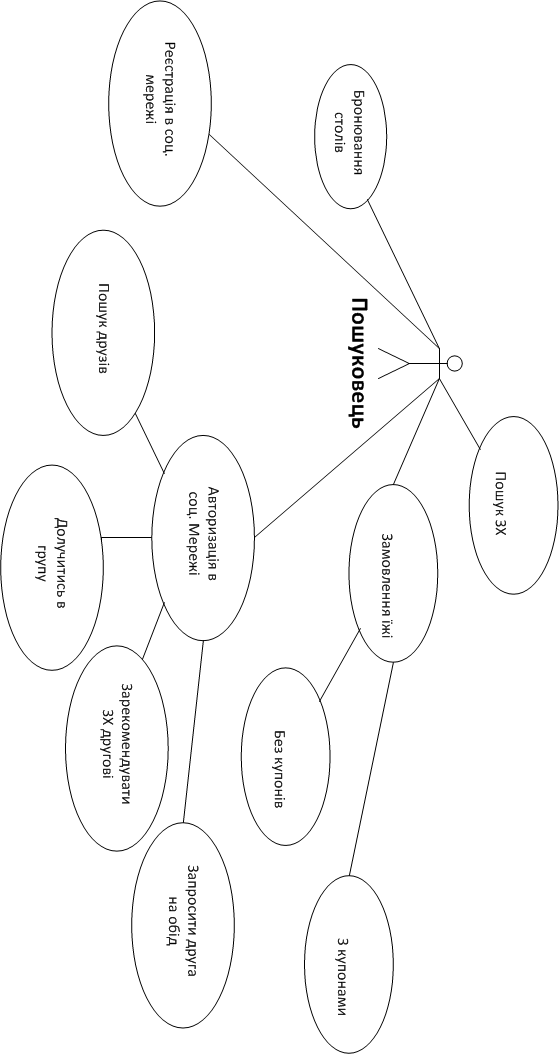
Загальна схема прецедентів для ролі пошуковця показує можливі послідовності дій актора. Основним видом діяльності пошуковця є пошук закладів зарчування (ЗХ), замовлення їжі, бронювання столиків та діяльність в соціальній мережі, де пошуковці мають змогу мати друзів та спілкуватися. Схема прецедентів представлена на рис. 2.1.

Рис. 2.1 – Загальна схема прецедентів для ролі головного адміністратора

* 1. Прецеденти для ролі пошуковця

**Прецедент №1.** Замовлення страви в одному ЗХ.

**Учасники:** Пошуковець, система.

**Передумови:** Пошуковець хоче зробити замовлення.

**Результат:** Пошуковець знайшов відповідний ЗХ для замовлення .

**Основний сценарій:**

1. Пошуковець фільтрує результат за критеріями пошуку(назва ЗХ , оцінка ЗХ, наявність доставки ЗХ¸ назва страви, кухня страви, ціна страви, столики(відкривається додаткова панель, де описані розміри столиків(кількість місць) та є можливість ставити галку біля кожного такого запису).

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

10

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

1. Пошуковець натискає кнопку “Знайти”.
2. Система обробляє критерії.
3. Система представляє список ЗХ, задовольнивших критеріям пошуку(Першими у списку будуть заклади з найвищими оцінками та наявністю купонів(гаряча пропозиція, знижка).
4. Пошуковець обирає конкретний ЗХ.
5. Пошуковець робить замовлення (визначається дата замовлення та дата обслуговування)з купонами/без купонів .
6. Система обробляє замовлення.

**Виключні ситуації:**

1. Пошуковець ввів не правильно критерії оцінювання.
2. Система не знайшла жодного ЗХ.
3. Пошуковець передумав замовляти страву, після оброблення системою замовлення.

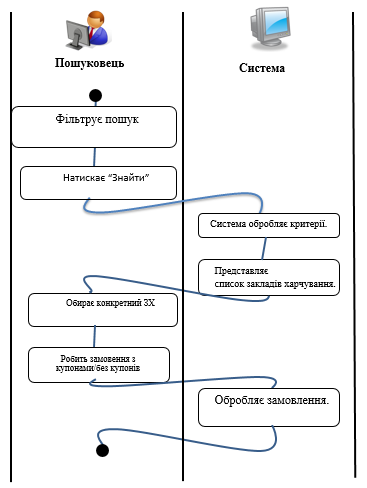


Рис. 2.2 – Схема призначення / видалення регіонального адміністратора

**Прецедент №2.** Реєстрація в системі.

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

11

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

**Учасники:** Пошуковець, система.

**Передумови:** Пошуковець не має доступу до соціальної мережі сервісу.

**Результат:** Пошуковець реєструється у соціальній мережі сервісу.

**Основний сценарій:**

1. Пошуковець відправляє запит на реєстрацію в соціальній мережі.
2. Система надає форму для реєстрації всоціальній мережі.
3. Пошуковець вводить свої дані(емейл, пароль, дата народження, фото, інформація про себе).
4. Пошуковець натискає кнопку “Зареєструватися”.
5. Система дає доступ пошуковцю до соціальної мережі сервісу.

**Виключні ситуації:**

1. Введено невірні персональні дані.

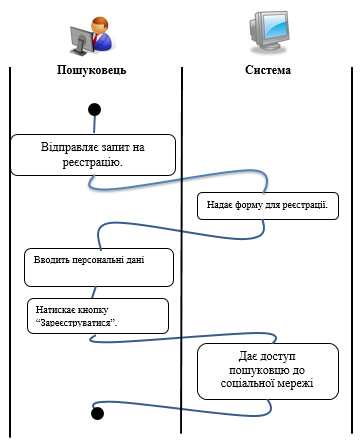


Рис. 2.3 – Схема реєстрації пошуковця в системі

**Прецедент №3.** Бронювання столика.

**Учасники:**, система, Пошуковець.

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

12

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

**Передумови:** немає.

**Результат:** Відправляється заявка на бронювання столика.

**Основний сценарій:**

1. Пошуковець заходить на сторінку закладу харчування.
2. Пошуковець вибирає столик зі списку доступних.
3. Пошуковець натискає «Забронювати».
4. Система реєструє запит на бронювання столика, який, в подальшому, буде підтверджений, або відхилений Менеджером.

**Виключні ситуації:**

1. В наявності не залишилося жодного столика

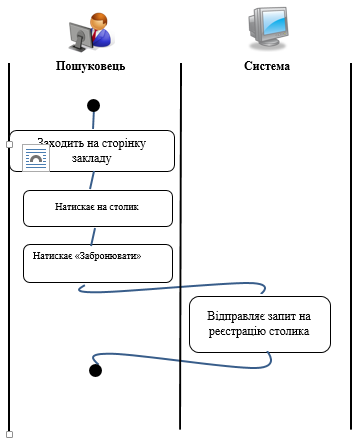


Рис. 2.4 – Схема відправлення запиту на бронювання столика

**Прецедент** **№4**. Додання у друзі

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

13

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

**Учасники:** Пошуковець(екземпляр1), пошуковець (екземпляр2)

**Передумови:** Пошуковець(ек1) хоче додати у друзі пошуковця(ек2)

**Результат:** Пошуковець(ек1) додав пошуковця(ек2) у друзі.

**Основний сценарій:**

1. Пошуковець(ек1) знаходить потенційного друга на сторінці пошуку людей;
2. Пошуковець(ек1) заходить на сторінку потенційного друга;
3. Пошуковець(ек1) натискає «Додати в друзі»
4. Пошуковець(ек2) отримує запит про приєднання до друзів пошуковця(ек1);
5. Пошуковець(ек2) натискає кнопку”Прийняти запит”.
6. Пошуковець(ек1) приєднався у друзі пошуковця(ек2).

**Виключні ситуації:**

1. Пошуковець(ек2) відмовив на запит про приєднання у друзі.



Рис. 2.5 – Схема додання у друзі.

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

13

6.050102 «Комп'ютерна інженерія»

Змн.

