МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ

по дисциплине «Введение нереляционные базы данных»

Тема: ИС продажи авиа- и жд-билетов (Neo4j)

Студентка гр. 7381 _____ Кревчик А.Б.

Студент гр.7381 _____ Кортев Ю.

Студентка гр.7381 _____ Процветкина А.В.

Преподаватель _____ Заславский М.М.

Санкт-Петербург

2020

ЗАДАНИЕ
НА ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ

Студентка Кревчик А.Б., группа 7381
Студент Кортев Ю., группа 7381
Студентка Процветкина А.В., группа 7381

Тема работы: ИС продажи авиа- и жд-билетов (Neo4j).

Исходные данные:
Создание приложения, в функциональность которого входят поиск покупка билетов на рейсы, добавление новых рейсов, вывод статистики, массовый импорт/экспорт.

Содержание пояснительной записки:
«Содержание»
«Введение»
«Сценарий использования»
«Модель данных»
«Разработка приложения»
«Вывод»
«Приложение»

Предполагаемый объем пояснительной записки:
Не менее 25 страниц.
Дата выдачи задания: 18.09.2020
Дата сдачи реферата:
Дата защиты реферата:

Студентка гр. 7381	_____	Кревчик А.Б.
Студент гр.7381	_____	Кортев Ю.
Студентка гр.7381	_____	Процветкина А.В.
Преподаватель	_____	Заславский М.М.

Содержание

1. ВВЕДЕНИЕ

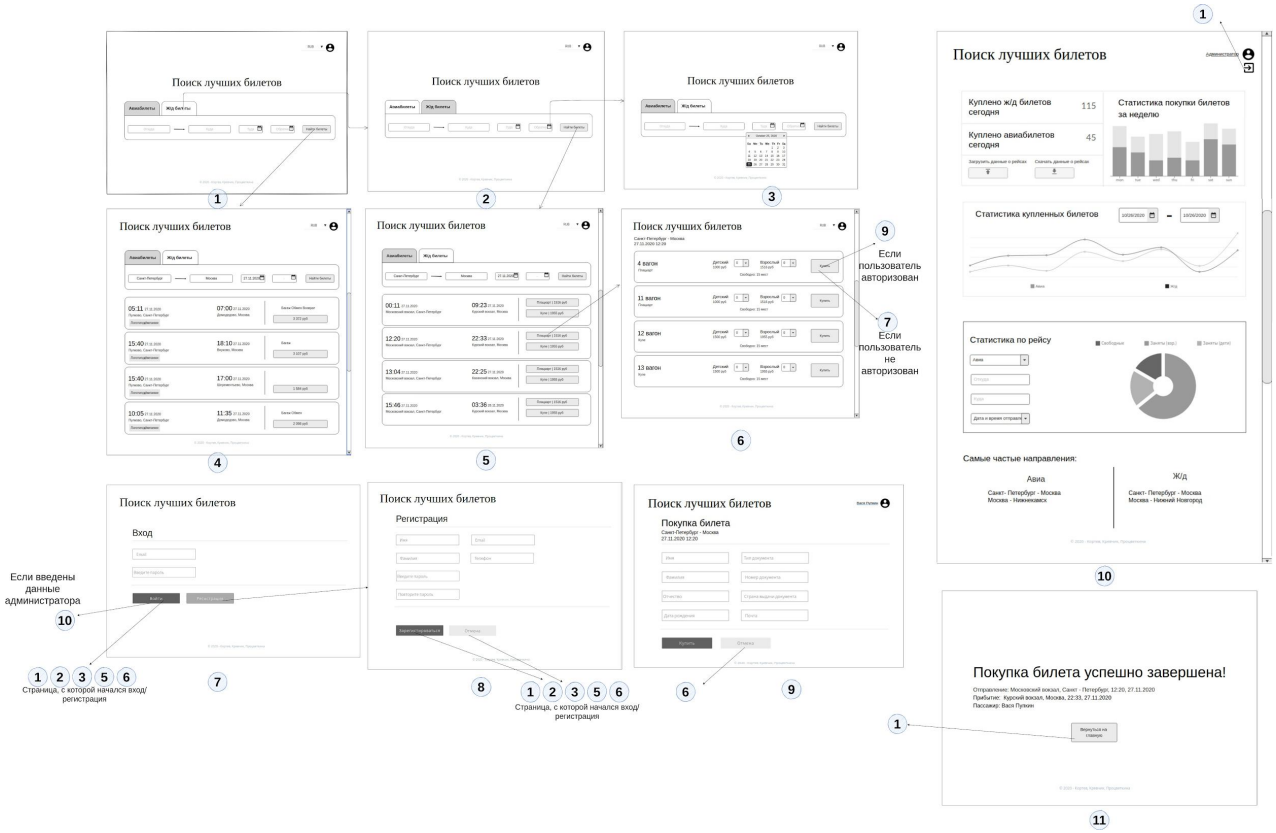
Цель. работы является создание приложения, в функциональность которого входят поиск покупка билетов на рейсы, добавление новых рейсов, вывод статистики, массовый импорт/экспорт.

