**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра ИС**

**КУРСОВАЯ РАБОТА**

**по дисциплине «Управление данными»**

**Тема: Разработка базы данных для хранения сведений о зарегистрированных происшествиях.**

|  |  |  |
| --- | --- | --- |
| Студенты гр. 9362 |  | Быченков С. А. |
|  |  | Малютин Д. Л. |
|  |  | Динь Ч. Т. |
| Преподаватель |  | Обухов А. В. |

Санкт-Петербург

2023

**ЗАДАНИЕ**

**НА КУРСОВУЮ РАБОТУ**

|  |  |  |
| --- | --- | --- |
| Студенты Быченков С. А, Малютин Д. Л, Динь Ч. Т. | | |
| Группа 9362 | | |
| Тема работы: Разработка базы данных для хранения информации о прошивках, нагрузках и проектах. | | |
| Исходные данные: база данных должна содержать данные о прошивках (дату, автора, проект и т.д.), о нагрузках (обноски, номера, тип платформы, проект, режимы работы и т.д.), о проектах (название, подходящие нагрузки, платформы) и о платформах (название, образы). Помимо SQL запросов для создания таблиц базы данных, необходимо поддержать следующий функционал:   1. Автоматизированная вставка записей во все таблицы (интеграция с CI на buildbot); 2. Изменение данных о нагрузках пользователем (подразумевается, что информация о нагрузках создается и редактируется только вручную); 3. User friendly поиск (пользователь должен иметь возможность получить нужную информацию используя широкий спектр фильтров, все полученные данные должны быть отсортированы по информативным полям, например дата сборки прошивки, или проект, но атрибут автор не подойдет):    1. по названию платформы получить все прошивки;    2. поиск прошивки по ключевому слову;    3. по названию проекта получаем все прошивки;    4. по названию ветки получаем все прошивки;    5. по названиям проекта и платформы получаем все прошивки;    6. по названиям проекта и ветки получаем все прошивки;    7. по названиям ветки и платформы получаем все прошивки;    8. по названиям проекта, платформы и ветки получаем все прошивки;    9. получить все проекты;    10. получить все ветки;    11. получить все платформы;    12. получить все нагрузки;    13. найти подходящие платформы для проекта;    14. найти подходящие проекты для платформы;    15. по проекту посмотреть все нагрузки;    16. по платформе посмотреть все нагрузки;    17. по id нагрузки получить путь к файлу. 4. Создание трех ролей:    1. Администратор (всемогущий);    2. Пользователь (может читать из всех таблиц, а редактировать только нагрузки)    3. Buildbot (может добавлять данные во все таблицы) 5. Осуществление механизмов резервного копирования и восстановления БД, миграция БД в другую СУБД (MS SQL Server, MySQL). 6. Дополнительно:    1. Провести анализ и оптимизацию запросов индексацией    2. Устойчивость БД (использование транзакций, обработка исключений)    3. Реализовать REST API для взаимодействия с бд | | |
| Содержание пояснительной записки: «Содержание», «Введение», «Анализ предметной области», «Обоснование модели данных», «Обоснование выбора СУБД», «Описание функций групп пользователей», «Описание  функций управления БД», «Организация защиты БД», «Заключение»,  «Список использованных источников». | | |
| Предполагаемый объем пояснительной записки:  Не менее 50 страниц. | |
| Дата выдачи задания: 07.03.2023 |  |
| Дата сдачи работы: 01.05.2023 |
| |  |  |  | | --- | --- | --- | | Студенты гр. 9362 |  | Быченков С. А. | |  |  | Малютин Д. Л. | |  |  | Динь Ч. Т. | | Преподаватель |  | Обухов А. В. | | | |

**АННОТАЦИЯ**

В рамках курсовой работы реализуется реляционная база данных (далее сокращенно БД) для хранения информации о прошивках и нагрузках. В базе будет предусмотрена вся информация о прошивках, нагрузках и их связях. К БД прилагается REST API веб сервис, позволяющий взаимодействовать с бд пользователю.

**SUMMARY**

As part of the course work, a relational database (hereinafter referred to as DB) is implemented to store information about firmware and loads. The database will provide all the information about the firmware, loads and their connections. The REST API web service is attached to the database, allowing the user to interact with the database.

**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ** 6](#_Toc131891648)

[**1.** **АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ** 7](#_Toc131891649)

[**1.1.** **Выбор подхода к проектированию БД** 7](#_Toc131891650)

[**1.2.** **Описание объектов предметной области** 7](#_Toc131891651)

[**1.3.** **Описание входных параметров** 9](#_Toc131891652)

[**2.** **ОБОСНОВАНИЕ МОДЕЛИ ДАННЫХ** 10](#_Toc131891653)

[**2.1.** **Построение ER-модели** 10](#_Toc131891654)

[**2.2.** **Нормализация БД** 11](#_Toc131891655)

[**2.3.** **Реализация таблиц сущностей** 13](#_Toc131891656)

[**3.** **ОБОСНОВАНИЕ ВЫБРАННЫХ ТЕХНОЛОГИЙ** 15](#_Toc131891657)

[**4.**  **ОПИСАНИЕ ФУНКЦИЙ ГРУПП ПОЛЬЗОВАТЕЛЕЙ** 16](#_Toc131891658)