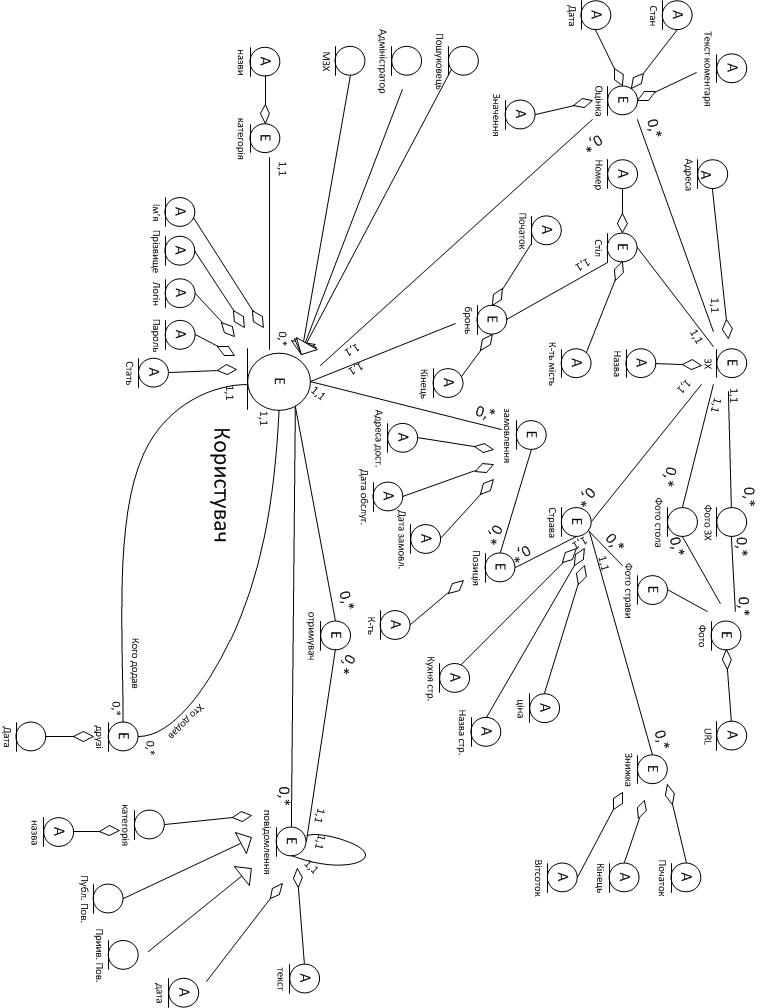
№ докум.

Арк.

Підпис

Дата

* 1. Діаграма бізнес-сутностей

Дана діаграма створюється на етапі бізнес моделювання. Вона відображає основні сутності та взаємозв’язки між ними. Діаграма бізнес-сутностей проекту зображена на рис. 2.8.

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

14

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

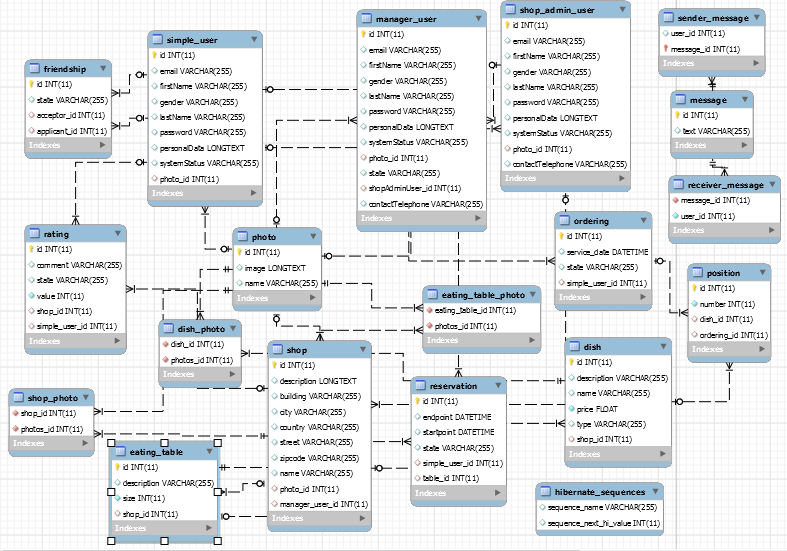
Підпис

Дата

Рис. 2.8 – Діаграма бізнес-сутностей

* 1. Реляційна модель бази даних

Реляційна модель бази даних (рис 2.3) зображує структуру таблиць бази даних, взаємозв’язки між ними та поля кожної з таблиць. Наведена діаграма має багато схожого з діаграмою бізнес-сутностей. Кожній основній бізнес-сутності відповідає таблиця баз даних. Script для створення бази даних наведений у додатку А



Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

15

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

Рис 2.9 – Реляційна модель

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

23

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

* 1. Архітектура проекту

Проект складається с двох модулів: RESTful сервіс та веб-сайт. Модулі можуть працювати на різних комп’ютерах, завдяки такій архітектурі модель відділена від контроллера та призентації. Це дозволяє розвиток даного проекту в майбутньому

* 1. Реляційно-об’єктне відображення

Для реляційно-об’єктного відображення в програмі використовується бібліотека Hibernate. Вона надає можливість легко встановити зв’язок з будь-якою базою даних та створити відображення між об’єктно-орієнтованою моделлю та традиційною реляційною моделлю баз даних. На рис. 3.1 зображено діаграму Entity класів для ролі SimpleUser. Детальна специфікація наведена в додатку Б.

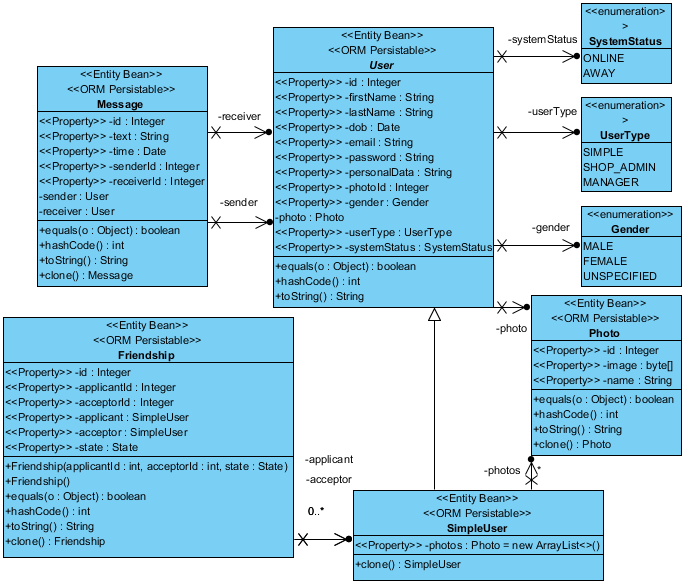


Рис 3.1 – Діаграма entity класів

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

24

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

16

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

36

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

* 1. Специфікація RESTful сервіс класів

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

17

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

RESTful – Representational State Transfer є сучасним підходом для створення масштабованих, гнучких веб-програм. Дані повинні передаватися у вигляді невеликої кількості стандартних форматів ([HTML](http://uk.wikipedia.org/wiki/HTML), [XML](http://uk.wikipedia.org/wiki/XML), [JSON](http://uk.wikipedia.org/wiki/JSON)). Мережевий протокол (як і HTTP) повинен підтримувати кешування, не повинен залежати від мережевого прошарку, не повинен зберігати інформацію про стан між парами «запит-відповідь». Стверджується, що такий підхід забезпечує масштабування системи і дозволяє їй еволюціонувати з новими вимогами. Говорячи у контексті паттерна MVC – Model View Controller можна стверджувати, що RESTful сервіс є моделлю (у деяких випадках і частиною контролера). Така архітектура дозволяє створювати додаткові клієнтські програми, які ставатимуть частиною великої системи, не змінюючи іншу частину системи. У даному проекті дані передаються у форматі JSON по синхронному протоколу HTTP. Діаграму класів RESTful ресурсів для сутностей, що мають пряме відношення до ролі SimpleUser наведено на рис. 3.2. Детальна специфікація наведена в додатку Б.



Рис. 3.2 – Діаграма класів RESTful ресурсів

* 1. Специфікація DAO-класів

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

18

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

Класи, що тут представлені, містять методи для роботи з базою даних. Ці методи використовують бібліотеку Hibernate. Діаграму цих класів можна побачити на рис. 3.3.

