**Отчёт по проекту по РПБД.**

**Цель**

Разработать сайт питомник для собак .

**Исходные данные**

База данных с таблицами:

* Животные
* Щенки
* Клиенты
* Заявки
* Работники

**Модельные представления**

Веб приложения собачьего питомника, где реализована регистрация, авторизация, если админ, то возможность редактировать, удалять и добавлять, для всех пользователей просмотр, в том числе и не зарегистрированных, для обычных пользователей возможность покупки собаки.

**Ожидаемый результат**

Веб приложение, интегрированное с базой данный, реализованное на *Node.js, framework – Express.js* для соединения *front-end* c *back-end.*

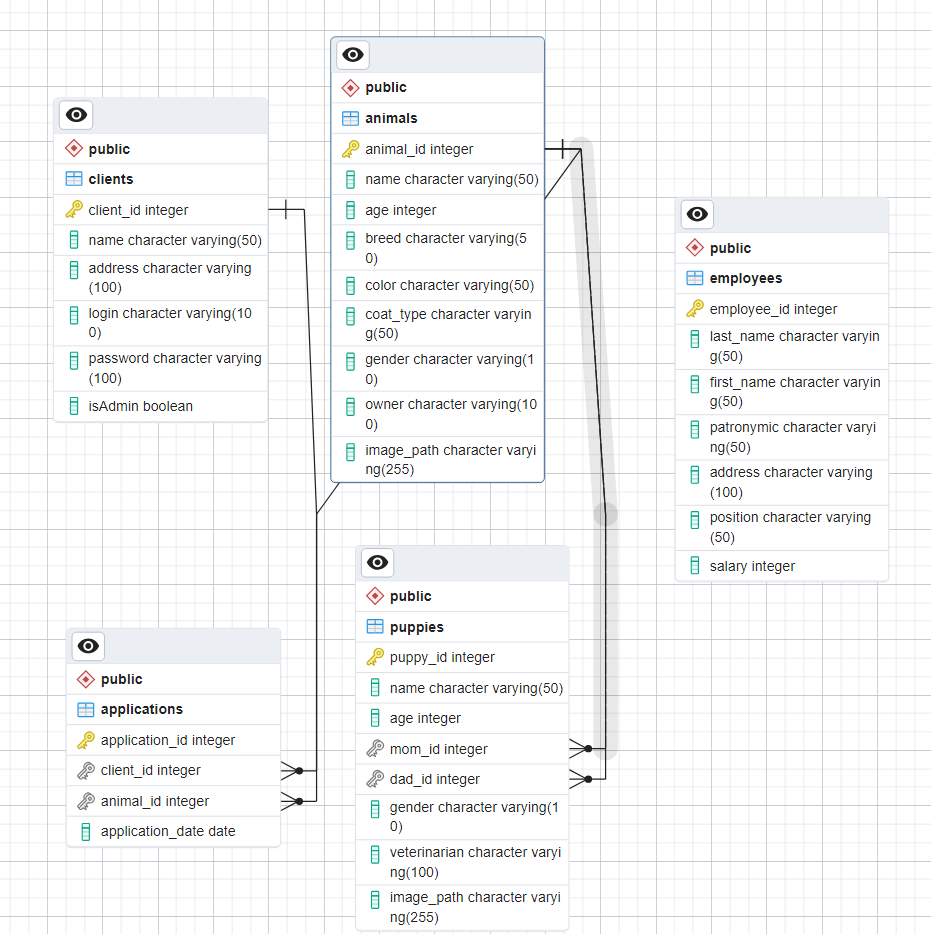
**Критерии оценки результата**

* Разработанная функциональность приложения.
* Интеграция приложения с базой данных.