2. КАЧЕСТВЕННЫЕ ТРЕБОВАНИЯ К РЕШЕНИЮ

Требуется разработать приложение, в функциональность которого будут входить: страница с поиском билетов, на которой имеется возможность выбрать дату, пункты отправления и назначения. Для администратора существует возможность вывода статистики, загрузку и выгрузка данных о рейсах. В качестве системы управления базами данных использовать Neo4j.

3.СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

a. Макет UI.



b. Сценарии использования.

Сценарий использования - “Покупка билета на поезд”

Действующее лицо: Пользователь

Основной сценарий:

1. Пользователь заходит в приложение
2. Нажимает на вкладку “Ж/д билеты”

3. Вводит пункты отправления и прибытия, выбирает дату.
4. Попадает на страницу с билетами на данную дату и время. Выбирает время отправления, рядом видит примерные цены на билеты. Нажимает на кнопку с надписью “Плацкарт”.
5. Переходит на выбор билетов. Выбирает вагон и количество пассажиров определенной категории.
6. Т.к. пользователь не вошел на сайт, он попадает на окно входа.
7. Находясь на сайте впервые, пользователь нажимает на кнопку “Регистрация”.
8. Вводит нужные для регистрации данные.
9. Попадает на страницу, с которой он начинал покупку билетов (страница с выбором вагона).
10. Опять нажимает “Купить”, т.к. он зарегистрирован и вошел в приложение, попадает на страницу с заполнением данных пассажира. Нажимает купить.
11. Открывается сообщение об успешной покупке.
12. Пользователь нажимает на кнопку возврата на главную страницу.

Альтернативный сценарий

- При выборе вагона пользователь передумал покупать билет на поезд. Нажимает кнопку “На главную”.
- В процессе записывания данных пассажира пользователь передумал и вернулся обратно для выбора другого билета, нажав кнопку “Отмена”.

Сценарий использования - “Просмотр статистики администратором”:

Действующее лицо: Администратор

Основной сценарий:

