Энскгортранс

# Описание

Веб-приложение, обладающее следующим функционалом: отправка и составление запросов к REST API, вывод ответа на запрос на страницу в качестве Json файла.

## Наименование

Энскгортранс (сокращенно ЭГТ)

## Предметная область

Разработка прикладного ПО

# Данные

В приложении будут использоваться следующие модели данных:

**Model** — модель транспортного средства. Модель включает, прежде всего, название типа ТС (автобус, троллейбус, трамвай, ТУАХ, электробус), название модели, определяемое производителем («ПКТС-6281 «Адмирал»»), вместимость в количестве пассажиров.

id SERIAL PRIMARY KEY,  
type Vehicle NOT NULL,  
name TEXT NOT NULL,  
capacity INT NOT NULL CHECK ( capacity > 0 )

Vehicle AS ENUM ('Автобус', 'Троллейбус', 'Трамвай', 'ТУАХ', 'Электробус');

**Transport** — тип транспортного средства. У каждого транспортного средства (ТС) есть бортовой номер-идентификатор, год его выпуска, актуальное состояние -- значение из конечного множества «исправен», «некритические неисправности», «требует ремонта» -- и модель ТС. Типы транспортного средства перечислены в Vehicle.

id INT PRIMARY KEY,  
year INT CHECK ( year > 0 ),  
condition Condition NOT NULL,  
model\_id INT NOT NULL REFERENCES Model (id)

Condition AS ENUM ('Требует ремонта', 'Некритические неисправности', 'Исправен');

**STOP** — множество остановок ОТ. У каждой остановки есть персональный номер и адрес, записываемый в довольно произвольном виде (например, «перекрёсток улиц Ленина и Николая Второго») и количество платформ - мест для размещения одного ТС. Платформы одной остановки пронумерованы начиная с 1.

id INT PRIMARY KEY,  
address TEXT NOT NULL,  
platform\_count INT NOT NULL CHECK ( platform\_count > 0 ),  
  
UNIQUE(address)

**Route -** маршруты ТС(транспортного средства). У маршрута есть уникальный номер, известный пассажирам, тип ТС, который его обслуживает, остановка, условно называемая начальной и условная конечная остановка. В реальности ТС ходят по маршруту туда-сюда и вполне могут двигаться в обратном направлении, от «конечной» остановки к «начальной».

id INT PRIMARY KEY,  
transport\_type Vehicle NOT NULL,  
first\_stop\_id INT NOT NULL REFERENCES Stop (id),  
last\_stop\_id INT NOT NULL REFERENCES Stop (id)

**Timetable** — расписания работы ТС. В расписании написано с точностью до минуты, в какой момент времени ТС какого маршрута должен прибыть на ту или иную остановку и к какой платформе должен подъехать. ТС стоит у платформы одну минуту, и разумеется никакое другое ТС в это время у этой платформы стоять не может. Для рабочих и выходных дней расписания одного маршрута могут быть разные, но больше дни никак не различаются.

id SERIAL PRIMARY KEY,  
route\_id INT NOT NULL REFERENCES Route (id),  
stop\_id INT NOT NULL REFERENCES Stop (id),  
platform INT NOT NULL,  
arrival\_time TIME NOT NULL,  
weekend BOOLEAN NOT NULL,  
  
UNIQUE (stop\_id, route\_id, arrival\_time, platform, weekend),  
CHECK ( platform > 0 AND platform <= *GetPlatformCount*(stop\_id) )

**DialyRoute** — наряды на работу. Это задание какому-то конкретному ТС следовать в этот день по заданному маршруту, начиная с заданной остановки в указанное время.

id SERIAL PRIMARY KEY,  
day DATE NOT NULL,  
transport\_id INT NOT NULL REFERENCES Transport (id),  
route\_id INT NOT NULL REFERENCES Route (id),  
departure\_time TIME NOT NULL,  
driver\_licence INT NOT NULL REFERENCES Driver (licence),  
  
UNIQUE (day, driver\_licence),  
UNIQUE (day, transport\_id)

**Driver** — модель водителя транспортного средства, выполняющего наряд.

licence INT PRIMARY KEY,  
name TEXT NOT NULL,  
surname TEXT NOT NULL,  
patronymic TEXT

**Logbook -** модель логов записи с GPS.(диспетчерская следит за выполнением наряда при помощи GPS и записывает, в какое время исполнитель того или иного наряда действительно прибыл на ту или иную остановку.)

daily\_route\_id INT NOT NULL REFERENCES DailyRoute (id),  
stop\_id INT NOT NULL REFERENCES Stop (id),  
arrival\_time TIME NOT NULL,  
  
UNIQUE (daily\_route\_id, arrival\_time)

**Ticket -**  модель билетного меню. В каждом пункте меню записано его название («пересадочный билет на 90 минут», «беспересадочный билет на 15 минут», etc.) и стоимость.

id SERIAL PRIMARY KEY,  
name TEXT NOT NULL,  
cost INT NOT NULL CHECK ( cost > 0 ),

UNIQUE(name, cost)