**Средства реализации:**

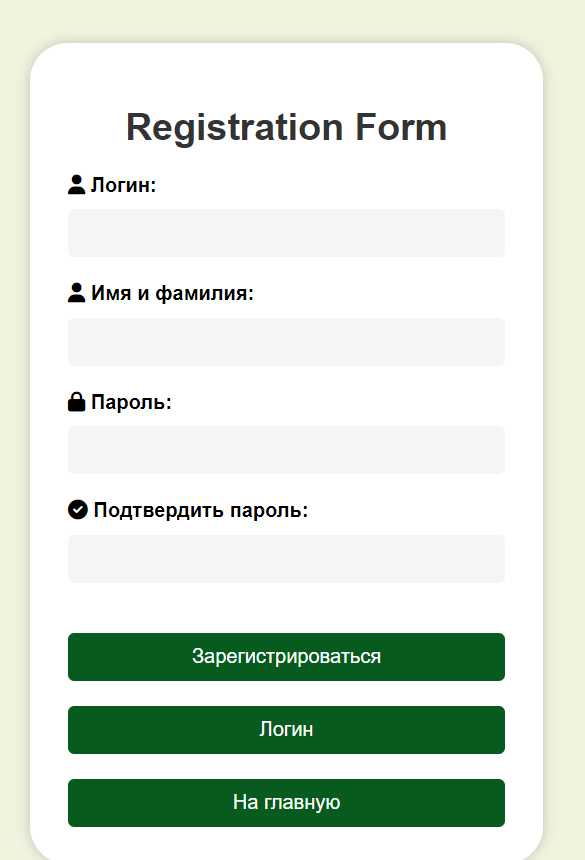
* *Visual Studio code*
* *HTML*
* *CSS*
* *JavaScript*
* *Node.js*
* *Express.js*
* *PostgreSQL*