[**5.** **ОПИСАНИЕ ФУНКЦИЙ УПРАВЛЕНИЯ ДАННЫМИ** 19](#_Toc131891659)

[**5.1.** **Добавление и обновление данных в таблицах:** 19](#_Toc131891660)

[**5.2.** **Получение информации из БД** 22](#_Toc131891661)

[**6.** **ОРГАНИЗАЦИЯ ЗАЩИТЫ БД** 23](#_Toc131891662)

[**ЗАКЛЮЧЕНИЕ** 24](#_Toc131891663)

[**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ** 25](#_Toc131891664)

[ПРИЛОЖЕНИЕ А «ТЕКСТ ПРОГРАММЫ ПАРСЕРА (Python)» 34](#_Toc131891665)

[ПРИЛОЖЕНИЕ Б «ТЕКСТ ПРОГРАММЫ (SQL)» 38](#_Toc131891666)

[ПРИЛОЖЕНИЕ В «ТЕКСТ ПРОГРАММЫ (Python)» 38](#_Toc131891667)

# **ВВЕДЕНИЕ**

В представленной курсовой работе будет разработана система для хранения сведений о прошивках и нагрузках.

Идея работы заключается в упрощении и увеличении функциональности поиска прошивок, а также хранения и информации о нагрузках.

Обозначим несколько базовых определений:

* Прошивка – архив, загружаемый на платформу, необходимый для ее функционирования.
* Платформа – программируемая микросхема, основа для нагрузки
* Нагрузка – устройство управляющее техническим аппаратом, например БПЛА(БЛА).
* Ветка – ветка разработки в системе контроля версий.

# **АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

# **Выбор подхода к проектированию БД**

В задании к курсовой работе четко определены почти все необходимые сущности, поэтому функциональный метод отлично подходит для дальнейшего проектирования.

# **Описание объектов предметной области**

В рамках данной задачи можно выделить 6 объектов: прошивка, нагрузка, автор, платформа, ветка и проект. Для каждого из объектов определим соответствующую информацию, которая должна храниться в БД (Таблица 1):

Таблица 1. Информационные объекты БД

|  |  |
| --- | --- |
| **Объект** | **Хранимая информация** |
| Автор | Идентификатор |
| Имя в системе контроля версий |
| Email в системе контроля версий |
| Проект | Идентификатор |
| Название |
| Платформа | Идентификатор |
| Название |
| Образ |
| Ветка | Идентификатор |
|  | Название |
| Прошивка | Идентификатор |
|  | Ветка |
|  | Платформа |
|  | Проекты, для которых подходит прошивка |
|  | Автор |
|  | Пройденные тесты |
|  | Дата-время сборки прошивки |
|  | Хэш коммита запустившего сборку |
|  | Имя файла (прошивки) |
| Нагрузка | Идентификатор |
| Децимальный номер (тип изделия) (опционален) |
| Заводской номер (опционален) |
| Платформа |
| Проект |
| Имя |
| Поддержка режима RSA |
| Поддержка режима RTR |
| Поддержка режима RC |
| Поддержка режима RTC |
| Поддержка режима RP |
| Поддержка режима RLO |
| Поддержка режима FC |
| Путь к файлу с измеренными характеристиками  (К примеру, АЧХ) |
| Текстовое описание |

# **Описание входных параметров**

На вход мы получаем строку без четкой структуры, являющуюся названием файла прошивки. Пример названия: “Emulator-Aniva-master-6361-a6635c\_RTR.tar.xz”. Также, в ходе бизнес-анализа были выявлены правила сопоставления псевдонимов проекта официальному названию и проекта к нагрузкам. Пример сопоставления в составленном для автоматизации парсинга названий прошивок json файле: “‘Emulator-Aniva': {'Анива': ['РТК']}”. Пара Платформа-ПсевдонимПроекта (Emulator-Aniva) определяет принадлежность прошивки к настоящим названиям проектов (только один проект 'Анива'), а следовательно, и к нагрузкам из этого проекта (только одна нагрузка 'РТК').

Сам парсинг заключается в последовательном вычленении атрибутов из названия прошивки. Сначала мы достаем имя ветки и хеш. Затем переходим в папку проекта на нужную ветку и находим информацию об авторе коммита по хэшу коммита. Также в информации о коммите мы находим дату и время, которые приводим к iso формату, (TIMESTAMP (5) в postgres). Далее из названия мы достаем платформу и псевдоним проекта.

