1. **Тестування методу AddUser()**

**AddUser()** являє собою метод, що перевіряє введені користувачем під час реєстрації поля та проводить їхню валідацію. Метод виглядає наступним образом:

@PostMapping("/registration\_form")

public String addUser(@Valid User user, BindingResult bindingResult) {

if (bindingResult.hasErrors()) {

return "registration\_form";

}

user.setActive(true);

user.setRoles("ROLEUSER");

user.setPassword(passwordEncoder.encode(user.getPassword()));

userService.saveUser(user);

return "redirect:/";

}

Тобто введені значення полей перевіряються відповідно до обмежень, що накладено на них у певному класі. Якщо введенне значення відповідає обмеженню, то юзеру надається роль “USER” та значення полей класів додаються у відповідні поля таблиць в базу даних.

Продемонструємо **клас User**, де можна побачити власне обмеження на кожне поле:

@Id

@GeneratedValue

@Column(name = "user\_id")

private long user\_id;

@Size(min = 2, max= 50, message = "Last name should be from 2 to 50 characters")

@Column(name = "last\_name")

private String lastName;

@Size(min = 2, max= 50, message = "First name should be from 2 to 50 characters")

@Column(name = "first\_name")

private String firstName;

@Size(min = 2, max= 50, message = "Patronymic should be from 2 to 50 characters")

@Column(name = "patronymic")

private String patronymic;

@Pattern(regexp = "^([a-z0-9\_-]+\\.)\*[a-z0-9\_-]+@[a-z0-9\_-]+(\\.[a-z0-9\_-]+)\*\\.[a-z]{2,6}$",

message = "Invalid email format")

@Column(name = "userName")

private String userName;

@Pattern(regexp = "^\\+\\d{2}\\(\\d{3}\\)\\d{3}-\\d{2}-\\d{2}$",

message = "Invalid phone number")

@Column(name = "phone")

private String phone;

@Size(min = 2, max= 50, message = "City should be from 2 to 50 characters")

@Column(name = "city")

private String city;

@Size(min = 2, max= 200, message = "Password should be from 6 to 200 characters")

@Column(name = "passwd")

private String password;

@Column(name = "active")

private boolean active;

@Column(name = "roles")

private String roles;

