

СПбПУ

**Пояснительная записка  
к курсовой работе  
по предмету  
“Базы данных”**

2023г.

## Оглавление

Задание – Автоматизация работы автосервиса.....	3
Схема БД.....	4
Реализация курсового проекта.....	6
Общие слова по архитектуре проекта.....	6
Реализация системы пользователей и ролей.....	7
Разделение прав доступа.....	8
Реализация остальных функциональных требований.....	9
Запуск проекта, триггеры, хранимые процедуры.....	9
Основные экраны.....	13
Справочники.....	13
Отчёты.....	14
Реализация инфографики, требуемой в задаче.....	14
Реализация инфографики, требуемой в задаче.....	16
Реализация главного окна оператора.....	17
Добавить новую работу.....	18
Посмотреть занятость мастера.....	19
Выводы.....	20
Литература.....	21

# Задание – Автоматизация работы автосервиса

В рамках данного цикла лабораторных работ необходимо автоматизировать работу автосервиса. Для этого в рамках базы данных PostgreSQL необходимо создать объекты в схеме вашего пользователя (логин и пароль пользователя для доступа к базе данных студент должен получить у преподавателя) и написать клиентское приложение на базе компонентов ADO.NET или JDBC.

Автосервис специализируется на ремонте и техническом обслуживании автомобилей. При этом цена услуг для отечественных и зарубежных отличается. В сервисе есть ограниченный круг клиентов для которых производятся работы. Каждый автомобиль клиента характеризуется государственным номером, цветом и маркой. Периодически для автомобиля мастер выполняет работу, данные о которой заносятся в журнал работ. Одну услугу оказывает один мастер.

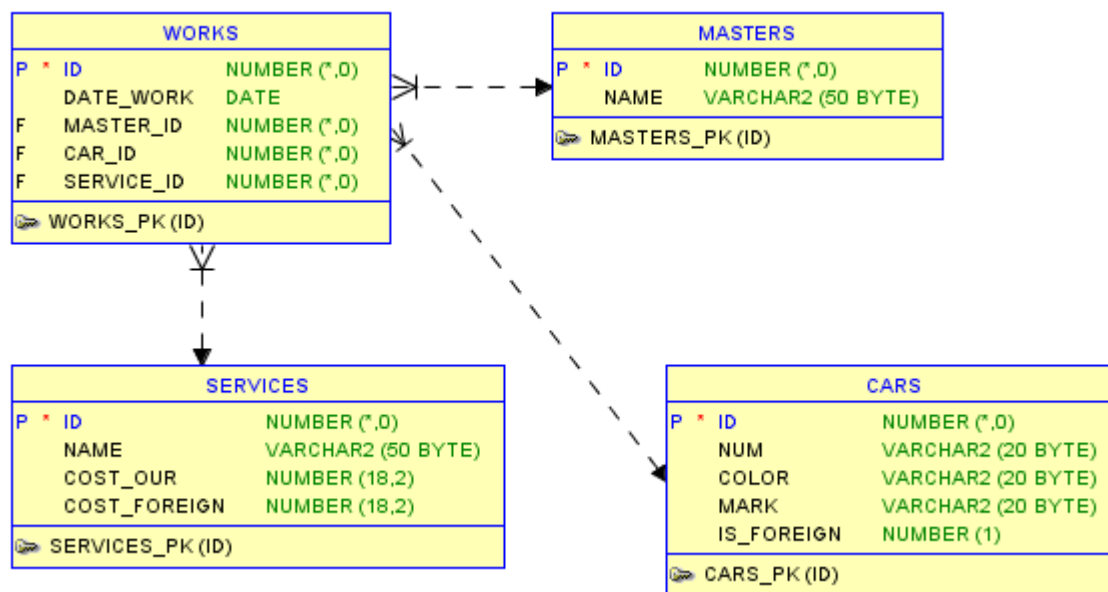
## **База данных должна удовлетворять следующим требованиям:**

1. Контроль целостности данных, используя механизм связей
2. Операции модификации групп данных и данных в связанных таблицах должны быть выполнены в рамках транзакций.
3. Логика работы приложения должна контролироваться триггерами. В частности:
  - Не позволяет добавить автомобиль с уже существующим номером
  - Не позволяет добавить мастера, если их уже больше 10
  - Не позволяет дать работу мастеру, если он в этот день уже выполнил больше одной работы
4. Все операции вычисления различных показателей (из требований к клиентскому приложению) должны реализовываться хранимыми процедурами.