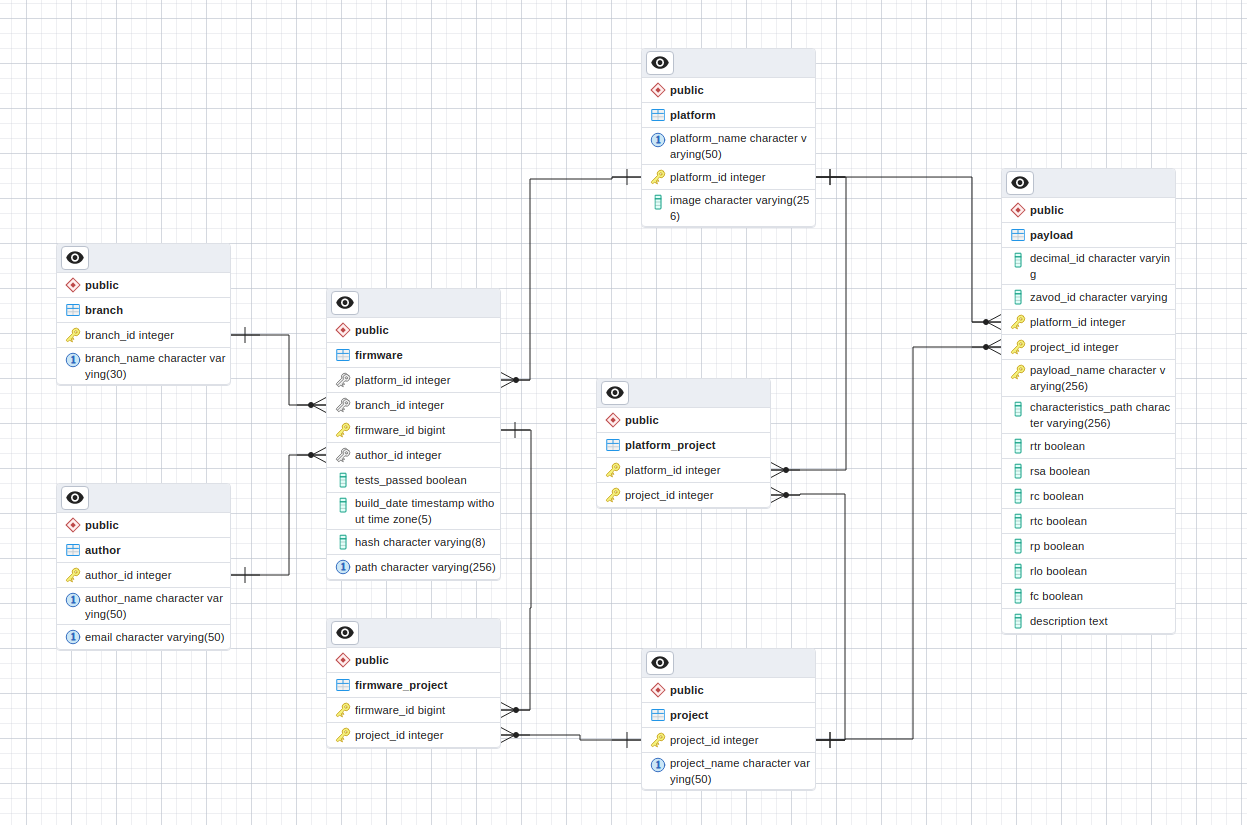
Всю информацию мы сохраняем и получаем из “ Emulator-Aniva-master-6361-a6635c\_RTR.tar.xz” следующую строку: “Emulator-Aniva-master-6361-a6635c\_RTR.tar.xz;;master;;a6635c;;Ivan Ivanoff;;iv.ivanov@spb.ru;;2022-10-12 14:25:09;;Aniva;;Emulator”.

Код принимающий файл, содержащий список названий прошивок, и создающий файл с полной информацией о прошивках находится в приложении А.

# **ОБОСНОВАНИЕ МОДЕЛИ ДАННЫХ**

# **Построение ER-модели**

В результате анализа полученных данных была сформирована следующая ER-модель (рисунок 1):

 Рисунок 1 – «ER-модель»

# **Нормализация БД**

Бизнес-модель определяет следующие ограничения и отношения:

* Каждая прошивка работает только на одной платформе
* На одной платформе работает много прошивок

**Прошивка M:1 платформа**

* Проекту подходит много прошивок (как минимум разные версии одной и той же прошивки)
* Каждая прошивка может подходить для нескольких проектов

**Прошивка M:N проект**

* Каждая прошивка собрана с конкретной ветки
* На каждой ветке было собрано много прошивок

**Прошивка M:1 ветка**

* Каждая прошивка собрана после коммита конкретного автора
* Каждый автор триггерит множество сборок (много прошивок)

**Прошивка M:1 автор**

* Проект может “переехать” на новую платформу, это значит, что один проект может работать на нескольких платформах, так как не происходит обновление данных, просто новые прошивки будут уже на другой платформе
* На одной платформе работает множество проектов

**Проект M:N платформа**

* Изделие/нагрузка основано на одной платформе
* На одной платформе работает множество нагрузок

**Нагрузка M:1 платформа**

* Изделие/нагрузка неотрывно от проекта, названия могут повторятся, но уникальными являются пары проект-нагрузка

**Проект 1:M нагрузка**

Вышеперечисленные условия наглядно отражены в ER-модели.

Суррогатные ключи были введены для экономии памяти и упрощения работы с бд. Понятно, что можно использовать в отношениях ветка, проект, платформа, автор естественный ключ, но при создании внешних ключей в таблице firmware, в которой больше всего записей, мы будем хранить строки, хотя гораздо выгоднее хранить

На пересечении каждого столбца и каждой строки находятся только элементарные значения атрибутов и не может содержать повторяющиеся строки благодаря уникальным индексам, следовательно отношение находится в первой нормальной форме.