1. Розглянемо випадок, коли на вхід поступає **пустий об’єкт.**

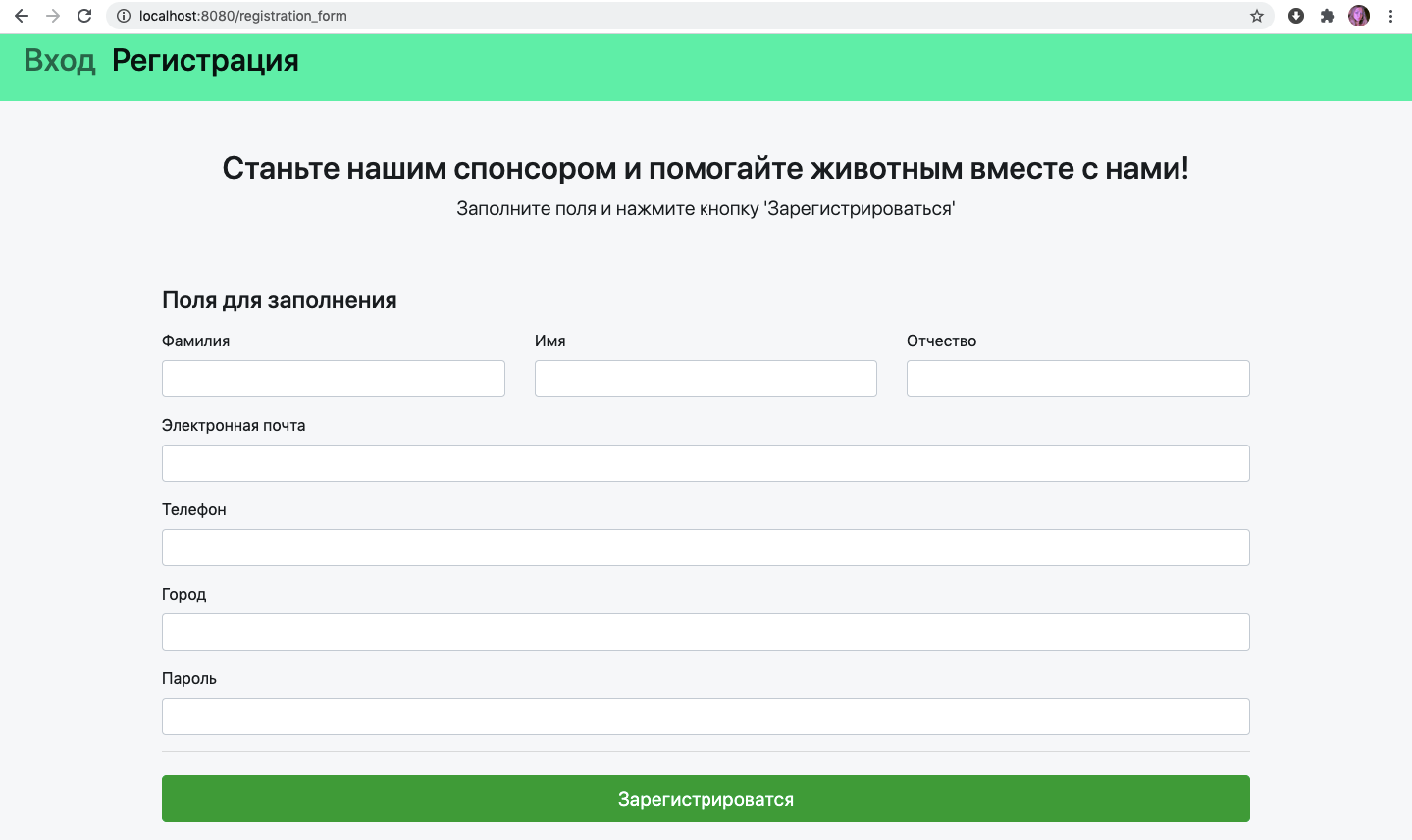


Рис. 1 – Пустий об’єкт під час реєстрації

Система працює таким чином, що повідомлення про помилку передаєтся прямо на екран за допомогою змінної *bindingResult.* Та, звичайно, у консолі нема повідомлення про додавання нового об’єкту. Користувача перенаправлено знову на сторінку реєстрації, виведено повідомлення про помилки та об’єкт не створено.