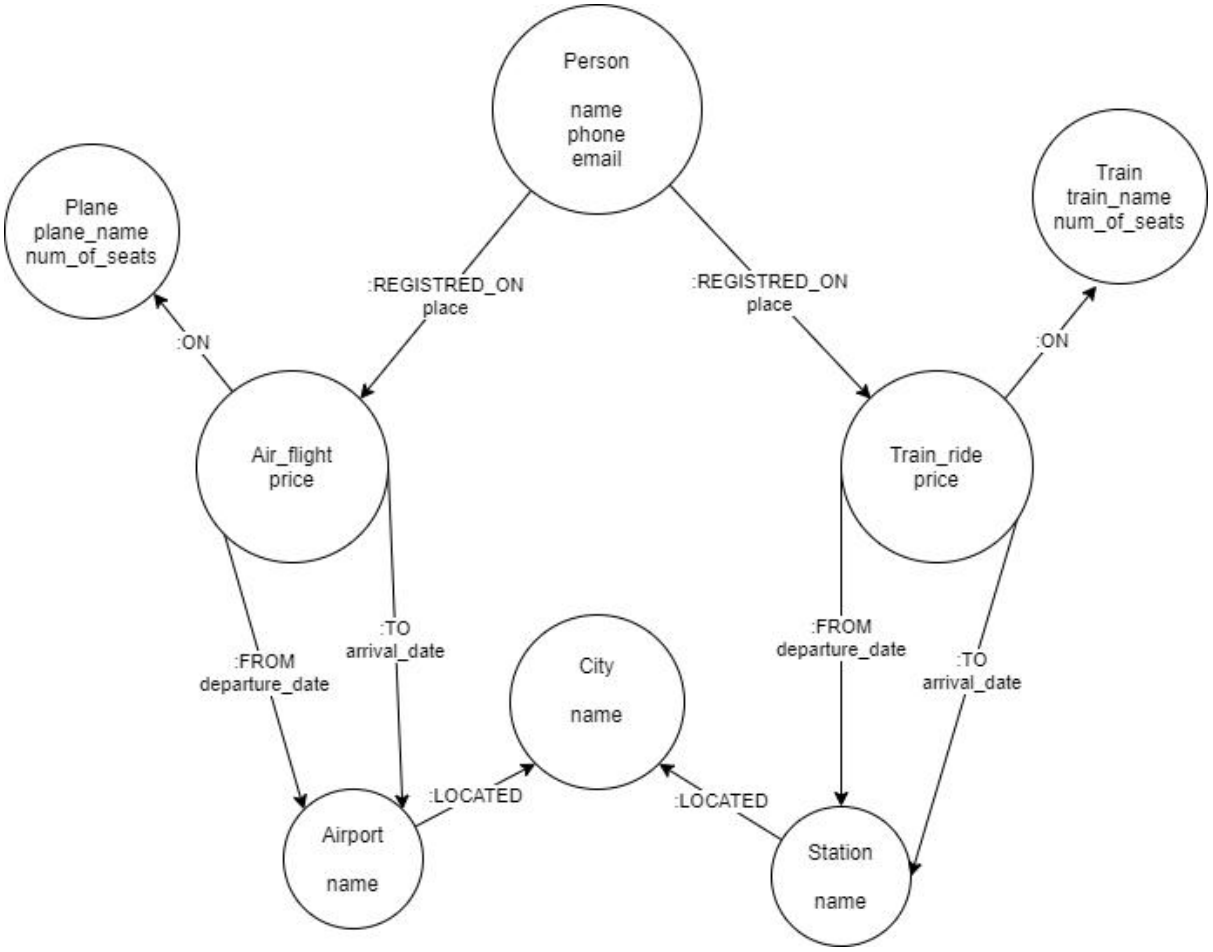
1. Администратор нажимает значок пользователя в правом верхнем углу
2. Попадает на страницу входа, вводит данные администратора
3. Переход на страницу администратора
4. Для просмотра определенного рейса выбирается категория транспорта (авиа- или ж/д), город отправления и прибытия, дата и время отправления
5. Появляется диаграмма со статистикой количества занятых и свободных мест

Альтернативный сценарий:

- Администратор добавляет сведения о новых рейсах, нажав на кнопку загрузки данных о рейсах. Данный находятся в csv файле
- Администратор просматривает статистику купленных билетов за определенный период. Для этого выбирает границы периода и видит график: одна кривая соответствует авиарейсам, вторая - ж/д.
- Администратор передумал заходить и на странице входа нажимает кнопку возврата на главную.

4.МОДЕЛЬ ДАННЫХ

4.1 Схема базы данных (графическое представление нереляционной модели данных)



4.2 Список сущностей модели

Разработанная модель данных включает следующие коллекции: Plane, Air_flight, Airport, City, Station, Train_ride, Train, Person.

4.3 Описание назначения коллекций, типов данных и сущностей.

1.Person.

Характеризует пользователя сервиса.

Свойства:

●name - имя пользователя;

●phone - телефон пользователя;

●email - телефон пользователя;

2.Air_flight.

Характеризует авиа рейс.

Свойства:

●Price - стоимость рейса;

3.Train_ride.

Характеризует поездку на поезде.

Свойства:

●Price - стоимость поездки;

4.Plain.

Характеризует самолет.

Свойства:

●Plane_name - название самолета;

●Num_of_seats - количество мест;

5.Train.

Характеризует поезд.

Свойства:

●train_name - название поезда;

●Num_of_seats - количество мест;

6.Airport.

Характеризует аэропорт.

Свойства:

●Name - название аэропорта;

7.Station.

Характеризует станцию.

Свойства:

●Name - название станции;

8.City.

Характеризует город.

Свойства:

●Name - название города;

4.4 Оценка удельного объема информации, хранимой в модели.

1.Person.

●Id - int = 4b

●Name - string - 2b * 50 = 100b

●Phone - string - 2b * 20 = 40b

●Email - string - 2b * 50 - 100b

Суммарно данная сущность занимает 244b

2.Plane.

●Id - int = 4b

●Plane_name - string - 2b*20=40b

●Num_of_seats - int =4b

Суммарно данная сущность занимает 48b

3.Train = Plane = 48b

4.Air_flight.