Все отношения находятся в первой нормальной форме, в них есть суррогатный первичный ключ, и, соответственно, нет неполных функциональных зависимостей непервичных атрибутов от первичного ключа, следовательно отношение находится во второй нормальной форме. Исключением является отношение нагрузка, в котором не был добавлен суррогатный ключ для экономии памяти, так как на это отношение нет внешних ключей. В этой таблице есть ограничение на уникальность кортежа атрибутов проект, имя нагрузки, платформа, который и является составным первичным ключом, но в этом отношении так же нет неполных функциональных зависимостей непервичных атрибутов от первичного ключа, ведь каждое подмножество ключа не является уникальным, следовательно отношение находится во второй нормальной форме.

Все отношения находятся во второй нормальной форме и в них отсутствуют транзитивные функциональные зависимости, следовательно, все отношения находятся в третьей нормальной форме. Исключением является отношение firmware с атрибутом path, содержащим в себе информацию из других атрибутов, но бизнес-логика не подразумевает обновления и удаления записей в firmware. Соответственно мы можем игнорировать это и не бояться аномалий обновления и удаления, то есть нарушения целостности базы данных.

Все отношения находятся в третьей нормальной форме и все первичные ключи простые, следовательно все отношения находятся в нормальной форме Бойса-Кодда. Исключением является отношение нагрузки с составным ключом, рассмотрим остальные атрибуты: все потенциальные ключи находятся в первичном ключе – здесь нет зависимостей ключевых атрибутов от неключевых. Заводской номер, конечно, однозначно определяет запись, но является опциональным, а, следовательно, не может быть уникальным, и вообще быть ключем. Это значит, что и отношение payload находится в нормальной форме Бойса-Кодда.

# **Реализация таблиц сущностей**

Создание бд происходит после следующих команд:

psql -d postgres  
create database firmwares;  
psql -d firmwares  
\i sql/create\_db.sql  
\i sql/procedures.sql

Создание таблиц:

*--DROP TABLE IF EXISTS author;*CREATE TABLE IF NOT EXISTS author(  
author\_id SERIAL PRIMARY KEY,  
author\_name VARCHAR(50) NOT NULL DEFAULT '',  
email VARCHAR(50) NOT NULL DEFAULT '',  
UNIQUE (author\_name, email));  
  
  
*--DROP TABLE IF EXISTS platform;*CREATE TABLE IF NOT EXISTS platform(  
platform\_name VARCHAR(50) UNIQUE NOT NULL DEFAULT '',  
platform\_id SERIAL PRIMARY KEY,  
image VARCHAR(256) NOT NULL DEFAULT '');  
  
*--DROP TABLE IF EXISTS project;*CREATE TABLE IF NOT EXISTS project(  
project\_id SERIAL PRIMARY KEY,  
project\_name VARCHAR(50) UNIQUE NOT NULL DEFAULT '');  
  
*--DROP TABLE IF EXISTS branch;*CREATE TABLE IF NOT EXISTS branch(  
branch\_id SERIAL PRIMARY KEY,  
branch\_name VARCHAR(30) UNIQUE NOT NULL DEFAULT '');  
  
  
*--DROP TABLE IF EXISTS payload;*CREATE TABLE IF NOT EXISTS payload(  
decimal\_id VARCHAR NOT NULL DEFAULT '',  
zavod\_id VARCHAR NOT NULL DEFAULT '',  
platform\_id SERIAL NOT NULL,  
project\_id SERIAL NOT NULL,  
payload\_name VARCHAR(256) NOT NULL DEFAULT '',  
characteristics\_path VARCHAR(256) NOT NULL DEFAULT '',  
rtr BOOLEAN NOT NULL DEFAULT FALSE,  
rsa BOOLEAN NOT NULL DEFAULT FALSE,  
rc BOOLEAN NOT NULL DEFAULT FALSE,  
rtc BOOLEAN NOT NULL DEFAULT FALSE,  
rp BOOLEAN NOT NULL DEFAULT FALSE,  
rlo BOOLEAN NOT NULL DEFAULT FALSE,  
fc BOOLEAN NOT NULL DEFAULT FALSE,  
description TEXT NULL,  
CONSTRAINT proj  
 FOREIGN KEY(project\_id)  
 REFERENCES project(project\_id),  
CONSTRAINT plat  
 FOREIGN KEY(platform\_id)  
 REFERENCES platform(platform\_id),  
PRIMARY KEY (payload\_name, project\_id, platform\_id));  
  
*--DROP TABLE IF EXISTS firmware;*CREATE TABLE IF NOT EXISTS firmware(  
platform\_id SERIAL,  
branch\_id SERIAL,  
firmware\_id BIGSERIAL PRIMARY KEY,  
author\_id SERIAL,  
tests\_passed BOOLEAN NOT NULL DEFAULT FALSE,  
build\_date TIMESTAMP(5) NOT NULL DEFAULT CURRENT\_TIMESTAMP,  
hash VARCHAR(8) NOT NULL, *-- хэш не является ключом, так как один коммит порождает несколько прошивок*path VARCHAR(256) UNIQUE NOT NULL,  
CONSTRAINT auth  
 FOREIGN KEY(author\_id)  
 REFERENCES author(author\_id),  
CONSTRAINT plat  
 FOREIGN KEY(platform\_id)  
 REFERENCES platform(platform\_id),  
CONSTRAINT branch\_constraint  
 FOREIGN KEY(branch\_id)  
 REFERENCES branch(branch\_id)  
);  
  