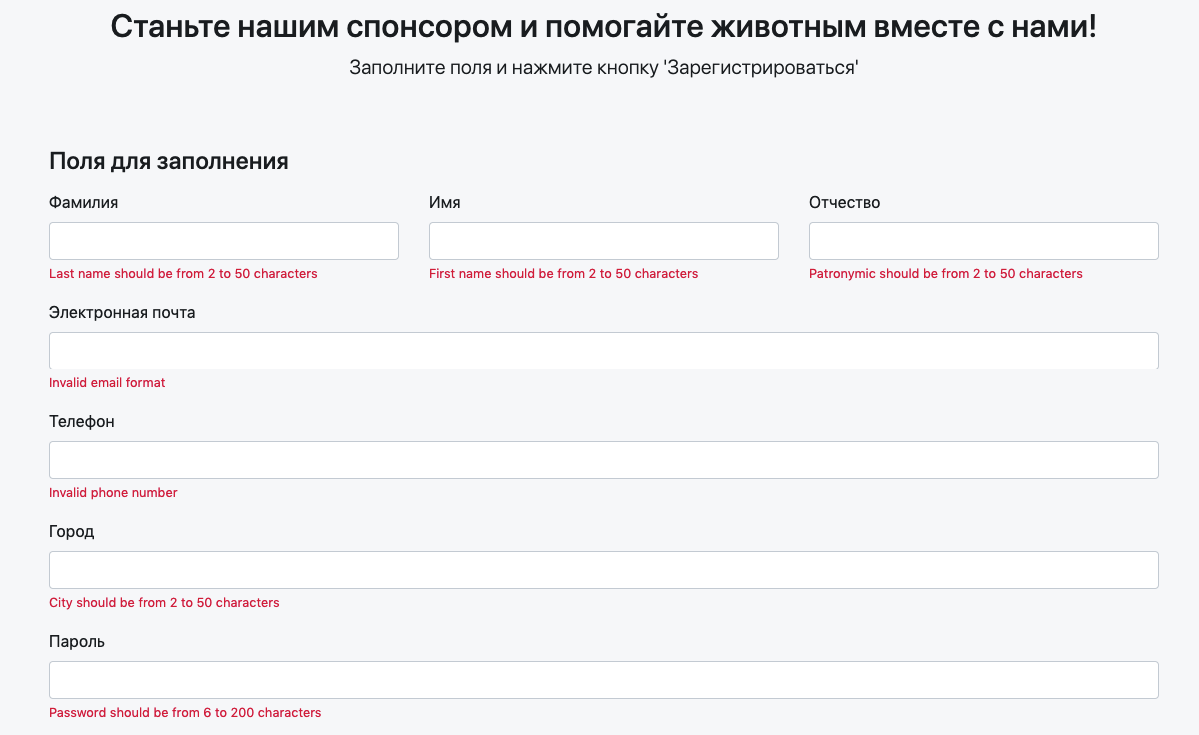


Рис. 2 – Повідомлення про помилку під час тесту №1

1. Розглянемо випадок, коли на вхід поступає **об’єкт з неправильним полем *прізвище*** {1, Диана, Вячеславовна, [dianaaa@gmail.com](mailto:dianaaa@gmail.com), +380931263743, Одесса, 1234}

Система працює таким чином, що повідомлення про помилку передаєтся прямо на екран за допомогою змінної *bindingResult.* Прізвище має бути текстом обсягом від 2 до 50 символів. Та, звичайно, у консолі нема повідомлення про додавання нового об’єкту. Користувача перенаправлено знову на сторінку реєстрації, виведено повідомлення про помилки та об’єкт не створено.

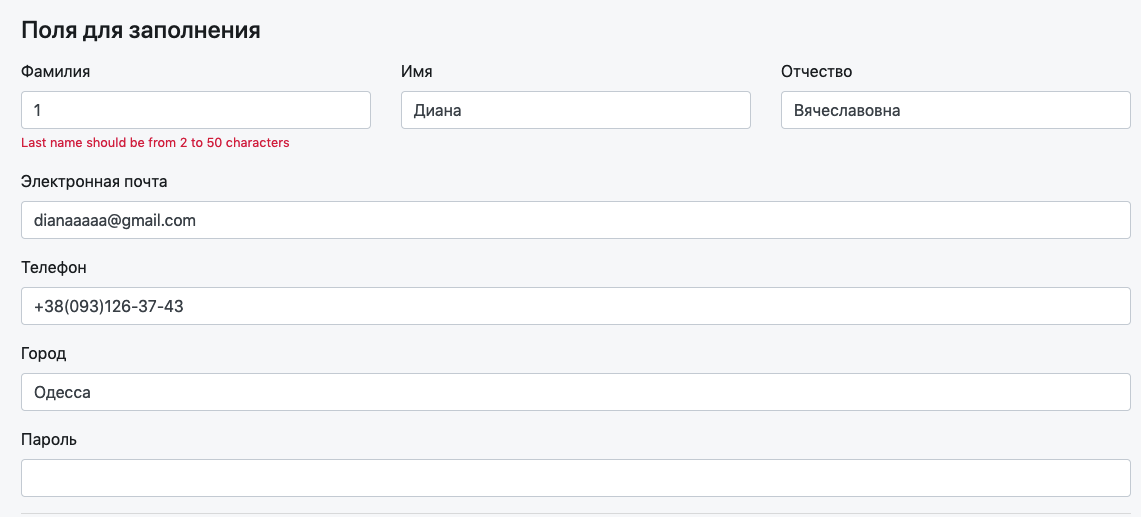


Рис. 3 – Повідомлення про помилку під час тесту №2

1. Розглянемо випадок, коли на вхід поступає **об’єкт з неправильним полем *електронна адреса*** {Соловйова, Диана, Вячеславовна, [dianaaa](mailto:dianaaa@gmail.com), +380931263743, Одесса, 1234}

Система працює таким чином, що повідомлення про помилку передаєтся прямо на екран за допомогою змінної *bindingResult.* На пошту накладено регулярне вираження *regexp = "^([a-z0-9\_-]+\\.)\*[a-z0-9\_-]+@[a-z0-9\_-]+(\\.[a-z0-9\_-]+)\*\\.[a-z]{2,6}$"*. Та, звичайно, у консолі нема повідомлення про додавання нового об’єкту. Користувача перенаправлено знову на сторінку реєстрації, виведено повідомлення про помилки та об’єкт не створено.