●Id - int = 4b

●Price - int = 4b

Суммарно данная сущность занимает 8b

5.Train_ride = Air_flight = 8b

6.City.

●Id - int = 4b

●Name - string - 2b*50 = 100b

Суммарно данная сущность занимает 104b

7.Airport.

●Id - int = 4b

●Name - string = 2b*50 = 100b

Суммарно данная сущность занимает 104b

8.Station = Airport = 104b

Также рассчитаем память, которая потребуется для хранения связей.

1.:On.

●Id - int = 4b

Sum = 4b

2.:Registed_on.

●Id - int = 4b

●Place - string = 2b*10 = 20b

Sum = 24b

3.:From.

●Id - int = 4b

●Departue_date - datetime = 15*4b = 60b

Sum = 64b

4.:To = :From = 64b

5.:Located.

●Id - int = 4b

Sum = 4b

Каждый авиарейс и поездка на поезде обязательно имеет :From, :To и [:On]->(Plane | Train)

Получается V_air_flight(N)=V_train_ride(N)=(8b+64b*2+4b)*N=140b * N

Каждый аэропорт и станция обязательно имеет :Located V_airport(N)=V_station(N)=(104b+4b)*N=108b*N

Пусть имеется A городов, B пользователей, C авиарейсов, D поездок на поезде, E аэропортов, F станций, G покупок билетов, H видов самолетов и I видов поездов, получается чистый объем:

(A*104b* + *B*244b + (C + D)*140b + (E + F)*108b + G * 20b + (H + I)*48b)

Пусть A = 1 500 000, C = 200 000, D = 500 000, E = 44 123, F = 100 000, G = 210 000 000, H = 500, I = 1 000

Тогда чистый объем от количества пользователей:

B*244b + 4.16 gigabytes

4.5. Запросы к модели, с помощью которых реализуются сценарии использования.

Добавить авиарейс из заданных аэропортов.

```
match (st1:Airport{name:'St_1'})
match (st2:Airport{name:'St_4'})
create (r:Air_flight{price:200})
create(r)-[:_FROM{diparture_time:datetime("2019-06-01")}]->(st1)
create (r)-[:_TO{arrival_time:datetime("2019-06-01")}]->(st2)
create (r)-[:_ON]->(:Plane{name:'aerobus_1', num_of_seats:300})
```

Зарегистрировать пользователя на рейс из города А в Б

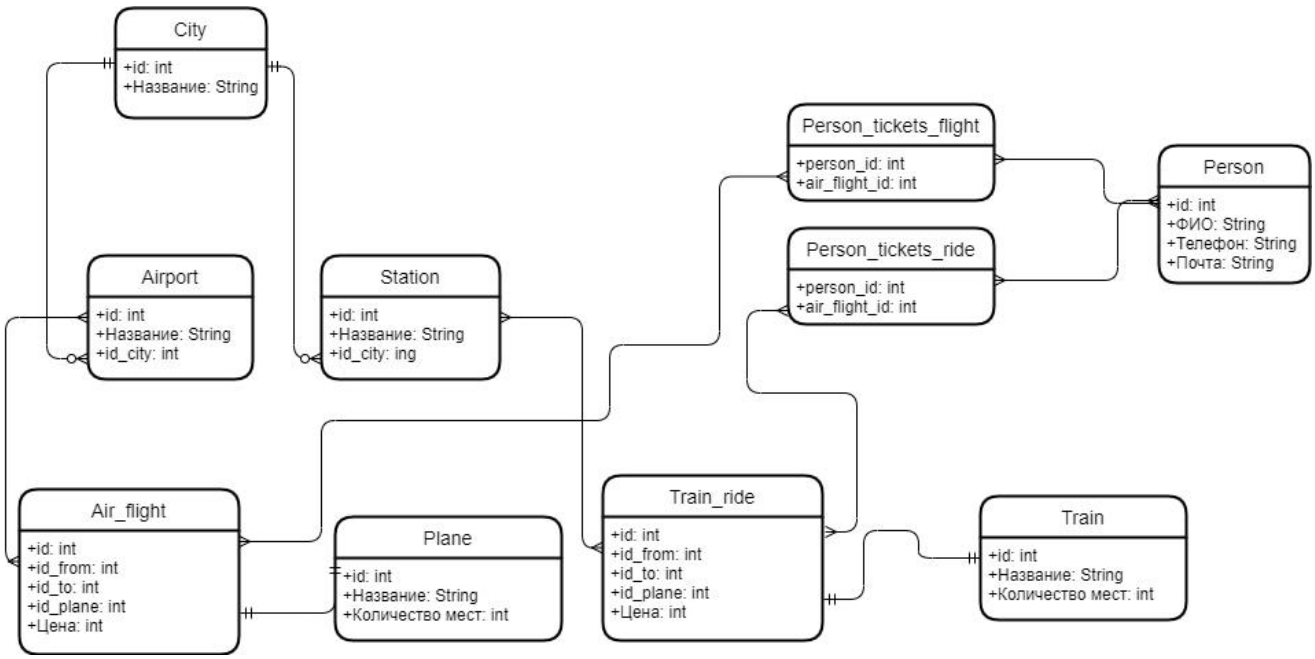
```
match (a1:Airport)-[:LOCATED]->(:City{name:'A'})
match (a2:Airport)-[:LOCATED]->(:City{name:'B'})
match (a1)<-[:`_FROM`]-(af:Air_flight)-[:`_TO`]->(a2)
create (p:Person{phone:'43252'})-[:REGISTRED_ON]->(af)
```