## **Требования к клиентскому приложению:**

1. Необходимо реализовать интерфейсы для ввода, модификации и удаления справочников:
  - Мастеров;
  - Автомобилей;
  - Услуг.
2. В главном окне приложения должен быть реализован интерфейс, позволяющий оператору назначать работы из перечня услуг мастерам автосервиса для обслуживания имеющейся базы клиентов с возможностью указания даты проведения работы.
3. Необходимо реализовать возможность просмотра оператором следующих показателей:
  - Общая стоимость обслуживания отечественных и импортных автомобилей (с возможностью фильтрации по датам оказания услуги).
  - Пять мастеров, которые в заданном месяце выполнили наибольшее число работ для разных автомобилей.

## Схема БД



1. Запустить pgAdmin или psql.
2. Создать соединение, используя логин и пароль.
3. Изменить пароль вашего пользователя на свой, используя команду

```
ALTER USER your_user_name IDENTIFIED BY new_password;
```

4. Создать необходимые таблицы.

Имя таблицы	Имя колонки	Расшифровка
cars		Таблица автомобилей
	id	Идентификатор записи
	num	Номер
	color	Цвет
	mark	Марка
	is_foreign	Иностранная (1) или нет (0)
masters		Таблица мастеров
	id	Идентификатор записи
	name	Имя
services		Таблица услуг
	id	Идентификатор записи
	name	Наименование
	cost_our	Стоимость для отечественной машины
	cost_foreign	Стоимость для иномарки
works		Таблица работ
	id	Идентификатор записи
	date_work	Дата работы
	master_id	Мастер
	car_id	Машина
	service_id	Услуга

5. Создать связи между таблицами:

Название	Primary Key	Foreign Key
fk_works_cars	cars.id	works.car_id
fk_works_masters	masters.id	works.master_id
fk_works_services	services.id	works.service_id

6. Создать Backup базы и запомнить место его расположения.  
7. Удалить базу с сервера  
8. Восстановить базу из Backup базы  
9. Для всех таблиц реализовать автоматическое заполнение первичного ключа при вставке данных (автоинкремент).

# Реализация курсового проекта

## Общие слова по архитектуре проекта

Проект писался с простотой запуска в уме, при этом чтобы результаты были надежными и воспроизводимыми, поэтому было принято решение создавать разворачивать решение в контейнерах. Для контейнеризации использовался Docker (version 24.0.6), для разворачивания – Docker Compose (v2.21.0).