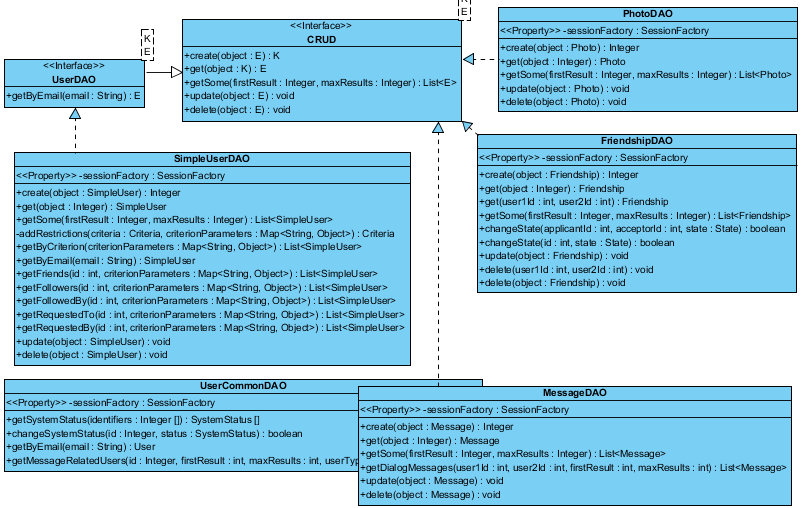


Рис 3.3 – Діаграма DAO класів

* 1. Класи контролерів та їх специфікація

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

19

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

Дані класи призначені для створення зв’язку між сервером та клієнтом та реалізації певної логіки. Вони знаходяться в клієнтському модулі проекту foodservice.com. Діаграму цих класів можна побачити на рис. 3.4.

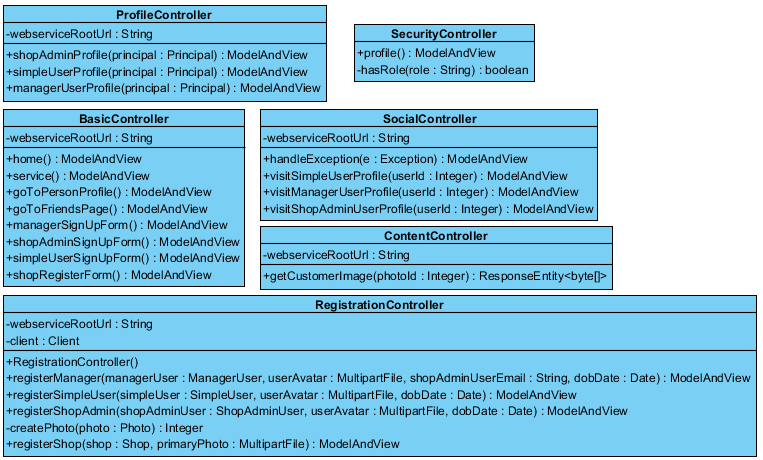


Рис. 3.4 – Діаграма класів контролерів

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

20

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

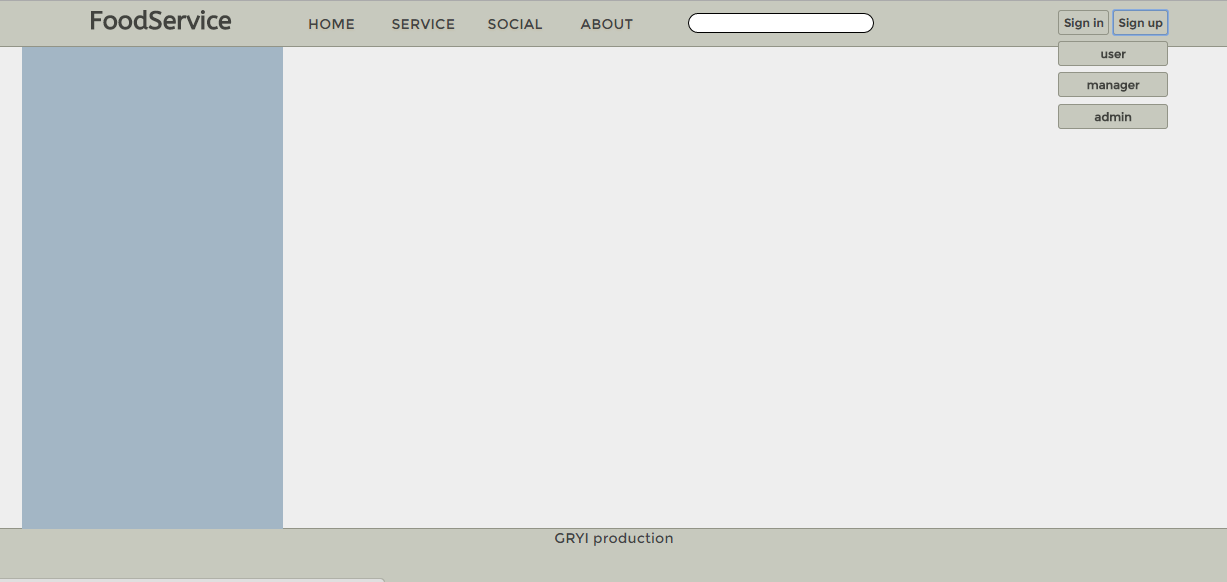
Підпис

Дата

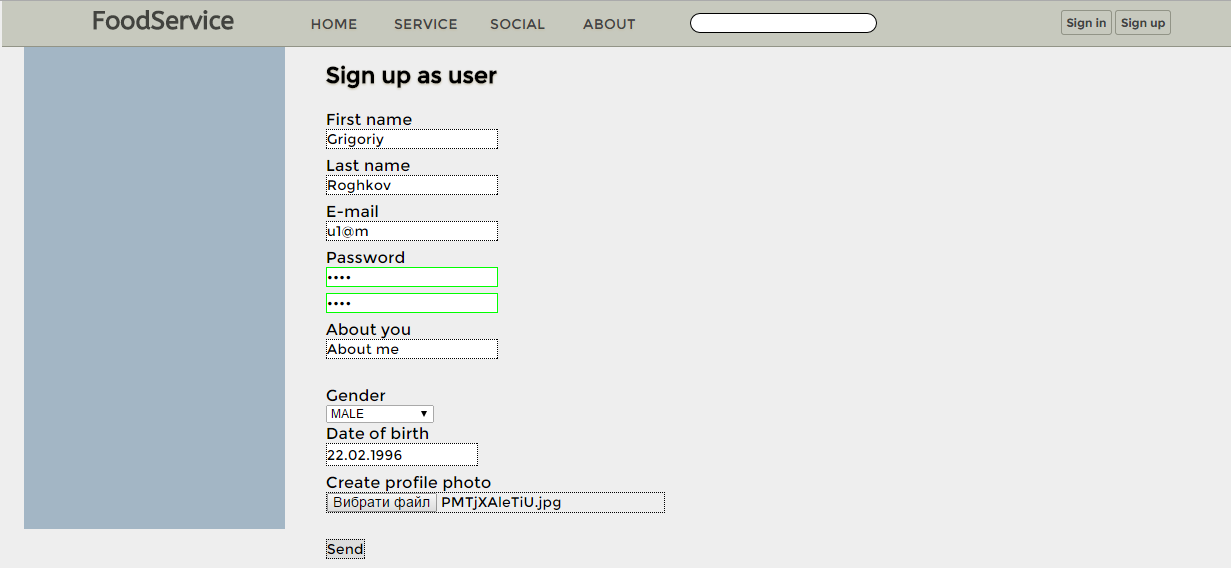
ІЛЮСТРАЦІЯ РОБОТИ ПРОГРАМИ

Для ілюстрації роботи програми в цьому розділі наведено графічні сценарії роботи проекту.

* 1. Реєстрація Пошуковця в системі та перехід на персональний профіль.
     1. Незареєстрований користувач знаходиться на головній сторінці. Далі він натискає на випадаюче меню sign up та вибирає user.



* + 1. Система відображає форму для реєстрації. Пошуковець заповнює форму відповідними даними та натискає send.



Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

20

6.050102 «Комп'ютерна інженерія»

Змн.

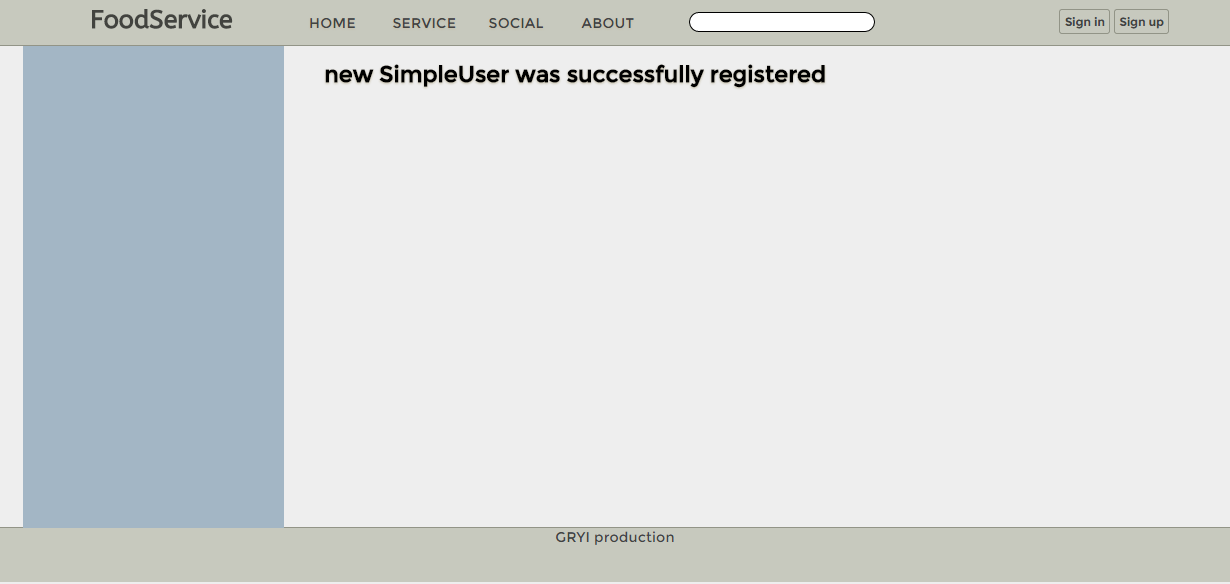
№ докум.