Вывести количество оставшихся свободных мест на авиа рейсе

```
match (p:Person)-[:REGISTRED_ON]->(a:Air_flight) where ID(a)=8
match (a)-[:`_ON`]->(plane:Plane)
return plane.num_of_seats-count(p)
```

4.6. Аналог модели данных для SQL СУБД

a. Графическое представление:



b. Описание назначений коллекций, типов данных и сущностей.

Таблица City:

Содержит информацию о городах.

- Id - уникальный идентификатор города. Тип - int. 4b
- Name - название города. Тип - String. 2b*50=100b

Таблица Airport:

Содержит информацию об аэропортах.

- Id - уникальный идентификатор аэропора. Тип - int. 4b
- Name - название аэропорта. Тип - String. 2b*50=100b
- Id_city - уникальный идентификатор города, к которому принадлежит аэропорт. Тип - int. 4b

Таблица Station:

Содержит информацию о жд станциях.

- Id - уникальный идентификатор станции. Тип - int. 4b
- Name - название станции. Тип - String. 2b*50 = 100b
- Id_city - уникальный идентификатор города, к которому принадлежит станция. Тип - int. 4b

Таблица Plane:

- Id - уникальный идентификатор аэропора. Тип - int. 4b
- Name - название самолета. Тип - String. 2b*20 = 40b
- Num_of_seats - количество мест в самолете. Тип - int. 4b

Таблица Train:

- Id - уникальный идентификатор аэропора. Тип - int. 4b
- Name - название поезда. 2b*20 = 40b
- Num_of_seats - количество мест в поезде Тип - int. 4b

Таблица Air_flight:

- Id - уникальный идентификатор авиа рейса. Тип - int. 4b
- Id_to - уникальный идентификатор аэропора прибытия. Тип - int. 4b
- Id_from - уникальный идентификатор аэропора отправления. Тип - int. 4b
- Id_plane - уникальный идентификатор самолета. Тип - int. 4b

- Price - цена за рейс. Тип - int. 4b

Таблица Train_ride:

- Id - уникальный идентификатор поездки на поезде. Тип - int. 4b
- Id_to - уникальный идентификатор аэропорта прибытия. Тип - int. 4b
- Id_from - уникальный идентификатор аэропорта отправления. Тип - int. 4b
- Id_train - уникальный идентификатор поезда. Тип - int. 4b
- Price - цена за поездку. Тип - int. 4b

Таблица Person:

Содержит информацию о пользователях.

- Id - уникальный идентификатор аэропора. Тип - int. 4b
- Name - имя пользователя. Тип - String. 2b*50 = 100b
- Phone - телефон пользователя. Тип - String. 2b*20 = 40b
- Email - почта пользователя. Тип - String. 2b*50 = 100b

Таблица Person_tickets_flight:

Содержит информацию о купленных пользователями билетах на авиа рейсы.

- Person_id - уникальный идентификатор пользователя. Тип - int. 4b
- Air_flight_id - уникальный идентификатор оплаченного рейса. Тип - int. 4b

Таблица Person_tickets_ride:

Содержит информацию о купленных пользователями билетах на поезда.