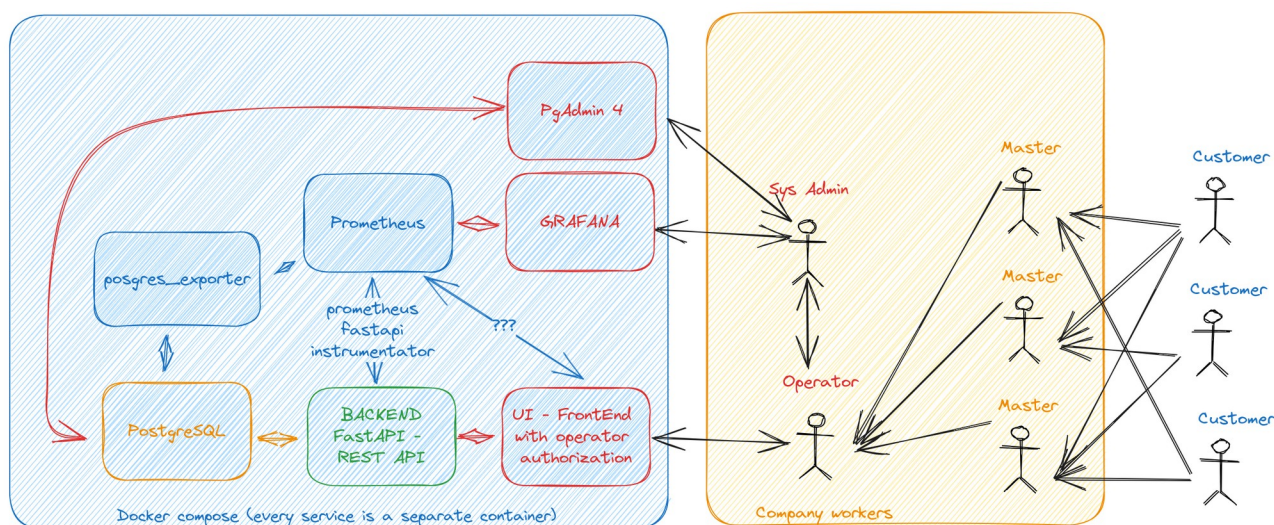
Изначальная архитектура была немного сложнее реализованной впоследствии, в силу того что потребность в отдельном Rest API сервисе оказалось значительно ниже, чем предполагалось, в связи с чем было принято решение полностью от него отказаться и общаться с помощью веб-клиента напрямую с базой данных.

Ниже представлена схема-черновик, где помимо двух основных контейнеров – PostgreSQL и UI-Frontend, который стал веб-сервисом на основе библиотеки Streamlit, имеются сервисы для мониторинга работы базы данных и остальных контейнеров.

При проектировании предполагалось, что системой в реальном бы сценарии пользовалось два человека – сисадмин, который обладает компетенциями по настройке системы и который регулирует ее работу – у него был бы доступ к сервисам мониторинга и сбора статистики – GRAFANA и PgAdmin 4, и он бы регулировал пользование сервисом посредством выдачи логинов и паролей для входа в систему, например, операторам.

Операторы в свою очередь через веб-интерфейс заносят информацию о работах мастеров, вносят актуальную информацию о них и в целом ведут учет работы.

Планировалось, что сервисом будут пользоваться в рамках одного сервисного центра (масштабирование не предполагалось), чтобы не совершать изменения в предоставленной базе данных.



## Реализация системы пользователей и ролей

Для реализации системы пользователей и ролей использовалось полуготовое решение – модуль-дополнение к библиотеке streamlit, streamlit-authenticator, который предоставляет удобную форму для аутентификации пользователей и содержал набросочные примеры системы пользователей, подгружаемых из файла. Решение подходило, поскольку в нем автоматически производилось хеширование паролей. Пример файла с информацией о пользователях представлен ниже.

config.yml:

```
credentials:
  usernames:
    sysadmin: # Notice that login is this line
      email: quakumei@gmail.com
      name: Ilya Tampio
      level: admin
      password: $2b$12$ByiI4veq94h3lRJwR6lwp0RJpml0sVb5FjmJvjgaUeZC83drqsl2q
    steven_hawking: # Sample operator
      email: steven.hawking@gmail.com
      name: Steven Hawking
      level: operator
      password: $2b$12$ByiI4veq94h3lRJwR6lwp0RJpml0sVb5FjmJvjgaUeZC83drqsl2q
  cookie:
    expiry_days: 7
    key: "dfhkgjhdhfkjghjdhkhlakjsdhlfkjwhqwqh" # Must be string
    name: car_service_authenticate
  preauthorized:
```

В разделе usernames указываются логины пользователей (логин так же доступен через email), также указывается некоторая личная информация о пользователе, а так же уровень его привилегий – level. В моем решении реализовано два уровня пользователей – admin и operator. Operator, в свою очередь, не имеет доступа к вкладке Tables (Справочники), что мы увидим немного позднее. Обращу внимание на то, что пароли захешированы по алгоритму. В свою очередь обратимся к тому, как выглядит логин, и посмотрим на разделение прав доступа.