Арк.

Підпис

Дата

* + 1. Система повідомляє пошуковця про успішну реєстрацію.



Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

21

6.050102 «Комп'ютерна інженерія»

Змн.

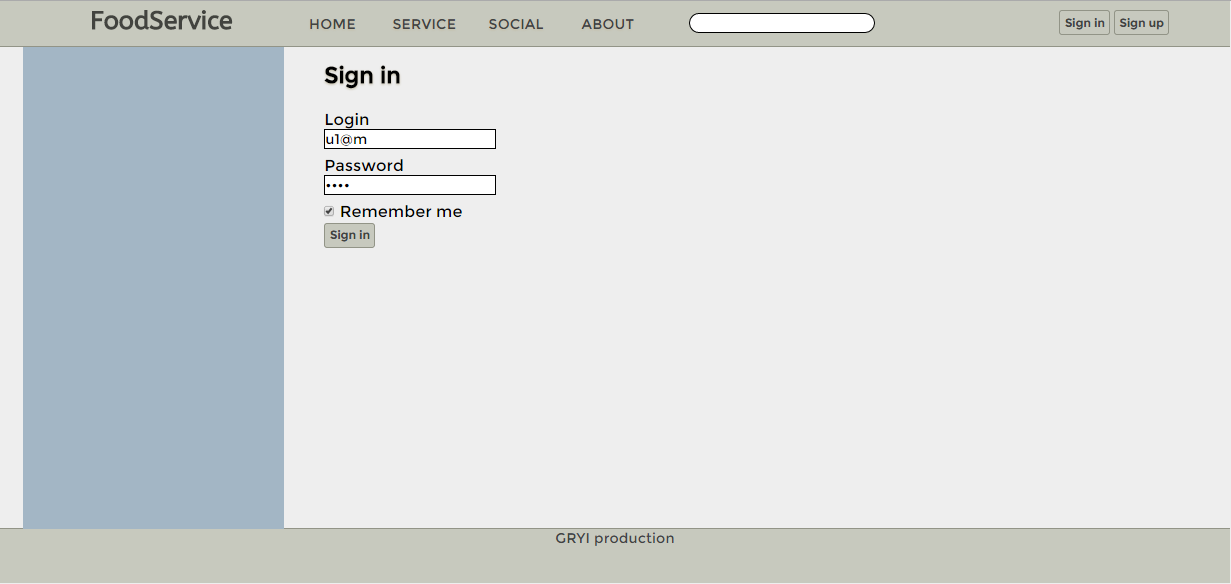
№ докум.

Арк.

Підпис

Дата

* + 1. Далі користувач натискає sign in і система надає форму для авторизації. Пошуковець заповнює поля для логіна і пароля та натискає sign in.



Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

39

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

* + 1. Після успішної аутентифікації та авторизації система перенаправляє користувача на сторінку персонального профілю.

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

21

6.050102 «Комп'ютерна інженерія»

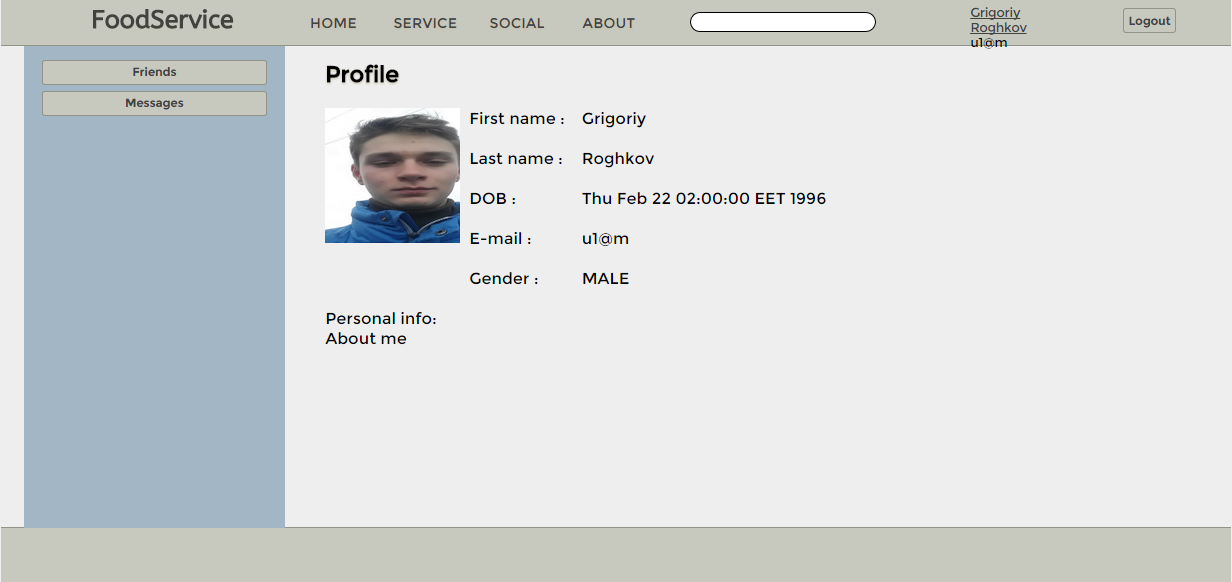
Змн.

№ докум.

Арк.

Підпис

Дата



Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

22

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

СПИСОК ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

23

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

1. Apache Tomcat. – Посилання: <http://tomcat.apache.org>
2. Hibernate. – Посилання: <http://hibernate.org>
3. Spring. – Посилання: http://docs.spring.io/spring/docs/current/spring-framework-reference/htmlsingle/
4. Maven. – Посилання: <https://maven.apache.org>
5. Git. – Посилання: <http://git-scm.com>
6. GitHub. – Посилання: <https://github.com>

ДОДАТОК А

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

24

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

CREATE SCHEMA IF NOT EXISTS `food\_service\_db` DEFAULT CHARACTER SET utf8 ;

USE `food\_service\_db` ;

-- -----------------------------------------------------

-- Table `food\_service\_db`.`dish`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `food\_service\_db`.`dish` (

`id` INT(11) NOT NULL,

`description` VARCHAR(255) NULL DEFAULT NULL,

`name` VARCHAR(255) NULL DEFAULT NULL,

`price` FLOAT NULL DEFAULT NULL,

`primary\_photo\_id` INT(11) NULL DEFAULT NULL,

`shop\_id` INT(11) NULL DEFAULT NULL,

`type` VARCHAR(255) NULL DEFAULT NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------

-- Table `food\_service\_db`.`eating\_table`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `food\_service\_db`.`eating\_table` (

`id` INT(11) NOT NULL,

`description` VARCHAR(255) NULL DEFAULT NULL,

`photo\_id` INT(11) NULL DEFAULT NULL,

`shop\_id` INT(11) NULL DEFAULT NULL,

`size` INT(11) NULL DEFAULT NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

25

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------

-- Table `food\_service\_db`.`simple\_user`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `food\_service\_db`.`simple\_user` (

`id` INT(11) NOT NULL,

`dob` DATE NULL DEFAULT NULL,

`email` VARCHAR(255) NOT NULL,

`firstName` VARCHAR(255) NULL DEFAULT NULL,

`gender` VARCHAR(255) NULL DEFAULT NULL,

`lastName` VARCHAR(255) NULL DEFAULT NULL,

`password` VARCHAR(255) NOT NULL,

`personalData` LONGTEXT NULL DEFAULT NULL,

`photo\_id` INT(11) NULL DEFAULT NULL,

`systemStatus` VARCHAR(255) NULL DEFAULT NULL,

`userType` VARCHAR(255) NULL DEFAULT NULL,

PRIMARY KEY (`id`),

UNIQUE INDEX `UK\_ppsffdtoh9bb8w4c1ey9r78oi` (`email` ASC))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------