**Таблица связей**

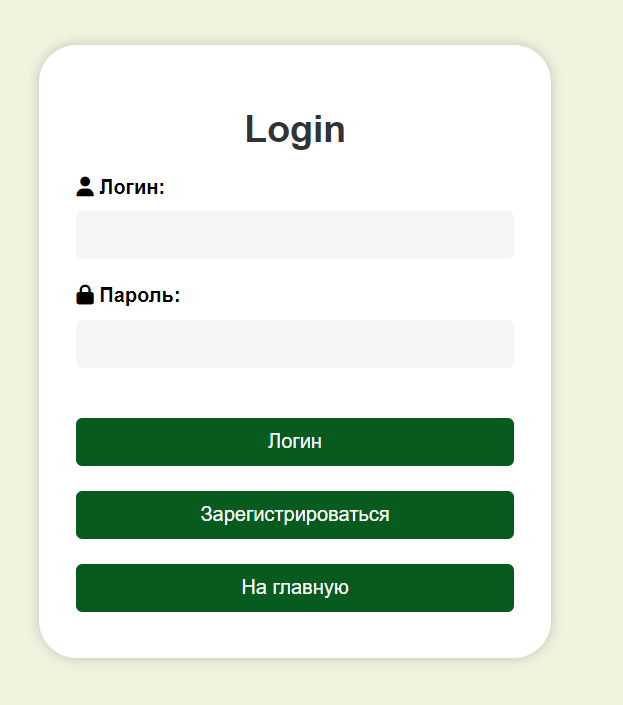


*Рис.1. Таблица связей*

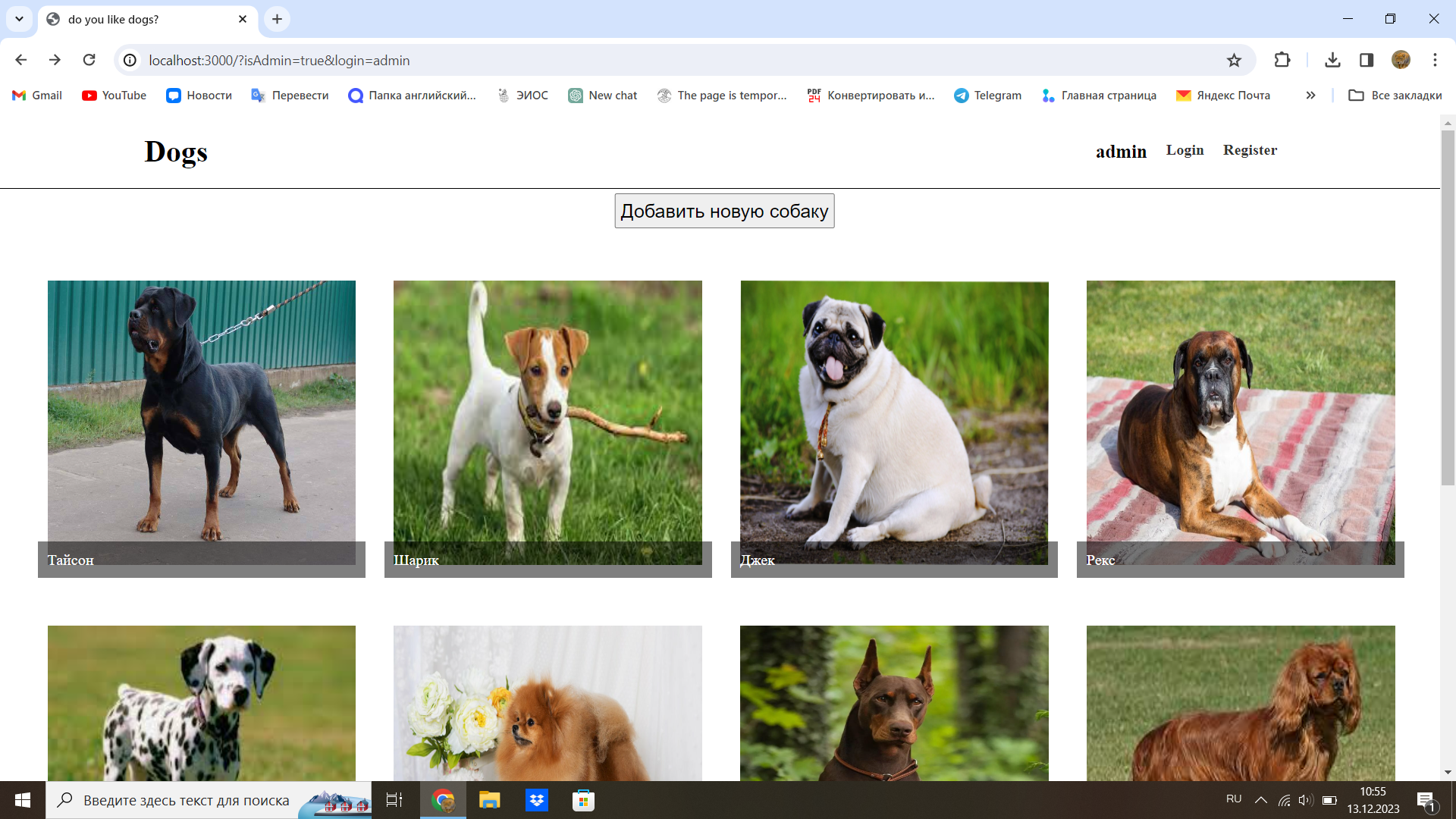
**Интерфейс**

****

*Рис.2. Регистрация*

**

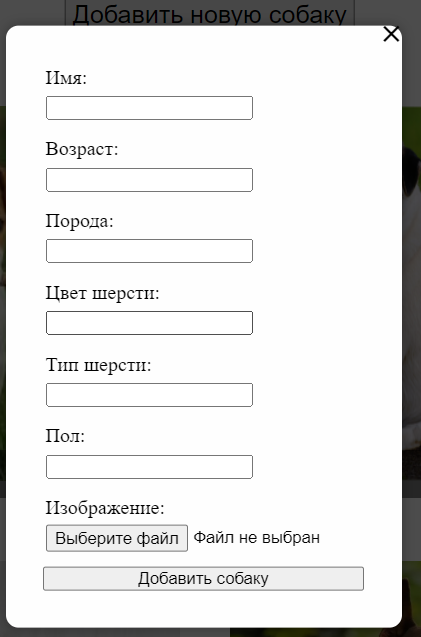