### Логин

Username

sysadmin

Password

5130904/10002



Login

Пожалуйста, введите имя пользователя и пароль

## Разделение прав доступа

Так выглядит страница при авторизации через пользователя с уровнем доступа operator

Профиль

Вы авторизованы как *Steven Hawking* (уровень доступа: operator)

Выйти из профиля

## Справочники

У вас недостаточно прав, чтобы просматривать эту страницу

А так, с уровнем доступа admin открываются возможности к просмотру и редактированию таблиц напрямую.

Профиль

Вы авторизованы как *Ilya Tarnio* (уровень доступа: admin)

Выйти из профиля

## Справочники

Masters

Cars

Services

Works



# Реализация остальных функциональных требований

## Запуск проекта, триггеры, хранимые процедуры

Для простоты демонстрации проекта и воспроизводимости результатов, я написал один .sql скрипт, который позволяет инициализировать базу данных, с некоторыми данными для примера, а так же объявить все функции, хранимые процедуры и соответствующие триггеры.

car\_service\_initdb.sql:

```
-- Создание базы данных
CREATE DATABASE car_service;

-- Использование созданной базы данных
\c car_service;
-- Создание таблицы "cars" с автоинкрементом для id
CREATE TABLE cars (
    id serial PRIMARY KEY,
    num varchar(20) NOT NULL,
    color varchar(20),
    mark varchar(20),
    is_foreign boolean
);

-- Создание таблицы "masters" с автоинкрементом для id
CREATE TABLE masters (
    id serial PRIMARY KEY,
    name varchar(50) NOT NULL
);

CREATE TABLE services (
    id serial PRIMARY KEY,
    name varchar(50) NOT NULL,
    cost_our numeric(18, 2),
    cost_foreign numeric(18, 2)
);

CREATE TABLE works (
    id serial PRIMARY KEY,
    date_work date,
    master_id int REFERENCES masters (id),
    car_id int REFERENCES cars (id),
    service_id int REFERENCES services (id)
);

-- Создание связи fk_works_cars между таблицами cars и works
ALTER TABLE works
    ADD CONSTRAINT fk_works_cars FOREIGN KEY (car_id) REFERENCES cars (id);

-- Создание связи fk_works_masters между таблицами masters и works
ALTER TABLE works
    ADD CONSTRAINT fk_works_masters FOREIGN KEY (master_id) REFERENCES masters (id);

-- Создание связи fk_works_services между таблицами services и works
ALTER TABLE works
    ADD CONSTRAINT fk_works_services FOREIGN KEY (service_id) REFERENCES services (id);

--
-- Data for Name: cars; Type: TABLE DATA; Schema: public; Owner: tampio
--
COPY public.cars (id, num, color, mark, is_foreign) FROM stdin;
1      a905ав777      Красный Honda      t
2      6123ка717          Белый   Лада      f
3      6128кф939          Чёрный Лада      f
4      а888щл913          Фиолетовый Mazda  t
5      г123ар834          Бирюзовый   УАЗ      f
\.

--
-- Data for Name: masters; Type: TABLE DATA; Schema: public; Owner: tampio
--
COPY public.masters (id, name) FROM stdin;
```

```

1      Василий
2      Сергей
3      Юлиана
4      Анатолий
5      Пётр
\..

--
-- Data for Name: services; Type: TABLE DATA; Schema: public; Owner: tampio
--
COPY public.services (id, name, cost_our, cost_foreign) FROM stdin;
1      Мытьё машины      575.00 1725.00
2      Замена карбюратора 5750.00 8625.00
3      Смена резины      2875.00 4600.00
\..

--
-- Data for Name: works; Type: TABLE DATA; Schema: public; Owner: tampio
--
COPY public.works (id, date_work, master_id, car_id, service_id) FROM stdin;
1      2023-01-25      1      1      1
2      2023-01-26      2      2      2
3      2023-01-27      3      3      3
4      2023-01-27      1      3      2
\..

--
-- Name: cars_id_seq; Type: SEQUENCE SET; Schema: public; Owner: tampio
--
SELECT
    pg_catalog.setval('public.cars_id_seq', 5, TRUE);

--
-- Name: masters_id_seq; Type: SEQUENCE SET; Schema: public; Owner: tampio
--
SELECT
    pg_catalog.setval('public.masters_id_seq', 5, TRUE);

--
-- Name: services_id_seq; Type: SEQUENCE SET; Schema: public; Owner: tampio
--
SELECT
    pg_catalog.setval('public.services_id_seq', 3, TRUE);

--
-- Name: works_id_seq; Type: SEQUENCE SET; Schema: public; Owner: tampio
--
SELECT
    pg_catalog.setval('public.works_id_seq', 4, TRUE);

--
-- Name: cars cars_pkey; Type: CONSTRAINT; Schema: public; Owner: tampio
--
--
-- Хранимая процедура
--
CREATE OR REPLACE FUNCTION service_cost_by_foreignness (date_one text, date_two text)
    RETURNS TABLE (
        is_foreign boolean,
        service_cost numeric
    )
    AS $$
BEGIN
    RETURN QUERY
    SELECT
        cars.is_foreign,
        SUM(
            CASE WHEN cars.is_foreign THEN
                services.cost_foreign
            ELSE
                services.cost_our
            END) AS service_cost
    FROM
        works
    LEFT JOIN cars ON works.car_id = cars.id
    LEFT JOIN services ON works.service_id = services.id
WHERE