-- Table `food\_service\_db`.`friendship`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `food\_service\_db`.`friendship` (

`id` INT(11) NOT NULL AUTO\_INCREMENT,

`acceptor\_id` INT(11) NULL DEFAULT NULL,

`applicant\_id` INT(11) NULL DEFAULT NULL,

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

26

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

`state` VARCHAR(255) NULL DEFAULT NULL,

PRIMARY KEY (`id`),

UNIQUE INDEX `UK\_lkr6i90jb40b65vwymd1tn76c` (`applicant\_id` ASC, `acceptor\_id` ASC),

INDEX `FK\_gcvesuuhbxedruo9eul9n9sv5` (`acceptor\_id` ASC),

CONSTRAINT `FK\_gcvesuuhbxedruo9eul9n9sv5`

FOREIGN KEY (`acceptor\_id`)

REFERENCES `food\_service\_db`.`simple\_user` (`id`),

CONSTRAINT `FK\_ldkg5fvqngb2u29glaff4y30v`

FOREIGN KEY (`applicant\_id`)

REFERENCES `food\_service\_db`.`simple\_user` (`id`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------

-- Table `food\_service\_db`.`hibernate\_sequences`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `food\_service\_db`.`hibernate\_sequences` (

`sequence\_name` VARCHAR(255) NULL DEFAULT NULL,

`sequence\_next\_hi\_value` INT(11) NULL DEFAULT NULL)

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------

-- Table `food\_service\_db`.`manager\_user`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `food\_service\_db`.`manager\_user` (

`id` INT(11) NOT NULL,

`dob` DATE NULL DEFAULT NULL,

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

27

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

`email` VARCHAR(255) NOT NULL,

`firstName` VARCHAR(255) NULL DEFAULT NULL,

`gender` VARCHAR(255) NULL DEFAULT NULL,

`lastName` VARCHAR(255) NULL DEFAULT NULL,

`password` VARCHAR(255) NOT NULL,

`personalData` LONGTEXT NULL DEFAULT NULL,

`photo\_id` INT(11) NULL DEFAULT NULL,

`systemStatus` VARCHAR(255) NULL DEFAULT NULL,

`userType` VARCHAR(255) NULL DEFAULT NULL,

`shop\_admin\_user\_id` INT(11) NULL DEFAULT NULL,

`state` VARCHAR(255) NULL DEFAULT NULL,

PRIMARY KEY (`id`),

UNIQUE INDEX `UK\_inso2c6rbj875rfb2t6qvuryo` (`email` ASC))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------

-- Table `food\_service\_db`.`message`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `food\_service\_db`.`message` (

`id` INT(11) NOT NULL AUTO\_INCREMENT,

`receiver\_id` INT(11) NULL DEFAULT NULL,

`sender\_id` INT(11) NULL DEFAULT NULL,

`text` VARCHAR(255) NULL DEFAULT NULL,

`time` DATETIME NULL DEFAULT NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

28

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

-- -----------------------------------------------------

-- Table `food\_service\_db`.`ordering`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `food\_service\_db`.`ordering` (

`id` INT(11) NOT NULL AUTO\_INCREMENT,

`finish\_date` DATETIME NULL DEFAULT NULL,

`manager\_user\_id` INT(11) NULL DEFAULT NULL,

`order\_date` DATETIME NULL DEFAULT NULL,

`orderingStatus` VARCHAR(255) NULL DEFAULT NULL,

`simple\_user\_id` INT(11) NULL DEFAULT NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------

-- Table `food\_service\_db`.`photo`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `food\_service\_db`.`photo` (

`id` INT(11) NOT NULL AUTO\_INCREMENT,

`image` LONGBLOB NULL DEFAULT NULL,

`name` VARCHAR(255) NULL DEFAULT NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB

AUTO\_INCREMENT = 2

DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------

-- Table `food\_service\_db`.`position`

-- -----------------------------------------------------

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

29

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

CREATE TABLE IF NOT EXISTS `food\_service\_db`.`position` (

`id` INT(11) NOT NULL AUTO\_INCREMENT,

`dish\_id` INT(11) NULL DEFAULT NULL,

`number` INT(11) NULL DEFAULT NULL,

`ordering\_id` INT(11) NULL DEFAULT NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------

-- Table `food\_service\_db`.`rating`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `food\_service\_db`.`rating` (

`id` INT(11) NOT NULL AUTO\_INCREMENT,

`comment` VARCHAR(255) NULL DEFAULT NULL,

`shop\_id` INT(11) NULL DEFAULT NULL,

`simple\_user\_id` INT(11) NULL DEFAULT NULL,

`value` FLOAT NULL DEFAULT NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------

-- Table `food\_service\_db`.`reservation`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `food\_service\_db`.`reservation` (

`id` INT(11) NOT NULL AUTO\_INCREMENT,

`endpoint` DATETIME NULL DEFAULT NULL,

`manager\_user\_id` INT(11) NULL DEFAULT NULL,

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

30

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

`simple\_user\_id` INT(11) NULL DEFAULT NULL,

`startpoint` DATETIME NULL DEFAULT NULL,

`state` VARCHAR(255) NULL DEFAULT NULL,

`table\_id` INT(11) NULL DEFAULT NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------

-- Table `food\_service\_db`.`shop`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `food\_service\_db`.`shop` (

`id` INT(11) NOT NULL AUTO\_INCREMENT,

`description` LONGTEXT NULL DEFAULT NULL,

`building` VARCHAR(255) NULL DEFAULT NULL,

`city` VARCHAR(255) NULL DEFAULT NULL,

`otherInfo` VARCHAR(255) NULL DEFAULT NULL,

`region` VARCHAR(255) NULL DEFAULT NULL,

`street` VARCHAR(255) NULL DEFAULT NULL,

`name` VARCHAR(255) NULL DEFAULT NULL,

`primary\_photo\_id` INT(11) NULL DEFAULT NULL,

`shop\_admin\_user\_id` INT(11) NULL DEFAULT NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------

-- Table `food\_service\_db`.`shop\_admin\_user`

-- -----------------------------------------------------

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

31

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

CREATE TABLE IF NOT EXISTS `food\_service\_db`.`shop\_admin\_user` (

`id` INT(11) NOT NULL,

`dob` DATE NULL DEFAULT NULL,

`email` VARCHAR(255) NOT NULL,

`firstName` VARCHAR(255) NULL DEFAULT NULL,

`gender` VARCHAR(255) NULL DEFAULT NULL,

`lastName` VARCHAR(255) NULL DEFAULT NULL,

`password` VARCHAR(255) NOT NULL,

`personalData` LONGTEXT NULL DEFAULT NULL,

`photo\_id` INT(11) NULL DEFAULT NULL,

`systemStatus` VARCHAR(255) NULL DEFAULT NULL,

`userType` VARCHAR(255) NULL DEFAULT NULL,

`contactData` LONGTEXT NULL DEFAULT NULL,

PRIMARY KEY (`id`),

UNIQUE INDEX `UK\_ikhks7o96i8j8r2ro3jf3w232` (`email` ASC))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------

-- Table `food\_service\_db`.`shop\_photo`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `food\_service\_db`.`shop\_photo` (

`photo\_id` INT(11) NOT NULL,

`shop\_id` INT(11) NOT NULL,

UNIQUE INDEX `UK\_b5vs7hjvtkpov1oo6klwt55g3` (`shop\_id` ASC),

INDEX `FK\_fyf92nl2sj5as15v2bvvte4qj` (`photo\_id` ASC),

CONSTRAINT `FK\_b5vs7hjvtkpov1oo6klwt55g3`

FOREIGN KEY (`shop\_id`)

REFERENCES `food\_service\_db`.`photo` (`id`),

CONSTRAINT `FK\_fyf92nl2sj5as15v2bvvte4qj`

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

32

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

FOREIGN KEY (`photo\_id`)

REFERENCES `food\_service\_db`.`shop` (`id`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------

-- Table `food\_service\_db`.`simple\_user\_photo`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `food\_service\_db`.`simple\_user\_photo` (

`photo\_id` INT(11) NOT NULL,

`shop\_id` INT(11) NOT NULL,

UNIQUE INDEX `UK\_l62vpvg5psyvxl6q4yo5102gd` (`shop\_id` ASC),

INDEX `FK\_2wxuh84xxur2djq175dlkyd0g` (`photo\_id` ASC),

CONSTRAINT `FK\_2wxuh84xxur2djq175dlkyd0g`

FOREIGN KEY (`photo\_id`)