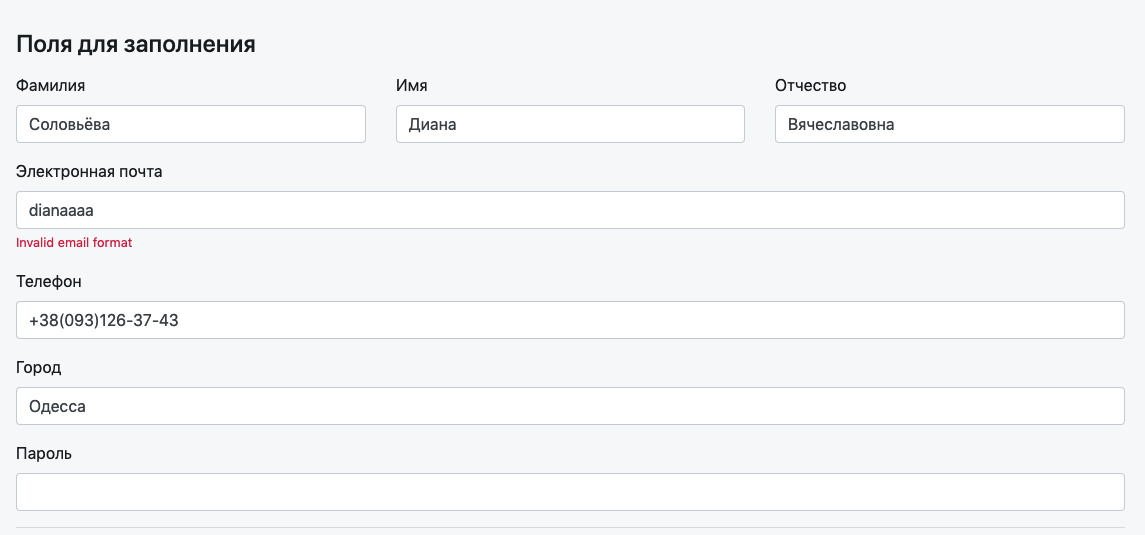


Рис. 4 – Повідомлення про помилку під час тесту №3

1. Розглянемо випадок, коли на вхід поступає **об’єкт з неправильним полем номер *телефону*** {Соловйова, Диана, Вячеславовна, [dianaaa](mailto:dianaaa@gmail.com), 094125, Одесса, 1234}

Система працює таким чином, що повідомлення про помилку передаєтся прямо на екран за допомогою змінної *bindingResult.* На пошту накладено регулярне вираження *regexp = "^\\+\\d{2}\\(*[*\\d{3}\\)\\d{3}-\\d{2}-\\d{2}$*](file://d%7b3%7d//)//d%7b3%7d-//d%7b2%7d-//d%7b2%7d$)*".* Та, звичайно, у консолі нема повідомлення про додавання нового об’єкту. Користувача перенаправлено знову на сторінку реєстрації, виведено повідомлення про помилки та об’єкт не створено.

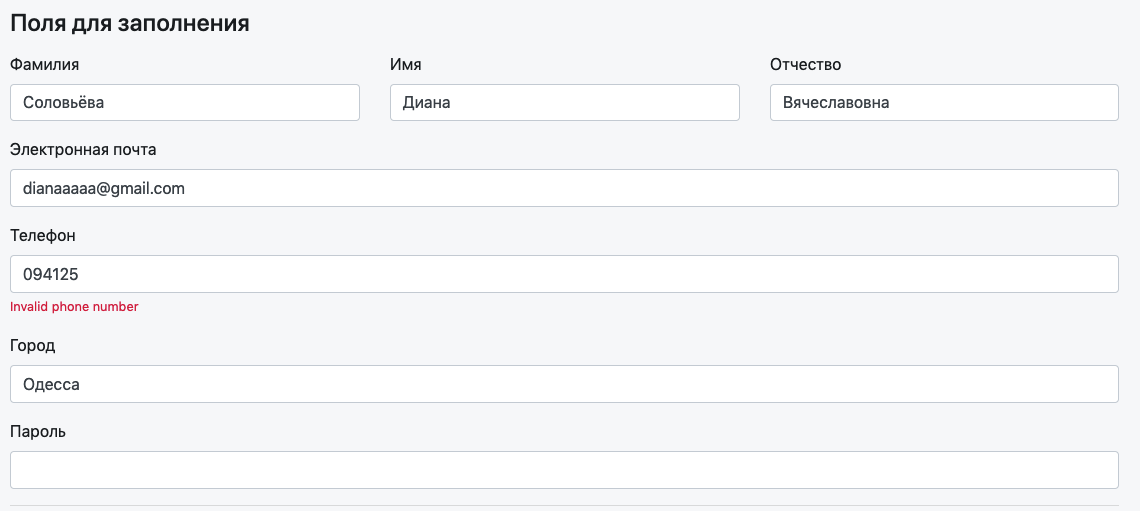


Рис. 5 – Повідомлення про помилку під час тесту №4

1. Розглянемо випадок, коли на вхід поступає **об’єкт з усіма правильними полями**{Поліщук, Катерина, Андріївна, [kate@gmail.com](mailto:kate@gmail.com), +380634564576, Одеса, 1234}