- Person_id - уникальный идентификатор пользователя. Тип - int. 4b

- Train_ride_id - уникальный идентификатор поездки на поезде. Тип - int. 4b

c. Оценка удельного объема информации, хранимой в модели.

Пусть имеется A городов, B пользователей, C авиарейсов, D поездок на поезде, E аэропортов, F станций, G покупок билетов, H самолетов, I поездов, получается чистый объем:

(A*104b* + *B*244b + (C + D)*20b + (E + F)*108b + G * 8b + (H+I)*48b)

Пусть A = 1 500 000, C = 200 000, D = 500 000, E = 44 123, F = 100 000, G = 210 000 000, H = 500, I = 1 000

Тогда чистый объем от количества пользователей:

B*244b + 4.08 gigabytes

d. Запросы к модели, с помощью которых реализуются сценарии использования.

Получить id рейсов из города А в Б.

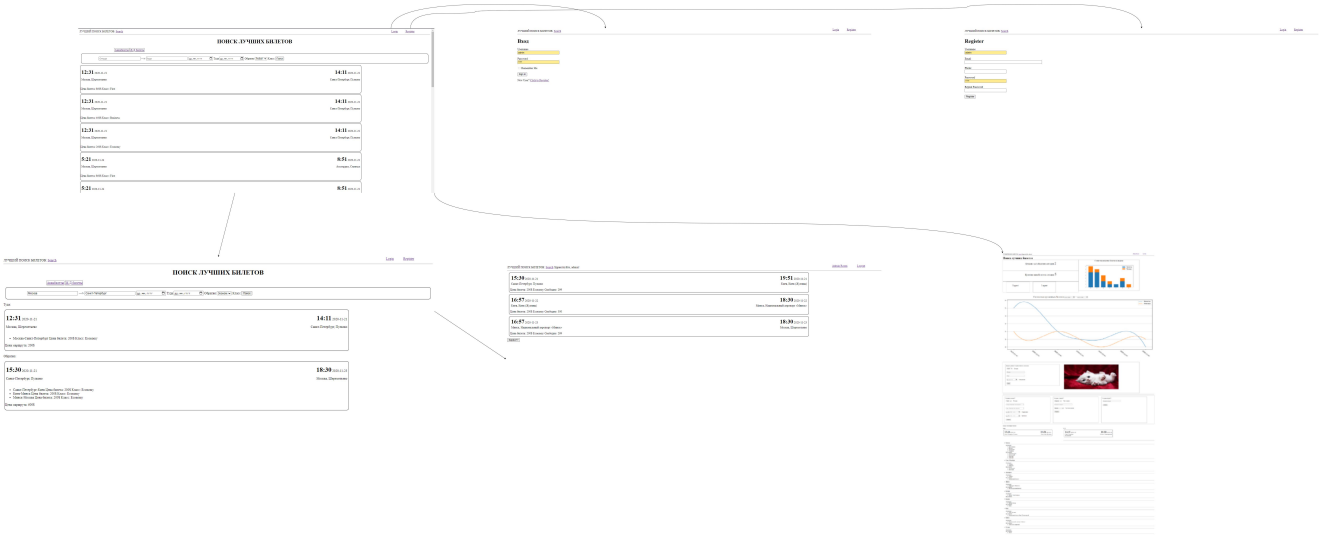
```
Select af.id from Air_flight af
Left join Airport from On from.id = id_from
Left Join Airport to On to.id = id_to
Left join City fc On from.id_city = fc.id
Left join City tc On to.id_city = tc.id
Where fc.name = ‘Saint-Petersburg’ and tc.name = ‘Moscow’;
```

Зарегистрировать пользователя на рейс.

```
Insert into Person_tickets_flight
Values(flight_id,person_id)
```

6. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

Схема экранов приложения



Использованные технологии

БД: Neo4j

Back-end: Python, Flask

Front-end: HTML, CSS, JavaScript

6.ВЫВОДЫ

Результаты

В ходе выполнения задания было реализовано приложение, позволяющее осуществлять покупку и продажу авиа- и жд-билетов. Имеется возможность массового импорта и экспорта данных о рейсах.

Недостатки и пути для улучшения полученного решения

Решение может быть улучшено, внедрением в приложение современных frontend фреймворков, добавлением выбора места пассажира по карте расположения мест в транспорте.

7. Приложения

Документация по сборке и развертыванию приложения

1. Клонировать репозиторий.
2. Перейти в папку репозитория
3. Вызвать docker-compose up --build