REFERENCES `food\_service\_db`.`simple\_user` (`id`),

CONSTRAINT `FK\_l62vpvg5psyvxl6q4yo5102gd`

FOREIGN KEY (`shop\_id`)

REFERENCES `food\_service\_db`.`photo` (`id`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

ДОДАТОК Б

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

33

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

**Клас User**

@Entity  
@Inheritance(strategy= InheritanceType.*TABLE\_PER\_CLASS*)  
public abstract class User {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*TABLE*)  
 private Integer id;  
  
 private String firstName;  
 private String lastName;  
  
 @Temporal(TemporalType.*DATE*)  
 private Date dob;  
  
 @Enumerated(EnumType.*STRING*)  
 private Gender gender;  
   
 @Column(unique = true, nullable = false)  
 private String email;  
 @Column(nullable = false)  
 private String password;  
  
 @Lob  
 private String personalData;  
  
 @Transient  
 @OneToOne(targetEntity = Photo.class, fetch = FetchType.*EAGER*)  
 @JoinColumn(name = "photo\_id", insertable = false, updatable = false)  
 private Photo photo;  
  
 /\*\* foreign key \*/  
 @Column(name = "photo\_id")  
 private Integer photoId;  
  
 @Enumerated(EnumType.*STRING*)  
 private UserType userType;  
 /\*\*  
 \* Determins the system status of current user  
 \*/  
 @Enumerated(EnumType.*STRING*)  
 private SystemStatus systemStatus;  
  
  
 public Integer getId() {  
 return id;  
 }  
  
 public void setId(Integer id) {  
 this.id = id;  
 }  
  
 public UserType getUserType() {  
 return userType;  
 }  
  
 public void setUserType(UserType userType) {  
 this.userType = userType;  
 }  
  
 public SystemStatus getSystemStatus() {  
 return systemStatus;  
 }  
  
 public void setSystemStatus(SystemStatus systemStatus) {  
 this.systemStatus = systemStatus;  
 }  
  
 public Integer getPhotoId() {  
 return photoId;  
 }  
  
 public void setPhotoId(Integer photoId) {  
 this.photoId = photoId;  
 }  
  
 public String getPersonalData() {  
 return personalData;  
 }  
  
 public void setPersonalData(String personalData) {  
 this.personalData = personalData;  
 }  
  
 public String getPassword() {  
 return password;  
 }  
  
 public void setPassword(String password) {  
 this.password = password;  
 }  
  
 public String getEmail() {  
 return email;  
 }  
  
 public void setEmail(String email) {  
 this.email = email;  
 }  
  
 public Gender getGender() {  
 return gender;  
 }  
  
 public void setGender(Gender gender) {  
 this.gender = gender;  
 }  
  
 public Date getDob() {  
 return dob;  
 }  
  
 public void setDob(Date dob) {  
 this.dob = dob;  
 }  
  
 public String getFirstName() {  
 return firstName;  
 }  
  
 public void setFirstName(String firstName) {  
 this.firstName = firstName;  
 }  
  
 public String getLastName() {  
 return lastName;  
 }  
  
 public void setLastName(String lastName) {  
 this.lastName = lastName;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
  
 User user = (User) o;  
  
 if (id != null ? !id.equals(user.id) : user.id != null) return false;  
  
 return true;  
 }  
  
 @Override  
 public int hashCode() {  
 return id != null ? id.hashCode() : 0;  
 }  
  
 @Override  
 public String toString() {  
 return "User{" +  
 "id=" + id +  
 ", firstName='" + firstName + '\'' +  
 ", lastName='" + lastName + '\'' +  
 ", dob=" + dob +  
 ", gender=" + gender +  
 ", email='" + email + '\'' +  
 ", password='" + password + '\'' +  
 ", personalData='" + personalData + '\'' +  
 ", photoId=" + photoId +  
 ", userType=" + userType +  
 ", systemStatus=" + systemStatus +  
 '}';  
 }  
}

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

34

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

35

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

36

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

**Клас SimpleUser**

package com.foodservice.entities.user;  
  
import com.fasterxml.jackson.annotation.JsonIgnoreProperties;  
import com.foodservice.entities.data.LazyClonable;  
import com.foodservice.entities.Photo;  
  
import javax.persistence.\*;  
import java.util.ArrayList;  
import java.util.List;  
@JsonIgnoreProperties(value = {"hibernateLazyInitializer", "handler"})  
@Entity  
@javax.persistence.Table(name = "simple\_user")  
public class SimpleUser extends User implements LazyClonable<SimpleUser> {  
  
 @OneToMany(targetEntity = Photo.class, fetch = FetchType.*EAGER*)  
 @JoinTable(name = "simple\_user\_photo",  
 joinColumns = @JoinColumn(name = "photo\_id"),  
 inverseJoinColumns = @JoinColumn(name = "shop\_id"))  
 private List<Photo> photos = new ArrayList<>();  
  
 public List<Photo> getPhotos() {  
 return photos;  
 }  
  
 public void setPhotos(List<Photo> photos) {  
 this.photos = photos;  
 }  
  
 @Override  
 public SimpleUser clone() {  
 SimpleUser object = new SimpleUser();  
 object.setId(this.getId());  
 object.setUserType(this.getUserType());  
 object.setPassword(this.getPassword());  
 object.setEmail(this.getEmail());  
 object.setDob(this.getDob());  
 object.setGender(this.getGender());  
 object.setFirstName(this.getFirstName());  
 object.setLastName(this.getLastName());  
 object.setPersonalData(this.getPersonalData());  
 object.setPhotoId(this.getPhotoId());  
 object.setSystemStatus(this.getSystemStatus());  
 return object;  
 }  
}

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

37

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

**Клас Friendship**

package com.foodservice.entities.friendship;  
  
import com.fasterxml.jackson.annotation.JsonIgnoreProperties;  
import com.foodservice.entities.data.LazyClonable;  
import com.foodservice.entities.data.State;  
import com.foodservice.entities.user.SimpleUser;  
import org.codehaus.jackson.annotate.JsonIgnore;  
  
import javax.persistence.\*;  
  
@JsonIgnoreProperties(value = {"applicant", "acceptor"})  
@Entity  
@javax.persistence.Table(name = "friendship", uniqueConstraints=  
@UniqueConstraint(columnNames = {"applicant\_id", "acceptor\_id"}))  
public class Friendship implements LazyClonable<Friendship> {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Integer id;  
  
// @Transient  
 @ManyToOne(targetEntity = SimpleUser.class, fetch=FetchType.*LAZY*)  
 @JoinColumn(name = "applicant\_id", insertable = false, updatable = false)  
 private SimpleUser applicant;  
  
// @Transient  
 @ManyToOne(targetEntity = SimpleUser.class, fetch=FetchType.*LAZY*)  
 @JoinColumn(name = "acceptor\_id", insertable = false, updatable = false)  
 private SimpleUser acceptor;  
  
  
 /\*\* foreign key \*/  
 @Column(name = "applicant\_id")  
 private Integer applicantId;  
  
 /\*\* foreign key \*/  
 @Column(name = "acceptor\_id")  
 private Integer acceptorId;  
  
  
 @Enumerated(EnumType.*STRING*)  
 private State state;  
  
 public Friendship(int applicantId, int acceptorId, State state) {  
 this.applicantId = applicantId;  
 this.acceptorId = acceptorId;  
 this.state = state;  
 }  
  
 public Friendship() {  
 }  
  
 public Integer getId() {  
 return id;  
 }  
  
 public void setId(Integer id) {  
 this.id = id;  
 }  
  
 public State getState() {  
 return state;  
 }  
  
 public void setState(State state) {  
 this.state = state;  
 }  
  
 public Integer getAcceptorId() {  
 return acceptorId;  
 }  
  
 public void setAcceptorId(Integer acceptorId) {  
 this.acceptorId = acceptorId;  
 }  
  
 public Integer getApplicantId() {  
 return applicantId;  
 }  
  
 public void setApplicantId(Integer applicantId) {  
 this.applicantId = applicantId;  
 }  
  
 public SimpleUser getApplicant() {  
 return applicant;  
 }  
//  
 public void setApplicant(SimpleUser applicant) {  
 this.applicant = applicant;  
 }  
//  
 public SimpleUser getAcceptor() {  
 return acceptor;  
 }  
  
 public void setAcceptor(SimpleUser acceptor) {  
 this.acceptor = acceptor;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
  
 Friendship that = (Friendship) o;  
  
 if (id != null ? !id.equals(that.id) : that.id != null) return false;  
  
 return true;  
 }  
  
 @Override  
 public int hashCode() {  
 return id != null ? id.hashCode() : 0;  
 }  
  