**Booking -** модель продажи билетов на маршрут (наряд).Пассажиры, заходя в салон ТС валидируют свои билеты. Информация о валидациях обрабатывается в онлайн-режиме некой сторонней системой, а в конце работы каждого наряда записывается, сколько валидаций каждого типа билетов было сделано за время работы. Потом эта статистика используется для анализа пассажиропотоков и доходов.

id SERIAL PRIMARY KEY,  
ticket\_id INT NOT NULL REFERENCES Ticket (id),  
daily\_route\_id INT NOT NULL REFERENCES DailyRoute (id),  
daily\_validation\_count INT NOT NULL,  
  
UNIQUE(daily\_route\_id, ticket\_id)

**FeedBack —** модель фидбэка по маршрутам, которую потом анализирует умный помощник «Аннушка».

id SERIAL PRIMARY KEY,  
user\_id BYTEA NOT NULL REFERENCES AppUser (id),  
stop\_id INT REFERENCES Stop (id),  
route\_id INT REFERENCES Route (id),  
backend\_id INT NOT NULL REFERENCES Backend (id),  
features\_map JSONB NOT NULL,  
datetime\_feedback TIMESTAMP NOT NULL,  
  
CONSTRAINT chk\_target CHECK (stop\_id IS NOT NULL OR route\_id IS NOT NULL)

**Backend —** модель бекендов. Эти бекенды выдают рекомендации по улучшению сервиса. От них нас интересует только их название и краткое описание.

id SERIAL PRIMARY KEY,  
name TEXT NOT NULL,  
description TEXT NOT NULL

**AppUser —** модель пользователей приложения. Аутентификация пользователей происходит на стороннем сервисе, от которого вы получаете только анонимизированный 8-байтовый уникальный номер пользователя, дату его регистрации в Аннушке и номер версии мобильного приложения.

id BYTEA PRIMARY KEY,  
datetime\_registration TIMESTAMP NOT NULL,  
app\_version TEXT NOT NULL

**POI** - каталог POI (Point of Interest), где записан адрес точки, её тип (это собор, бассейн, библиотека или что-то ещё), название и URL фотографии.

CREATE TABLE POIType   
(  
 type TEXT PRIMARY KEY  
);

name TEXT PRIMARY KEY,  
photo\_url TEXT,  
poi\_type TEXT NOT NULL REFERENCES POIType,  
address TEXT NOT NULL

**POIPath -** список ближайших остановок ОТ, с указанием, сколько минут идти пешком от остановки до POI, есть ли на пути преграды, создающие помехи маломобильным пассажирам, и сколько раз на пути нужно перейти через автомобильную дорогу.

id SERIAL PRIMARY KEY,  
poi\_name TEXT NOT NULL REFERENCES POI,  
stop\_id INT NOT NULL REFERENCES Stop,  
route\_time INT NOT NULL,  
has\_obstacles BOOLEAN NOT NULL,  
road\_crosses INT NOT NULL,  
  
UNIQUE(poi\_name, stop\_id)

**POIPathStats -** ежедневную статистика (собираемая сторонними сервисами) о том, сколько человек воспользовалось остановкой, чтоб добраться до точки.

id SERIAL PRIMARY KEY,  
poi\_path\_id INT NOT NULL REFERENCES POIPath,  
date DATE NOT NULL,  
use\_count INT NOT NULL,  
  
UNIQUE(poi\_path\_id, date)

# Пользовательские роли

Существует единственная роль – гостевая .

API приложения  
В данном проекте реализован следующий API:

* API метод, возвращающий JSON с основными свойствами маршрутов.

**Путь**:/route/info

**Аргументы** route: number -- номер маршрута, опциональный.

**Пример запроса** /route/info?route=31.

**Структура ответа**: Если в запросе указан аргумент route, то в ответе должна быть информация только об указанном маршруте

* API метод, возвращающий JSON с информацией о расписании движения ТС по данному маршруту.

**Путь**: /schedule/info

**Аргументы** route: number -- номер маршрута, обязательный. weekend: boolean-- флаг, указывающий на то, требуем ли мы расписание выходного дня или буднего.

**Пример запроса:** /schedule/info?route=31&weekend=true

**Структура ответа:** в ответе должно быть время начала первого и последнего рейсов, и последовательность остановок на маршруте, соответствующая первому рейсу, в порядке от начальной до конечной остановки.

* API метод, возвращающий JSON с информацией о суммарном количестве валидаций билетов каждого типа из билетного меню и суммарной их стоимости. Код берет информацию из представления, не производя никаких действий, кроме перекладывания данных из одной структуры в другие.(место нахождения: src/main/sql/task7.sql)

**Путь**: /fares/stats.

* API метод, возвращающий JSON с информацией о суммарном количестве смен, отработанных каждым водителем. Код берет информацию из представления, не производя никаких действий, кроме перекладывания данных из одной структуры в другие.(место нахождения: src/main/sql/task8.sql)

**Путь**: /staff/workload

**Аргументы** id: text | number: -- номер служебного удостоверения водителя, опциональный.

**Структура ответа**: если номер удостоверения указан, выводится информация только об указанном водителе. Иначе выводится информация обо всех водителях.

## Язык программирования

В проекте использовались следующие языки программирования и технологии: Python, ORM фреймворк peewee, веб-фреймворк cherrypy. Архитектура приложения: клиент-сервер.

## СУБД

PostrgeSQL