Система працює таким чином, у консолі виводиться повідомлення про додавання нового об’єкту. Бачимо наступне:

Hibernate: select nextval ('hibernate\_sequence')

Hibernate: insert into users (active, city, first\_name, last\_name, passwd, patronymic, phone, roles, user\_name, user\_id) values (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)

Отже, об’єкт створено успішно та користувача перенаправлено у його особистий кабінет, що містить щойно введені дані.

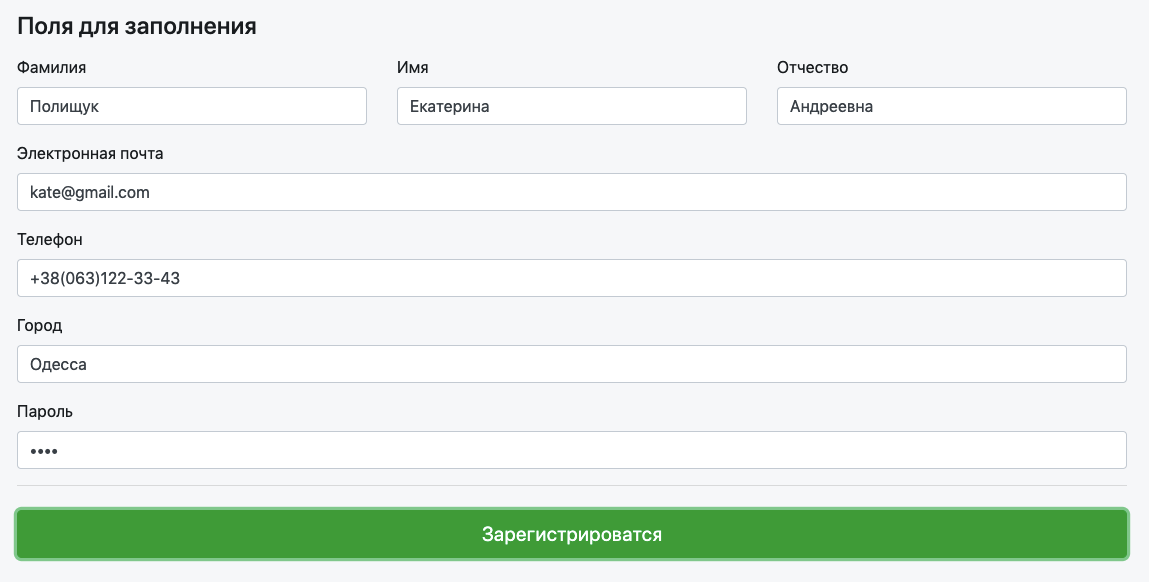


Рис. 6 – Успішне введення даних

1. **Тестування методу AddAnimal()**

**AddAnimal()** являє собою метод, що перевіряє введені користувачем під час додавання нової анкети поля та проводить їхню валідацію. Цей метод є доступним лише у адмінській версії сайту. Метод виглядає наступним образом:

@PostMapping("/admin/add\_animal")

public String processAddAnimal(Valid Animal animal, BindingResult bindingResult, @RequestParam("file") MultipartFile file) {

if (bindingResult.hasErrors()) {

return "admin/add\_child";

}

// normalize the file path

String fileName = StringUtils.cleanPath(file.getOriginalFilename());

// save the file on the local file system

try {

Path path = Paths.get("/Users/disvik/Documents/ONPU/3course/course\_work/src/main/resources/static/images/" + fileName);

Files.copy(file.getInputStream(), path, StandardCopyOption.REPLACE\_EXISTING);

child.setPhotoName("../images/" + fileName);

} catch (IOException e) {

e.printStackTrace();

}

if(animal.getIllness().length()<5)

animal.setIllness(null);

service.saveAnimal(animal);

return "redirect:/admin/index";

}

Тобто введені значення полей перевіряються відповідно до обмежень, що накладено на них у певному класі. Якщо введенне значення відповідає обмеженню, то фото зберігається на відповідний ресурс у базі даних та значення полей класів додаються у відповідні поля таблиць в базу даних.