  
*--DROP TABLE IF EXISTS platform\_project;*CREATE TABLE IF NOT EXISTS platform\_project(  
platform\_id SERIAL,  
project\_id SERIAL,  
CONSTRAINT proj  
 FOREIGN KEY(project\_id)  
 REFERENCES project(project\_id),  
CONSTRAINT plat  
 FOREIGN KEY(platform\_id)  
 REFERENCES platform(platform\_id),  
PRIMARY KEY (platform\_id, project\_id));  
  
  
*--DROP TABLE IF EXISTS firmware\_project;*CREATE TABLE IF NOT EXISTS firmware\_project(  
firmware\_id BIGSERIAL,  
project\_id SERIAL,  
CONSTRAINT proj  
 FOREIGN KEY(project\_id)  
 REFERENCES project(project\_id),  
CONSTRAINT firm  
 FOREIGN KEY(firmware\_id)  
 REFERENCES firmware(firmware\_id),  
PRIMARY KEY (firmware\_id, project\_id));

# **ОБОСНОВАНИЕ ВЫБРАННЫХ ТЕХНОЛОГИЙ**

Логическая модель отображения физической модели данных требует структуризации, БД будет использоваться для построения итоговых запросов, а также будет производиться поиск по нескольким характеристикам, следовательно будет удобно использовать реляционную базу данных. Для курсовой работы в качестве СУБД была выбрана база данных Postgresql. Она отлично подходит для реализации реляционной БД, а также предоставляет большой список поддерживаемых инструментов по управлению. Также немаловажным фактором выступила лицензия данного продукта, которая позволяет использовать его на бесплатных основаниях. Для запросов вставки, которые не подразумевают возвращения значений был использован язык plpgsql, который хорошо интегрирован в субд. Для бекенда был выбран python. Использовал Fastapi, так как это очень быстрый фреймворк, который отлично подходит для написания REST api. В нем очень удобная сериализация/валидация json через pydantic, который нативно работает с аннотациями типов питона, в том числе с дженериками. Fastapi асинхронный, а в этом api все обработчики без блокирующего ввода/вывода, поэтому все обработчики я реализовал асинхронными. Также огромным плюсом в копилке Fastapi является автоматическая генерация документации в Swagger/redoc. Я использовал реляционную PostgreSQL, так как у хранимых данных есть четкая структура. Вместо orm я использовал напрямую драйвер asyncpg, что позволило лучше контролировать реальные запросы отправляемые в бд. Также прямое использование драйвера вместо орм значительно ускоряет взаимодействие с бд.

# **4. ОПИСАНИЕ ФУНКЦИЙ ГРУПП ПОЛЬЗОВАТЕЛЕЙ**

1. **Пользователь**

Просмотр всей информации в бд; изменение данных о нагрузках (описание и режимы работы, путь к характеристикам).

1. **Администратор**

Может вносить любые правки в любую таблицу базы данных.

Мы не будем отдельно создавать администратора, пользователь создавший бд и будет администратором.

1. **Buildbot**

Может добавлять информацию в любую таблицу базы данных.

Создаются пользователи скриптом через psql:

\i sql/create\_users.sql

Сам скрипт:

REVOKE ALL ON ALL TABLES IN SCHEMA public FROM buildbot;  
REVOKE ALL ON ALL SEQUENCES IN SCHEMA public FROM buildbot;  
REVOKE ALL ON ALL PROCEDURES IN SCHEMA public FROM buildbot;  
DROP ROLE buildbot;  
CREATE ROLE "buildbot" WITH LOGIN PASSWORD 'qwe';  
GRANT SELECT ON ALL TABLES IN SCHEMA public TO buildbot;  
GRANT INSERT ON ALL TABLES IN SCHEMA public TO buildbot;  
GRANT EXECUTE ON ALL PROCEDURES IN SCHEMA public to buildbot;  
GRANT USAGE, SELECT ON ALL SEQUENCES IN SCHEMA public TO buildbot;  
  
  
REVOKE ALL ON ALL TABLES IN SCHEMA public FROM common\_user;  
REVOKE ALL ON ALL SEQUENCES IN SCHEMA public FROM common\_user;  
REVOKE ALL ON ALL PROCEDURES IN SCHEMA public FROM common\_user;  
DROP ROLE common\_user;  
CREATE ROLE "common\_user" WITH LOGIN PASSWORD '321';  
GRANT SELECT ON TABLE firmware\_project, firmware, author, platform\_project, payload, project, branch, platform TO common\_user;  
GRANT UPDATE ON TABLE payload TO common\_user;  
GRANT USAGE, SELECT ON ALL SEQUENCES IN SCHEMA public TO common\_user;  
GRANT UPDATE ON TABLE payload TO common\_user;  
GRANT EXECUTE ON PROCEDURE update\_payload TO common\_user;

После выполнения скрипта создания таблиц мы можем посмотреть на таблицу прав доступа с помощью \dp в psql (рисунок 2).