```

```

        date_work >= TO_DATE(date_one, 'YYYY-MM-DD')
        AND date_work <= TO_DATE(date_two, 'YYYY-MM-DD')
    GROUP BY
        cars.is_foreign
    ORDER BY
        service_cost DESC;
END;
$$
LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION top_five_masters (date_one text, date_two text)
    RETURNS TABLE (
        master_id int,
        master_name varchar(50),
        works_count bigint
    )
    AS $$
BEGIN
    RETURN QUERY
    SELECT
        masters.id AS master_id,
        masters.name AS master_name,
        COUNT(1) AS works_count
    FROM
        works
    LEFT JOIN masters ON works.master_id = masters.id
WHERE
    works.date_work >= TO_DATE(date_one, 'YYYY-MM-DD')
    AND works.date_work <= TO_DATE(date_two, 'YYYY-MM-DD')
    GROUP BY
        masters.id
    ORDER BY
        COUNT(1) DESC
    LIMIT 5;
END;
$$
LANGUAGE plpgsql;

--
-- TRIGGERS
--
-- Unique car nums
CREATE OR REPLACE FUNCTION enforce_unique_cars_num ()
    RETURNS TRIGGER
    AS $$
BEGIN
    IF EXISTS (
        SELECT
            1
        FROM
            cars
        WHERE
            num = NEW.num) THEN
        RAISE EXCEPTION 'Duplicate value detected for num: %', NEW.num;
    END IF;
    RETURN NEW;
END;
$$
LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER unique_num_insert_trigger BEFORE INSERT ON cars FOR EACH ROW EXECUTE
FUNCTION enforce_unique_cars_num ();

-- 10 Masters maximum
CREATE OR REPLACE FUNCTION enforce_ten_masters_cap ()
    RETURNS TRIGGER
    AS $$
BEGIN
    IF ((
        SELECT
            COUNT(*)
        FROM
            masters) > 9) THEN
        RAISE EXCEPTION 'Masters count is 10 or more, denying addition of new: master name %',
NEW.name;
    END IF;
    RETURN NEW;

```

```

END;
$$
LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER ten_masters_cap BEFORE INSERT ON masters FOR EACH ROW EXECUTE FUNCTION
enforce_ten_masters_cap ();

-- Prohibition of addition a new work if already has two or more.
CREATE OR REPLACE FUNCTION prevent_duplicate_works ()
RETURNS TRIGGER
AS $$
BEGIN
    IF (
        SELECT
            COUNT(*)
        FROM
            works
        WHERE
            date_work = NEW.date_work AND master_id = NEW.master_id) >= 2 THEN
        RAISE EXCEPTION 'Cannot add entry. Date_work already has more than two entries with the
same master_id.';
    END IF;
    RETURN NEW;
END;
$$
LANGUAGE plpgsql;

CREATE TRIGGER prevent_duplicate_works_trigger
BEFORE INSERT ON works
FOR EACH ROW
EXECUTE FUNCTION prevent_duplicate_works ();