*Рис. 3. Авторизация*



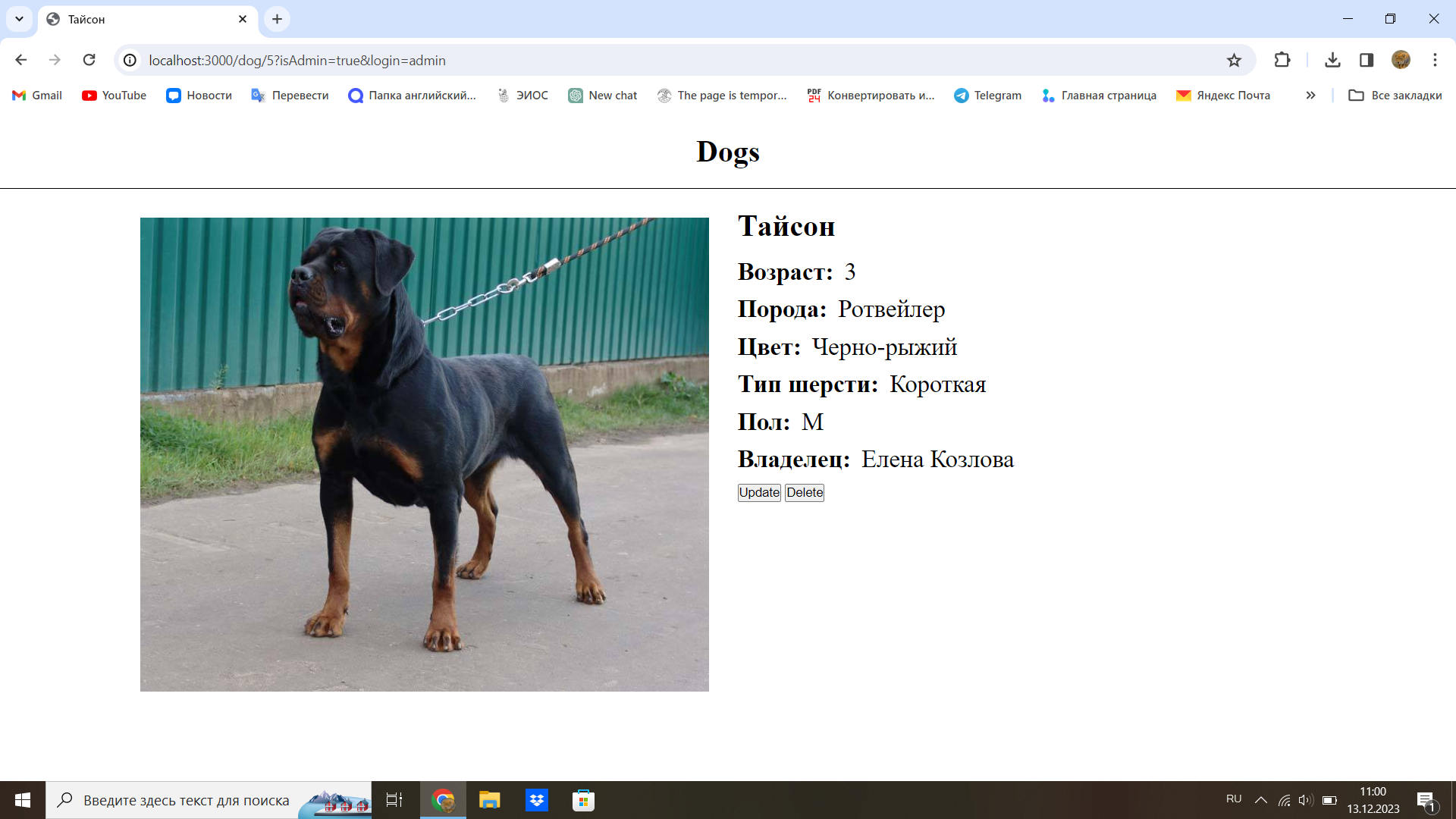
*Рис.4. Главная страница*

Справа вверху можно зарегистрироваться или авторизоваться. Кнопка добавить новую собаку доступна только админу.