Рисунок 2 – «таблица прав доступа часть 1»

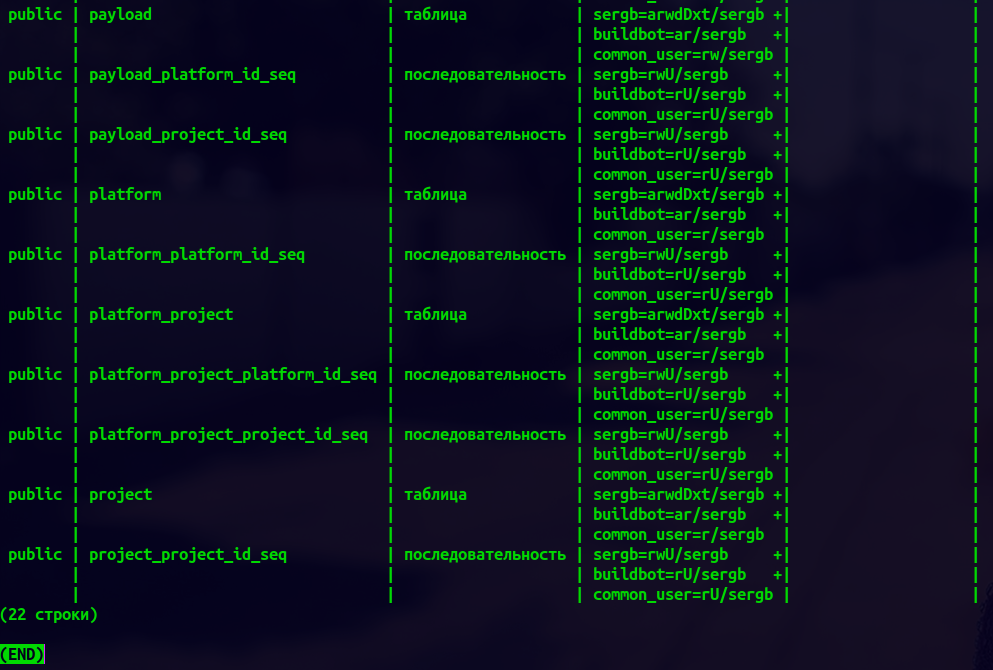


Рисунок 3 – «таблица прав доступа часть 2»

# **ОПИСАНИЕ ФУНКЦИЙ УПРАВЛЕНИЯ ДАННЫМИ**

# **Добавление и обновление данных в таблицах:**

Реализация процедур вставки и удаления данных см. в Приложении Б.

В комментарии перед процедурой описано ее назначение.

Заполнение реализовано через python и psycopg2. Все заполнения реализованы через транзакции и должны быть выполнены строго последовательно, поэтому заполнение выполняется синхронно. Код заполнения на python см. в Приложении В.

# **Получение информации из БД**

Для чтения разработки API для взаимодействия с бд были использованы asyncpg, python3, fastapi. Отдельно sql запросы см. в Приложении Г. Код бекенда в приложении Д.

В общих чертах работа на бекенде выглядит следующим образом:

При инициализации приложения создается пул соединений, позволяющий оптимально обрабатывать большое количество запросов - на личном 10 летнем ноутбуке за 2 минуты обработал 50000 посланных асинхронно, то есть, по сути, одновременно, http-запросов такого вида:

SELECT f.firmware\_id, pl.platform\_name, pa.payload\_name, a.author\_name, a.email,  
 b.branch\_name, f.build\_date, f.tests\_passed  
 FROM author a  
 JOIN firmware f USING (author\_id)  
 JOIN branch b USING (branch\_id)  
 JOIN platform pl USING (platform\_id)  
 JOIN firmware\_project fp USING (firmware\_id)  
 JOIN project pr USING (project\_id)  
 JOIN payload pa USING (project\_id)  
 WHERE pr.project\_name = '**{**project**}**'  
 ORDER BY f.platform\_id, f.branch\_id, f.build\_date DESC;

Выглядит это так:

**@app.on\_event**("startup")  
async def **get\_pool**():  
 try:  
 app.pool = await asyncpg.**create\_pool**(host="localhost",  
 port=5432,  
 user="common\_user",  
 database="firmwares",  
 password="321",  
 command\_timeout=60)  
 except ConnectionRefusedError:  
 print("ERROR: Can not connect to database")

Далее мы создаем промежуточное программное обеспечение (middleware) и зависимость, которую будем использовать для инъекции. Не вдаваясь в подробности, это код, часть которого будет выполняться перед обработкой запроса, а другая часть после. Мы используем этот механизм для соблюдения принципа “DRY”, ведь иначе пришлось бы в каждом обработчике писать создание подключения в пуле соединений. Выглядит зависимость, middleware и один из обработчиков так:

**@app.middleware**("http")  
async def **get\_connection**(request: Request, call\_next):  
 try:  
 request.state.db = await app.pool.**acquire**()  
 except Exception:  
 await request.state.db.**close**()  
 return await call\_next(request)  
  
  
async def **get\_conn**(request: Request) -> Union[asyncpg.pool.PoolConnectionProxy, None]:  
 yield request.state.db  
 await request.state.db.**close**()  
 del request.state.db

**@app.get**("/async\_async\_pool/{project}",  
 response\_model=List[FirmwarePlatformBranch],)  
async def **get\_fw\_by\_project**(project: Annotated[str, **Path**(title="The project of the firmware to get")],  
 connection=**Depends**(get\_conn)  
 ) -> Any:  
 result = await db.**fw\_by\_project**(connection, project)  
 if result:  
 return result  
 raise **HTTPException**(status\_code=status.HTTP\_404\_NOT\_FOUND,  
 detail=[{"loc": ["project"],  
 "msg": "no firmwares for this project"}])