```

И Makefile для удобства:

```

PORT=5432
USER=tampio
DATABASE_NAME=car_service
HOST=localhost

DB_FILE=./car_service_initdb.sql

connection:
    psql -h ${HOST} -p ${PORT} -U ${USER} -d ${DATABASE_NAME}

recreate_db:
    psql -h ${HOST} -p ${PORT} -U ${USER} -d ${DATABASE_NAME} -f ${DB_FILE}

```

Такой подход позволяет запустить проект в две команды:

```

# bash
docker compose up
make recreate_db

```

После чего проект становится доступен на <http://localhost:8501>

## Основные экраны

Подключение к базе данных происходит с помощью SQLAlchemy, а данные для подключения берутся из файла secrets.toml

secrets.toml:

```
[connections.car_service_db]
type = "sql"
url = "postgresql://tampio:5130904%2F10002@car_service-postgres:5432/car_service"
```

Для заметки: символ %2F обозначает “/”. Это URL-кодировка специальных символов, чтобы они не воспринимались как часть адреса.

## Справочники

Изменения применяются прямо в PostgreSQL, в рамках транзакций. При возникновениях исключений на тригерах возникают ошибки, и транзакция отменится. Пример при попытке добавить машину с существующим номером:

Интерфейсы для других справочников выглядят таким же образом.

Cars

Для редактирования таблицы, нажмите дважды на ее ячейку

id	num	color	mark	is_foreign
1	a905ав777	Красный	Honda	<input checked="" type="checkbox"/>
3	б128кф939	Чёрный	Лада	<input type="checkbox"/>
4	а888щл913	Фиолетовый	Mazda	<input checked="" type="checkbox"/>
5	г123ар834	Бирюзовый	ЛИАЗ	<input type="checkbox"/>
7	г123ар838	Зеленый	Porsche	<input checked="" type="checkbox"/>
2	б123ка717	Белый	Хёндай	<input checked="" type="checkbox"/>

Изменения в 2 строчках...

id	num	color	mark	is_foreign
5	г123ар834	Бирюзовый	ЛИАЗ	<input type="checkbox"/>
7	г123ар838	Зеленый	Porsche	<input checked="" type="checkbox"/>

Применить изменения

Добавить

num	color	mark	is_foreign
о123ло321	Фиолетовый	Лада	false

Добавить запись

Удалить

Id записи

1

Удалить запись

Сброс

**InternalError:** (psycopg2.errors.RaiseException) Duplicate value detected for num: б123ка717  
CONTEXT: PL/pgSQL function enforce\_unique\_cars\_num() line 10 at RAISE [SQL: INSERT INTO cars (num, color, mark, is\_foreign) VALUES ('б123ка717', 'Фиолетовый', 'Лада', 'false')] (Background on this error at: <https://sqlalche.me/e/20/2j85>)

Traceback:

Отчёты

Реализация инфографики, требуемой в задаче

Отчёты

Общая стоимость обслуживания  
отечественных и импортных автомобилей

Выберите период обслуживания машин

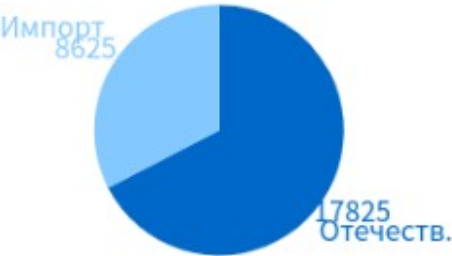
2023-08-31 – 2023-10-01

☒ Использовать хранимую функцию

Исполняемый SQL

```
SELECT * from service_cost_by_foreignness('2023-08-31','2023-10-01')
```