 @Override  
 public String toString() {  
 return "Friendship{" +  
 "id=" + id +  
 ", applicantId=" + applicantId +  
 ", acceptorId=" + acceptorId +  
 ", state=" + state +  
 '}';  
 }  
  
 @Override  
 public Friendship clone() {  
 Friendship friendship = new Friendship();  
 friendship.setId(this.id);  
 friendship.setState(this.state);  
 friendship.setAcceptorId(this.acceptorId);  
 friendship.setApplicantId(this.applicantId);  
 return friendship;  
 }  
}

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

38

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

39

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

40

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

**Клас Message**

package com.foodservice.entities;  
  
import com.foodservice.entities.data.LazyClonable;  
import com.foodservice.entities.user.User;  
  
import javax.persistence.\*;  
import java.util.Date;  
  
@Entity  
@javax.persistence.Table(name = "message")  
public class Message implements LazyClonable<Message> {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Integer id;  
  
 private String text;  
  
 @Temporal(TemporalType.*TIMESTAMP*)  
 private Date time;  
  
 @Transient  
 @ManyToOne(targetEntity = User.class, fetch = FetchType.*LAZY*)  
 @JoinColumn(name = "sender\_id", insertable = false, updatable = false)  
 private User sender;  
  
 @Transient  
 @ManyToOne(targetEntity = User.class, fetch = FetchType.*LAZY*)  
 @JoinColumn(name = "receiver\_id", insertable = false, updatable = false)  
 private User receiver;  
  
 /\*\* foreign key \*/  
 @Column(name = "sender\_id")  
 private Integer senderId;  
  
 /\*\* foreign key \*/  
 @Column(name = "receiver\_id")  
 private Integer receiverId;  
  
 public Integer getId() {  
 return id;  
 }  
  
 public void setId(Integer id) {  
 this.id = id;  
 }  
  
 public Integer getReceiverId() {  
 return receiverId;  
 }  
  
 public void setReceiverId(Integer receiverId) {  
 this.receiverId = receiverId;  
 }  
  
 public Integer getSenderId() {  
 return senderId;  
 }  
  
 public void setSenderId(Integer senderId) {  
 this.senderId = senderId;  
 }  
  
 public String getText() {  
 return text;  
 }  
  
 public void setText(String text) {  
 this.text = text;  
 }  
  
 public Date getTime() {  
 return time;  
 }  
  
 public void setTime(Date time) {  
 this.time = time;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
  
 Message message = (Message) o;  
  
 if (id != null ? !id.equals(message.id) : message.id != null) return false;  
  
 return true;  
 }  
  
 @Override  
 public int hashCode() {  
 return id != null ? id.hashCode() : 0;  
 }  
  
 @Override  
 public String toString() {  
 return "Message{" +  
 "id=" + id +  
 ", text='" + text + '\'' +  
 ", time=" + time +  
 ", senderId=" + senderId +  
 ", receiverId=" + receiverId +  
 '}';  
 }  
  
 @Override  
 public Message clone() {  
 Message message = new Message();  
 message.setId(this.getId());  
 message.setReceiverId(this.getReceiverId());  
 message.setSenderId(this.getSenderId());  
 message.setText(this.getText());  
 message.setTime(this.getTime());  
 return message;  
 }  
}

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

41

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

42

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

**Клас Photo**

package com.foodservice.entities;  
  
import com.foodservice.entities.data.LazyClonable;  
  
import javax.persistence.\*;  
import java.util.Arrays;  
  
@Entity  
@javax.persistence.Table(name = "photo")  
public class Photo implements LazyClonable<Photo> {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*AUTO*)  
 private Integer id;  
  
 @Lob  
 private byte[] image;  
  
 private String name;  
  
 public Integer getId() {  
 return id;  
 }  
  
 public void setId(Integer id) {  
 this.id = id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public byte[] getImage() {  
 return image;  
 }  
  
 public void setImage(byte[] image) {  
 this.image = image;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
  
 Photo photo = (Photo) o;  
  
 if (id != null ? !id.equals(photo.id) : photo.id != null) return false;  
  
 return true;  
 }  
  
 @Override  
 public int hashCode() {  
 return id != null ? id.hashCode() : 0;  
 }  
  
 @Override  
 public String toString() {  
 return "Photo{" +  
 "id=" + id +  
 ", image=" + Arrays.*toString*(image) +  
 ", name='" + name + '\'' +  
 '}';  
 }  
  
 @Override  
 public Photo clone() {  
 return this;  
 }  
}

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

43

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

44

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

**Клас SimpleUserResource**

package com.foodservice.resources;  
  
import com.foodservice.entities.data.Gender;  
import com.foodservice.entities.data.SystemStatus;  
import com.foodservice.entities.user.SimpleUser;  
import com.foodservice.services.SimpleUserService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Component;  
  
import javax.ws.rs.\*;  
import javax.ws.rs.core.MediaType;  
import javax.ws.rs.core.Response;  
import java.util.HashMap;  
import java.util.List;  
import java.util.Map;  
  
@Component  
@Path("/users/simple")  
public class SimpleUserResource {  
   
 private SimpleUserService simpleUserService;  
  
 @Autowired  
 public void setSimpleUserService(SimpleUserService simpleUserService) {  
 this.simpleUserService = simpleUserService;  
 }  
  
 @POST  
 @Consumes("application/json")  
 @Produces("application/json")  
 public Response create(SimpleUser simpleUser) {  
 try {  
 Integer id = simpleUserService.create(simpleUser);  
 return Response.*ok*(id).status(Response.Status.*CREATED*).build();  
 } catch (Exception e) {  
 e.printStackTrace();  
 return Response.*status*(Response.Status.*NOT\_ACCEPTABLE*).build();  
 }  
 }  
  
 @GET  
 @Path("/byId")  
 @Produces(MediaType.*APPLICATION\_JSON*)  
 public Response get(@QueryParam("id") int id) {  
 try {  
 SimpleUser simpleUser = simpleUserService.get(id);  
 return Response.*ok*(simpleUser).status(Response.Status.*OK*).build();  
 } catch (Exception e) {  
 e.printStackTrace();  
 return Response.*status*(Response.Status.*INTERNAL\_SERVER\_ERROR*).build();  
 }  
 }  
  
 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*friendship support-start\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//  
  
 @GET  
 @Path("/friendsOf")  
 @Produces(MediaType.*APPLICATION\_JSON*)  
 public Response getFriends(@QueryParam("id") int id,  
 @DefaultValue("") @QueryParam("firstNameLike") String firstNameLike,  
 @DefaultValue("") @QueryParam("lastNameLike") String lastNameLike,  
 @QueryParam("minAge") Integer minAge,  
 @QueryParam("maxAge") Integer maxAge,  
 @DefaultValue("") @QueryParam("systemStatus") SystemStatus systemStatus,  
 @DefaultValue("") @QueryParam("gender") Gender gender,  
 @DefaultValue("0") @QueryParam("firstResult") int firstResult,  
 @DefaultValue("10000") @QueryParam("maxResults") int maxResults) {  
 try {  
 Map<String, Object> map = new HashMap<>();  
 map.put("firstNameLike", firstNameLike);  
 map.put("lastNameLike", lastNameLike);  
 map.put("ageMax", minAge);  
 map.put("ageMin", maxAge);  
 map.put("gender", gender);  
 map.put("systemStatus", systemStatus);  
 List<SimpleUser> simpleUsers = simpleUserService.getFriends(id, map);  
 System.*out*.println("simpleUser's friends size: " + simpleUsers.size());  
 return Response.*ok*(simpleUsers).status(Response.Status.*OK*).build();  
 } catch (Exception e) {  
 e.printStackTrace();  
 return Response.*status*(Response.Status.*INTERNAL\_SERVER\_ERROR*).build();  
 }  
 }  
  