Функция, вызываемая в обработчике, принимает объект соединения с бд, а также необходимые для запроса на выборку параметры, то есть, в данном случае – название проекта. Сама функция выглядит так:

async def **fw\_by\_project**(conn, project: str):  
 *# async with pool.acquire() as conn:* return await conn.**fetch**(f"""SELECT f.firmware\_id, pl.platform\_name, pa.payload\_name, a.author\_name, a.email,  
 b.branch\_name, f.build\_date, f.tests\_passed  
 FROM author a  
 JOIN firmware f USING (author\_id)  
 JOIN branch b USING (branch\_id)  
 JOIN platform pl USING (platform\_id)  
 JOIN firmware\_project fp USING (firmware\_id)  
 JOIN project pr USING (project\_id)  
 JOIN payload pa USING (project\_id)  
 WHERE pr.project\_name = '**{**project**}**'  
 ORDER BY f.platform\_id, f.branch\_id, f.build\_date DESC;  
 """)

Рассматривать каждый обработчик подробно не вижу смысла, так как они отличаются только принимаемыми параметрами, и вызываемой функцией.

# **ОРГАНИЗАЦИЯ ЗАЩИТЫ БД**

Для решения проблемы целостности, возникающей из-за пользовательского ввода и обновления, были использованы обработчики исключений на каждом шаге пользовательского ввода на уровне языка программирования, а также были созданы триггеры, срабатывающие на исключительные пользовательские ситуации.

Для обеспечения физической защиты, администратору базы данных требуется проводить периодическое резервное копирование всех имеющихся данных. Полное резервное копирование стоит проводить раз в неделю, для сохранения отчетности и статистики. Частичное резервное копирование стоит проводить раз в день, под частичным резервным копированием понимается информация о происшествиях.

Для обеспечения конфиденциальности данных была сделана авторизация и однофакторная аутентификации пользователей на основе знания пароля.

Также для обеспечения ограниченности доступа к информации были предусмотрены ограничения возможностей пользователей: пользователи взаимодействуют с базой данных только через консольное приложение и не имеют возможности напрямую писать SQL запросы. После ввода логина и пароля определяется группа пользователя и только администратор имеет право обращаться к БД через SQL запросы.

# **ЗАКЛЮЧЕНИЕ**

В процессе выполнения курсового проекта была спроектирована и разработана реляционная база данных для хранения сведений о зарегистрированных происшествиях. Также в качестве инструмента по работе с базой было реализовано приложение, которое демонстрирует предоставление всех требуемых сведений из технического задания к работе.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Дейт, К. Дж. Введение в системы баз данных, 8-е издание.: Пер. с англ. — М.: Издательский дом "Вильяме", 2005. — 1328 с.;
2. Шварц Б., Зайцев П., Ткаченко В., Заводны Дж., Ленц А., Бэллинг Д. MySQL. Оптимизация производительности, 2-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2010. – 832 с., ил.

**ПРИЛОЖЕНИЕ А «РУКОВОДСТВО ПОЛЬЗОВАТЕЯ»**

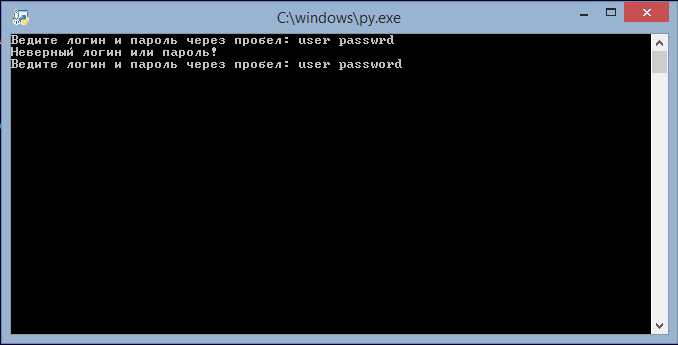
При запуске приложения необходимо пройти авторизацию и аутентификацию (рисунок 1). 

Рисунок 1 – авторизация

Если Вы авторизованы как пользователь, то Вам будет предложен следующий набор функций (рисунок 2).

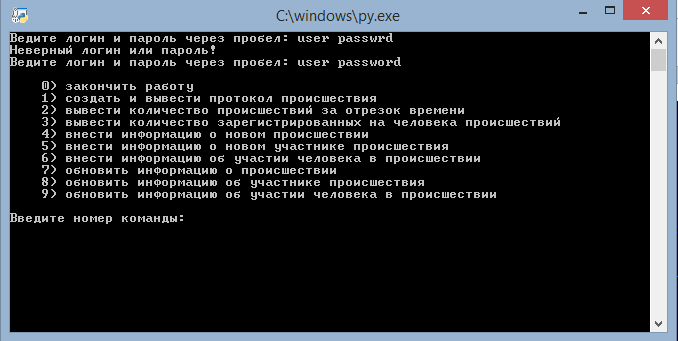


Рисунок 2 – функции

Чтобы создать и вывести протокол происшествия нажмите 1, а затем введите регистрационный номер происшествия (рисунок 3):