Стоимость обслуживания по "импортности" автомобилей [2023-08-31 - 2023-10-01] (в RUB)



is_foreign	service_cost
<input type="checkbox"/>	17,825
<input checked="" type="checkbox"/>	8,625

# 5 Мастеров, выполнивших наибольшее кол-во работ за период

Выберите месяц работы мастеров

2023-09-30

☒ Использовать хранимую функцию

Исполняемый SQL

SELECT \* from top\_five\_masters('2023-09-01','2023-10-01')

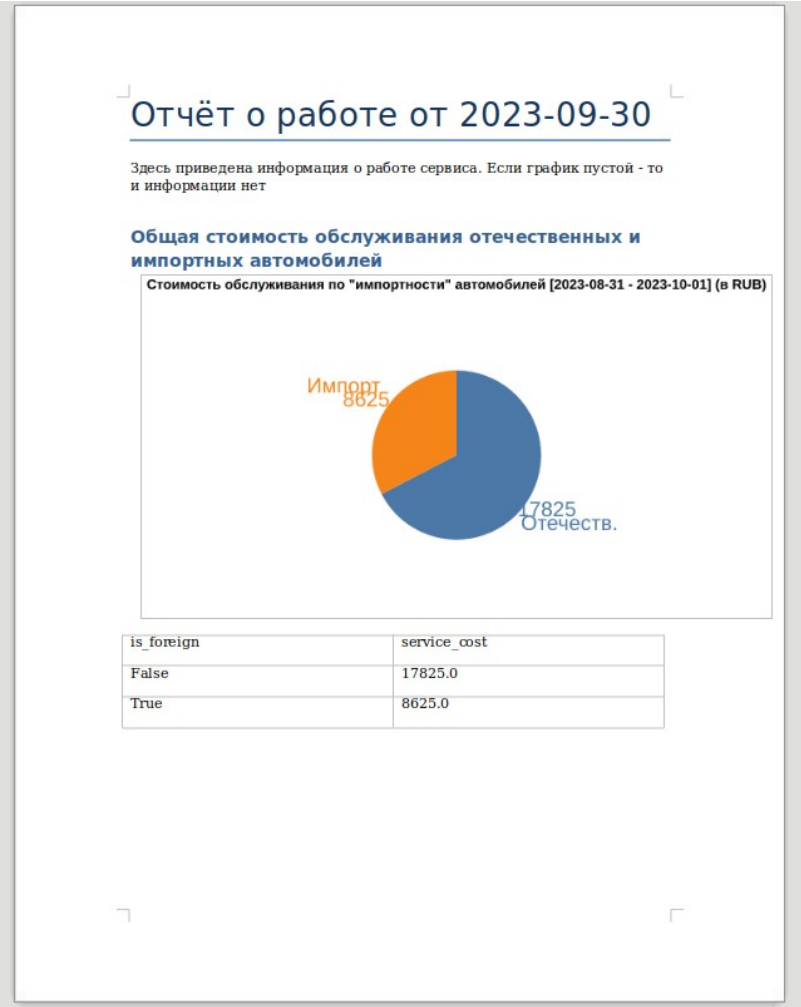
Топ 5 мастеров по кол-ву работ [2023-09-01 - 2023-10-01]



master_id	master_name	works_count
1	Василий	2
3	Юлиана	2
5	Пётр	2

Реализация инфографики, требуемой в задаче

Отчеты выгружаются в формате .docx и содержат инфографику и таблицы по требуемым метрикам. Использовалась библиотека python-docx, инфографика строилась с помощью vega-altair. При желании пользователь может экспортировать документ в любимый формат Пример отчета:





## Реализация главного окна оператора

Состоит из занятости операторов в сегодняшний день (чтобы оператор мог увидеть, кому можно назначить работы) Цветовая индикация и сортировку по количеству работ в день этому способствует.

Профиль

# Главная

Сегодня - 2023-09-30

## Занятость мастеров сегодня

master_id	name	works_count
5	Пётр	2
3	Юлиана	2
1	Василий	2
16	Афанасий	0
15	Афанасий	0
14	Афанасий	0
13	Афанасий	0
12	Афанасий	0
4	Анатолий	0
2	Сергей	0

Добавить новую работу

Посмотреть занятость мастера

Есть два основных меню – Добавить новую работу и посмотреть занятость мастера.