 @GET  
 @Path("/followersOf")  
 @Produces(MediaType.*APPLICATION\_JSON*)  
 public Response getFollowers(@QueryParam("id") int id,  
 @DefaultValue("") @QueryParam("firstNameLike") String firstNameLike,  
 @DefaultValue("") @QueryParam("lastNameLike") String lastNameLike,  
 @QueryParam("minAge") Integer minAge,  
 @QueryParam("maxAge") Integer maxAge,  
 @DefaultValue("") @QueryParam("systemStatus") SystemStatus systemStatus,  
 @DefaultValue("") @QueryParam("gender") Gender gender,  
 @DefaultValue("0") @QueryParam("firstResult") int firstResult,  
 @DefaultValue("10000") @QueryParam("maxResults") int maxResults) {  
 try {  
 Map<String, Object> map = new HashMap<>();  
 map.put("firstNameLike", firstNameLike);  
 map.put("lastNameLike", lastNameLike);  
 map.put("ageMax", minAge);  
 map.put("ageMin", maxAge);  
 map.put("gender", gender);  
 map.put("systemStatus", systemStatus);  
 List<SimpleUser> simpleUsers = simpleUserService.getFollowers(id, map);  
 return Response.*ok*(simpleUsers).status(Response.Status.*OK*).build();  
 } catch (Exception e) {  
 e.printStackTrace();  
 return Response.*status*(Response.Status.*INTERNAL\_SERVER\_ERROR*).build();  
 }  
 }  
  
 @GET  
 @Path("/followedBy")  
 @Produces(MediaType.*APPLICATION\_JSON*)  
 public Response getFollowedBy(@QueryParam("id") int id,  
 @DefaultValue("") @QueryParam("firstNameLike") String firstNameLike,  
 @DefaultValue("") @QueryParam("lastNameLike") String lastNameLike,  
 @QueryParam("minAge") Integer minAge,  
 @QueryParam("maxAge") Integer maxAge,  
 @DefaultValue("") @QueryParam("systemStatus") SystemStatus systemStatus,  
 @DefaultValue("") @QueryParam("gender") Gender gender,  
 @DefaultValue("0") @QueryParam("firstResult") int firstResult,  
 @DefaultValue("10000") @QueryParam("maxResults") int maxResults) {  
 try {  
 Map<String, Object> map = new HashMap<>();  
 map.put("firstNameLike", firstNameLike);  
 map.put("lastNameLike", lastNameLike);  
 map.put("ageMax", minAge);  
 map.put("ageMin", maxAge);  
 map.put("gender", gender);  
 map.put("systemStatus", systemStatus);  
 List<SimpleUser> simpleUsers = simpleUserService.getFollowedBy(id, map);  
 return Response.*ok*(simpleUsers).status(Response.Status.*OK*).build();  
 } catch (Exception e) {  
 e.printStackTrace();  
 return Response.*status*(Response.Status.*INTERNAL\_SERVER\_ERROR*).build();  
 }  
 }  
  
 @GET  
 @Path("/requestedTo")  
 @Produces(MediaType.*APPLICATION\_JSON*)  
 public Response getRequestedTo(@QueryParam("id") int id,  
 @DefaultValue("") @QueryParam("firstNameLike") String firstNameLike,  
 @DefaultValue("") @QueryParam("lastNameLike") String lastNameLike,  
 @QueryParam("minAge") Integer minAge,  
 @QueryParam("maxAge") Integer maxAge,  
 @DefaultValue("") @QueryParam("systemStatus") SystemStatus systemStatus,  
 @DefaultValue("") @QueryParam("gender") Gender gender,  
 @DefaultValue("0") @QueryParam("firstResult") int firstResult,  
 @DefaultValue("10000") @QueryParam("maxResults") int maxResults) {  
 try {  
 Map<String, Object> map = new HashMap<>();  
 map.put("firstNameLike", firstNameLike);  
 map.put("lastNameLike", lastNameLike);  
 map.put("ageMax", minAge);  
 map.put("ageMin", maxAge);  
 map.put("gender", gender);  
 map.put("systemStatus", systemStatus);  
 List<SimpleUser> simpleUsers = simpleUserService.getRequestedTo(id, map);  
 System.*out*.println("REQUESTED TO size: " + simpleUsers.size());  
 return Response.*ok*(simpleUsers).status(Response.Status.*OK*).build();  
 } catch (Exception e) {  
 e.printStackTrace();  
 return Response.*status*(Response.Status.*INTERNAL\_SERVER\_ERROR*).build();  
 }  
 }  
  
 @GET  
 @Path("/requestedBy")  
 @Produces(MediaType.*APPLICATION\_JSON*)  
 public Response getRequestedBy(@QueryParam("id") int id,  
 @DefaultValue("") @QueryParam("firstNameLike") String firstNameLike,  
 @DefaultValue("") @QueryParam("lastNameLike") String lastNameLike,  
 @QueryParam("minAge") Integer minAge,  
 @QueryParam("maxAge") Integer maxAge,  
 @DefaultValue("") @QueryParam("systemStatus") SystemStatus systemStatus,  
 @DefaultValue("") @QueryParam("gender") Gender gender,  
 @DefaultValue("0") @QueryParam("firstResult") int firstResult,  
 @DefaultValue("10000") @QueryParam("maxResults") int maxResults) {  
 try {  
 Map<String, Object> map = new HashMap<>();  
 map.put("firstNameLike", firstNameLike);  
 map.put("lastNameLike", lastNameLike);  
 map.put("ageMax", minAge);  
 map.put("ageMin", maxAge);  
 map.put("gender", gender);  
 map.put("systemStatus", systemStatus);  
 List<SimpleUser> simpleUsers = simpleUserService.getRequestedBy(id, map);  
 for (SimpleUser simpleUser :simpleUsers)  
 System.*out*.println(simpleUser);  
 System.*out*.println("REQUESTED BY size: " + simpleUsers.size());  
 return Response.*ok*(simpleUsers).status(Response.Status.*OK*).build();  
 } catch (Exception e) {  
 e.printStackTrace();  
 return Response.*status*(Response.Status.*INTERNAL\_SERVER\_ERROR*).build();  
 }  
 }  
  
  
 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*friendship support-end\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//  
  
 @GET  
 @Path("/byCriterion")  
 @Produces(MediaType.*APPLICATION\_JSON*)  
 public Response getByCriterion(@DefaultValue("") @QueryParam("firstNameLike") String firstNameLike,  
 @DefaultValue("") @QueryParam("lastNameLike") String lastNameLike,  
 @QueryParam("minAge") Integer minAge,  
 @QueryParam("maxAge") Integer maxAge,  
 @DefaultValue("") @QueryParam("systemStatus") SystemStatus systemStatus,  
 @DefaultValue("") @QueryParam("gender") Gender gender,  
 @DefaultValue("0") @QueryParam("firstResult") int firstResult,  
 @DefaultValue("10000") @QueryParam("maxResults") int maxResults) {  
 try {  
 Map<String, Object> map = new HashMap<>();  
 map.put("firstNameLike", firstNameLike);  
 map.put("lastNameLike", lastNameLike);  
 map.put("ageMax", minAge);  
 map.put("ageMin", maxAge);  
 map.put("gender", gender);  
 map.put("systemStatus", systemStatus);  
 List<SimpleUser> simpleUsers = simpleUserService.getByCriterion(map);  
 return Response.*ok*(simpleUsers).status(Response.Status.*OK*).build();  
 } catch (Exception e) {  
 e.printStackTrace();  
 return Response.*status*(Response.Status.*INTERNAL\_SERVER\_ERROR*).build();  
 }  
 }  
  
  
 @GET  
 @Path("/byEmail")  
 @Produces(MediaType.*APPLICATION\_JSON*)  
 public Response get(@QueryParam("email") String email) {  
 try {  
 SimpleUser simpleUser = simpleUserService.getByEmail(email);  
 return Response.*ok*(simpleUser).status(Response.Status.*OK*).build();  
 } catch (Exception e) {  
 e.printStackTrace();  
 return Response.*status*(Response.Status.*NOT\_ACCEPTABLE*).build();  
 }  
 }  
  
 @PUT  
 @Consumes(MediaType.*APPLICATION\_JSON*)  
 public Response update(SimpleUser simpleUser) {  
 try {  
 simpleUserService.update(simpleUser);  
 return Response.*status*(Response.Status.*OK*).build();  
 } catch (Exception e) {  
 e.printStackTrace();  
 return Response.*status*(Response.Status.*NOT\_ACCEPTABLE*).build();  
 }  
 }  
  
 @DELETE  
 @Consumes(MediaType.*APPLICATION\_JSON*)  
 public Response delete(SimpleUser simpleUser) {  
 try {  
 simpleUserService.delete(simpleUser);  
 return Response.*status*(Response.Status.*OK*).build();  
 } catch (Exception e) {  
 e.printStackTrace();  
 return Response.*status*(Response.Status.*NOT\_ACCEPTABLE*).build();  
 }  
 }  
   
}

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

45

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

46

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

47

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

48

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

49

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата

Інв. № пІдп

Підп. і дата

Взаєм. інв. №

Інв. № дубл.

Підп. і дата

Арк.

50

6.050102 «Комп'ютерна інженерія»

Змн.

№ докум.

Арк.

Підпис

Дата