Продемонструємо **клас Animal**, де можна побачити власне обмеження на кожне поле:

@Id

@GeneratedValue

private long id;

@Column(name = "first\_name")

@Size(min = 2, max= 50, message = "First name should be from 2 to 50 characters")

private String firstName;

@Column(name = "year\_of\_birth")

@Range(min = 2000, max= 2020, message = "Year of birth should be from 2000 to 2020")

private int yearOfBirth;

@Column(name = "gender")

private com.example.demo.animal.Animal.Gender gender;

@Size(min = 4, max= 6, message = "Code should be from 4 to 6 characters")

@Column(name = "code")

private String code;

@Size(min = 2, max= 50, message = "Description should be from 2 to 50 characters")

@Column(name = "about")

private String about;

@Column(name = "age")

private int age;

@Column (name = "photo\_name")

private String photoName;

@Column (name = "illness")

private String illness;

1. Розглянемо випадок, коли на вхід поступає **пустий об’єкт.**

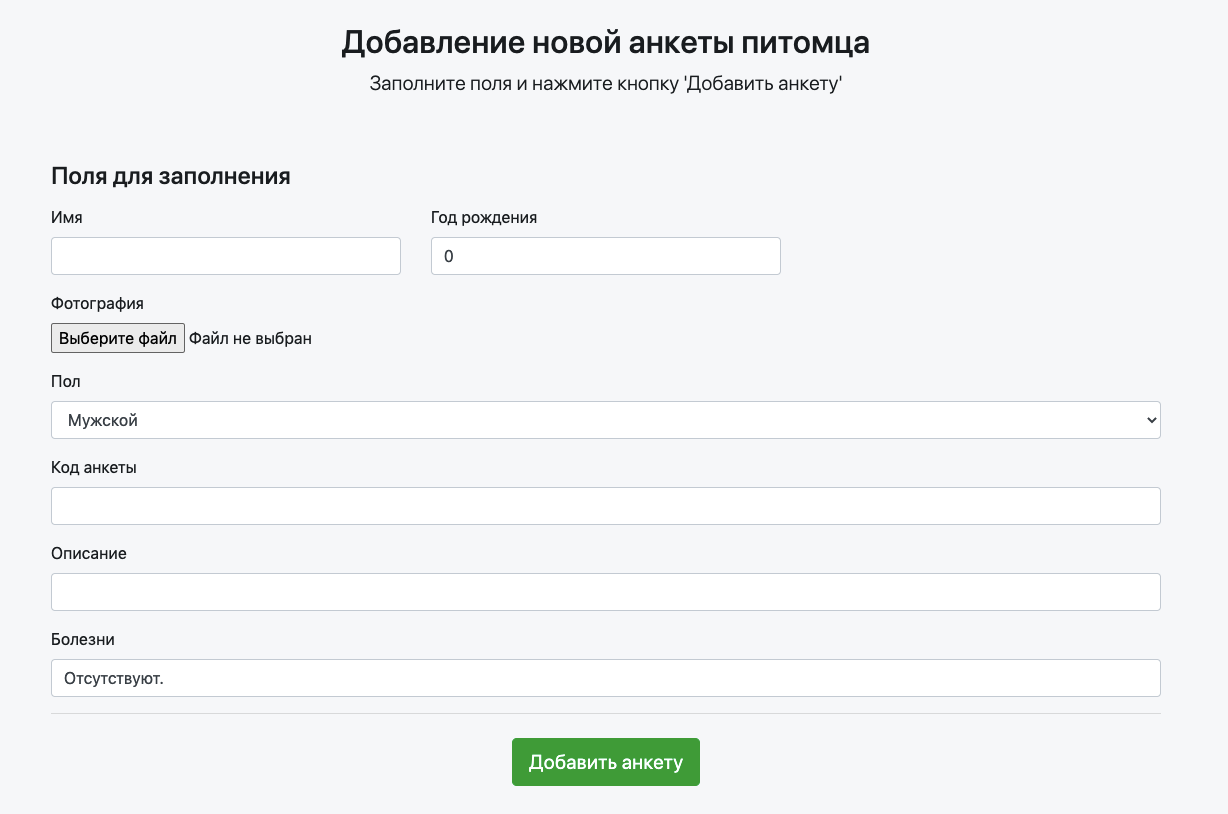


Рис. 7 – Пустий об’єкт під час додавання нової анкети

Система працює таким чином, що повідомлення про помилку передаєтся прямо на екран за допомогою змінної *bindingResult.* Та, звичайно, у консолі нема повідомлення про додавання нового об’єкту. Користувача (адміністратора) перенаправлено знову на сторінку додавання анкети знову, виведено повідомлення про помилки та об’єкт не створено.