Кликнув на собаку можно увидеть подробную информацию о ней.

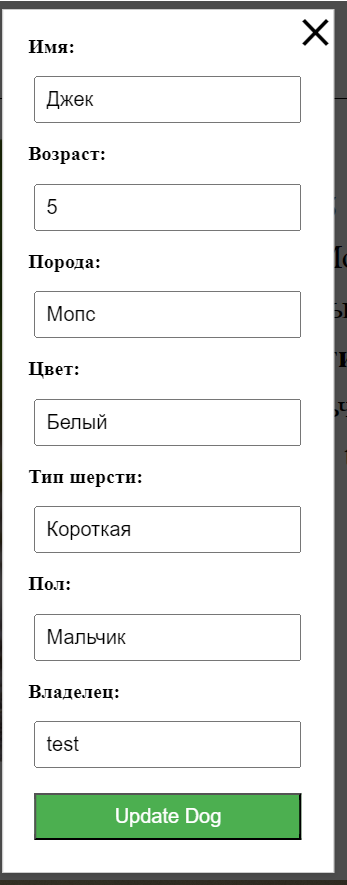
**

*Рис. 5. Добавление новой собаки*



*Рис. 6. Подробное описание*

Кнопки удалить и редактировать доступны только для админа, для не зарегистрированного пользователя их не будет видно, а для обычного пользователя будет доступна кнопка купить. При нажатии на кнопку удалить, будет выведено окно подтверждения.



*Рис. 7. Обновить данные собаки*

**Триггерная функция**

CREATE OR REPLACE FUNCTION on\_insert\_applications()

RETURNS TRIGGER AS $$

DECLARE

new\_owner VARCHAR(100);

BEGIN

IF (SELECT owner FROM Animals WHERE animal\_id = NEW.animal\_id) IS NULL THEN

SELECT name INTO new\_owner FROM Clients WHERE client\_id = NEW.client\_id;

UPDATE Animals SET owner = new\_owner WHERE animal\_id = NEW.animal\_id;

RETURN NEW;

ELSE

RAISE EXCEPTION 'Животное уже куплено';

END IF;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER after\_insert\_applications

AFTER INSERT ON Applications

FOR EACH ROW

EXECUTE FUNCTION on\_insert\_applications();

Если в таблице *animals* владелец уже есть, то триггер запретит создавать новую заявку.

**Пример функции**

app.delete('/deleteDog/:id', async (req, res) => {

  try {

      const dogId = req.params.id;

      const imagePathResult = await db.oneOrNone('SELECT image\_path FROM Animals WHERE animal\_id = $1', dogId);

      const imagePath = imagePathResult && imagePathResult.image\_path;

      await db.none('DELETE FROM Animals WHERE animal\_id = $1', dogId);

      if (imagePath) {

          const imagePathOnDisk = path.join(\_\_dirname, 'static', imagePath);

          fs.unlinkSync(imagePathOnDisk);

      }

      res.json({ message: 'Dog deleted successfully' });

  } catch (error) {

      console.error('Error deleting dog:', error);

      res.sendStatus(500);

  }

});

Функция удаления. Запросы в базу данных асинхронны, обернуты в *try catch* блок. Из *URL* адреса берётся id животного, делается запрос из базы данных для получения пути до картинки. Делается запрос на удаление животного из базы данных. По пути удаляется сам файл с картинкой.