Рассмотрим оба по очереди:

# Добавить новую работу

Интерфейс интуитивно понятен, выпадающие списки реализованы таким образом, чтобы было удобно выбирать неопытному пользователю.

Добавить новую работу

Чтобы создать запись о работе, пожалуйста заполните поля

Мастер

Анатолий (id 4)

Машина

а905ав777 - Красный - Honda (id 1)

Услуга

Замена карбюратора - Отеч. 5750.00р. - Зарубеж. 8625.00р. (id 2)

Стоимость услуги: 8625.00р. (Импортная)

Дата

2023-09-30

Превью добавляемой работы

date_work	master_id	car_id	service_id
2023-09-30	4	1	2

Добавить работу

Машина

а905ав777 - Красный - Honda (id 1)

а905ав777 - Красный - Honda (id 1)

6128кф939 - Чёрный - Лада (id 3)

а888щл913 - Фиолетовый - Mazda (id 4)

г123ар834 - Бирюзовый - УАЗ (id 5)

г123ар838 - Лиловый - Porsche (id 7)

6123ка717 - Белый - Хёндай (id 2)

Дата

2023-09-30

<

September

>

2023

<

>

Su	Mo	Tu	We	Th	Fr	Sa
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

### Посмотреть занятость мастера

Здесь отображаются работы мастера за последний месяц

Зеленым выделены сегодняшние работы.

Посмотреть занятость мастера

Мастер

Василий (id 1)

Прим.: Зелёным отмечена сегодняшняя занятость

work_id	date_work	service_name	service_id	car_num	car_color	car_id
9	2023-09-30	Смена резины	3	6128кф939	Чёрный	3
8	2023-09-30	Замена карбюратора	2	6123ка717	Белый	2
13	2023-09-29	Замена карбюратора	2	а905ав777	Красный	1
12	2023-09-29	Замена карбюратора	2	а905ав777	Красный	1
15	2023-09-28	Смена резины	3	6128кф939	Чёрный	3
14	2023-09-28	Смена резины	3	6128кф939	Чёрный	3

# Выводы

Во время выполнения работы удалось плотно познакомиться с БД Postgres, что я вижу перспективным для своей карьеры. Мне понравилось строить этот проект и я овладел навыками базового проектирования программных систем, потренировался в докере. Со Streamlit возникали небольшие сложности, поскольку не было полноценного готового решения для изменения таблицы на стороне базы данных, пришлось писать интерфейс самому, благо фреймворк под это гибкий.

По итогу курсовой работы удалось реализовать прототип функционирующей системы, которая возможно даже имела бы реальное практическое применение. Однако проект не лишен недостатков – хранение пользователей в файле, как никак, не является стандартом индустрии, база данных не версионруется, что усложняет ее поддержку. По проведению подобных систем в производственную среду необходимо проделать большое количество дополнительной работы, которая выходит за пределы данной работы.

Помимо PostgreSQL я попробовал для себя новую библиотеку визуализации данных Altair, однако ее подход к визуализации данных мне не близок, как, например, matplotlib.

При составлении отчетов изначально пытался пользоваться библиотекой rypdf, однако создание pdf сопровождалось кучей проблем и не подходил под эти задачи, поэтому вариант с использованием python-docx сработал лучше.

# Литература

По большей части, документация библиотек

- <https://streamlit.io/gallery>
- <https://www.sqlalchemy.org/>
- <https://python-docx.readthedocs.io/en/latest/>
- <https://pypdf.readthedocs.io/en/stable/index.html>
- <https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-compose-on-ubuntu-22-04>
- <https://habr.com/ru/articles/155201/>
- <https://altair-viz.github.io/index.html>
- <https://docs.python.org/3/library/index.html>
- <https://www.teamfortress.com/>
- <https://realpython.com/python311-tomllib/>
- <https://www.postgresql.org/docs/current/sql-createtrigger.html>