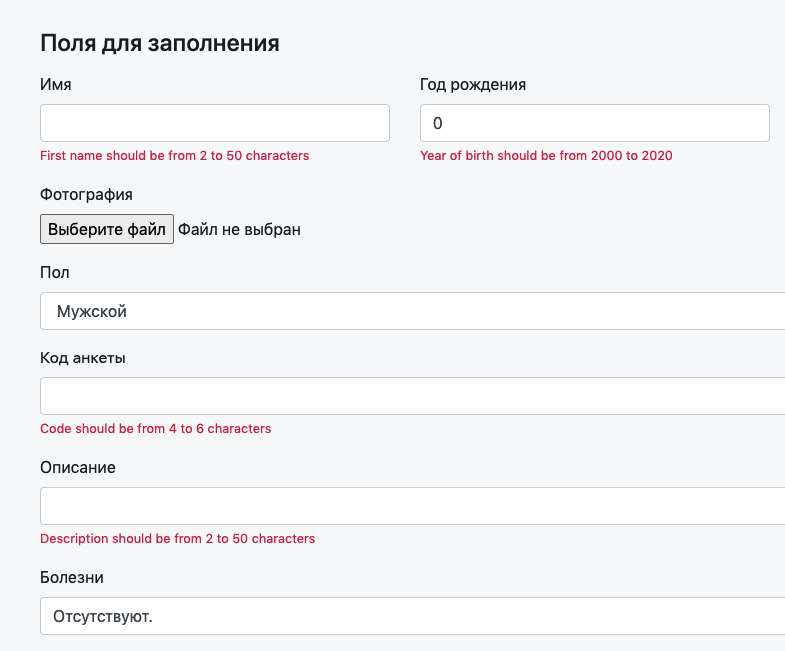


Рис. 8 – Повідомлення про помилку під час тесту №1

1. Розглянемо випадок, коли на вхід поступає **об’єкт з неправильним полем *рік народження***{Пушинка, *1910*, Женский, 0701}

Система працює таким чином, що повідомлення про помилку передаєтся прямо на екран за допомогою змінної *bindingResult.* Дата народження має бути цілим числом від 2000 до 2020. Та, звичайно, у консолі нема повідомлення про додавання нового об’єкту. Користувача перенаправлено знову на сторінку реєстрації, виведено повідомлення про помилки та об’єкт не створено.

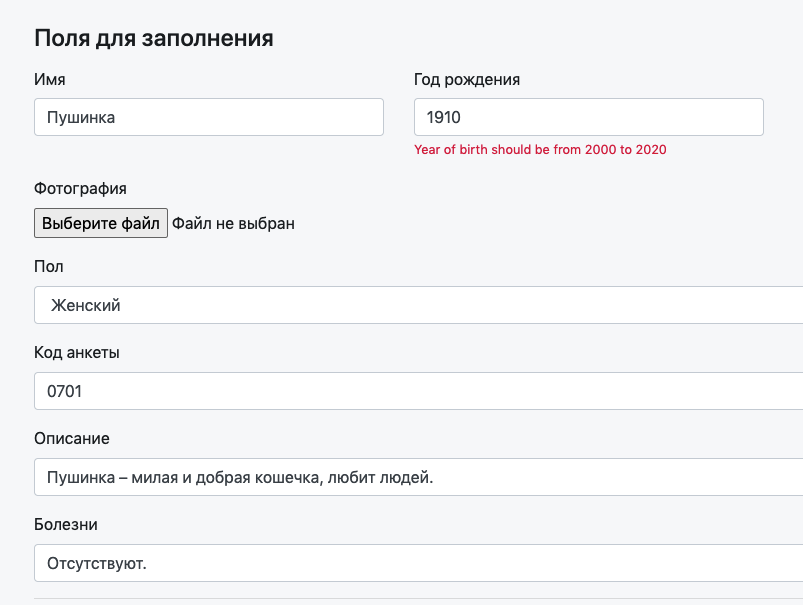


Рис. 9 – Повідомлення про помилку під час тесту №2

1. Розглянемо випадок, коли на вхід поступає **об’єкт з неправильним полем *рік ім’я*** {\_\_\_\_, 2019, *1910*, Мужской, 0701}

Система працює таким чином, що повідомлення про помилку передаєтся прямо на екран за допомогою змінної *bindingResult.* Ім’я має бути строкою від 2 до 50 символів. Та, звичайно, у консолі нема повідомлення про додавання нового об’єкту. Користувача перенаправлено знову на сторінку реєстрації, виведено повідомлення про помилки та об’єкт не створено.

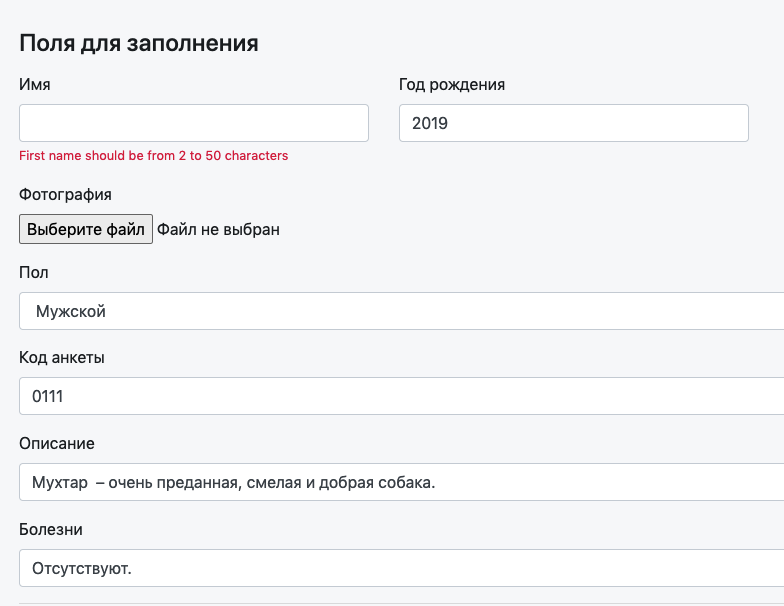


Рис. 10 – Повідомлення про помилку під час тесту №3

1. Розглянемо випадок, коли на вхід поступає **об’єкт з усіма правильними полями**{Мухтар, 2020, 2.png, Мужской, 0101}

Система працює таким чином, у консолі виводиться повідомлення про додавання нового об’єкту. Бачимо наступне:

Hibernate: select nextval ('hibernate\_sequence')

Hibernate: insert into animal (age, code, first\_name, gender, description, illness, photo\_name, year\_of\_birth, id) values (?, ?, ?, ?, ?, ?, ?, ?, ?)

Отже, об’єкт створено успішно та користувача (адміністратора) перенаправлено на сторінку з переліком тварин, що вже містить щойно введену інформацію.

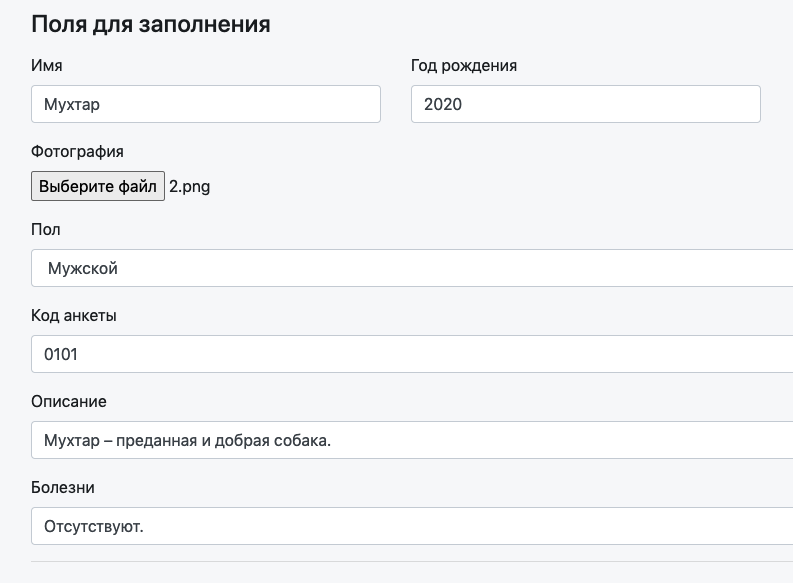


Рис. 11 – Успішне введення даних