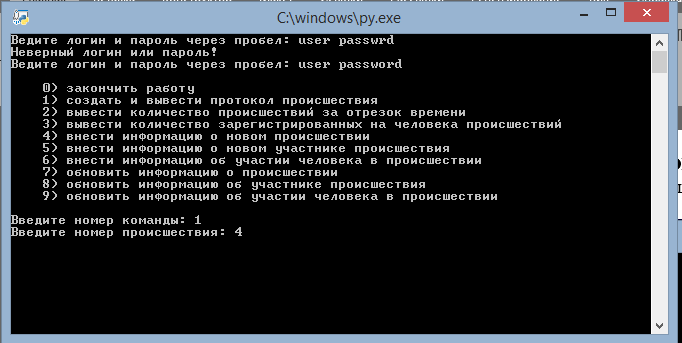


Рисунок 3 – генерация протокола

Далее на экран будет выведен требуемый протокол (рисунок 4), а также автоматически в папке, где находиться программа будет сгенерирован файл “происшествие №n” (рисунок 5).

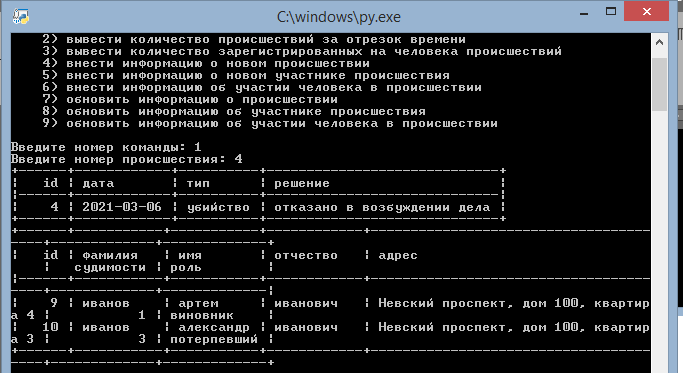


Рисунок 4 – вывод результата

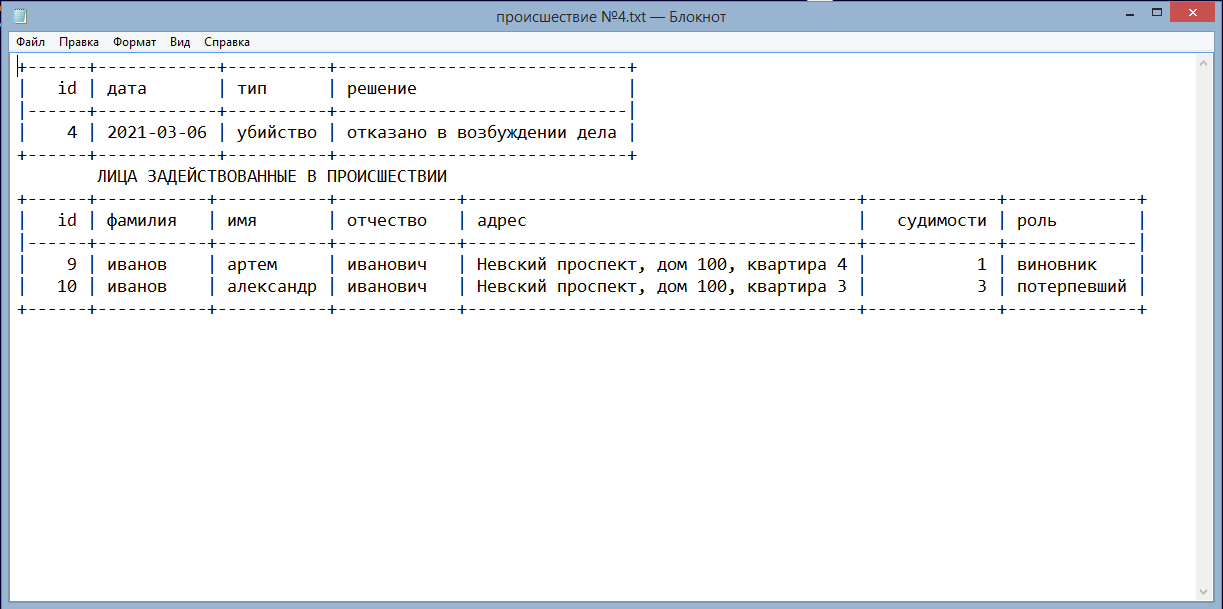


Рисунок 5 – сгенерированный протокол

Для того чтобы вывести количество происшествий, произошедших за определенный промежуток времени, нажмите 2 и введите две даты, начало и конец промежутка (рисунок 6):

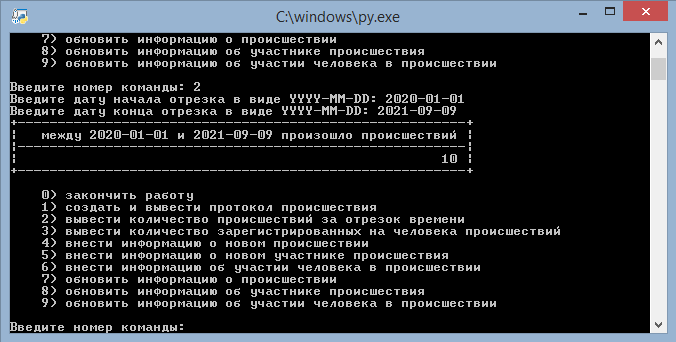
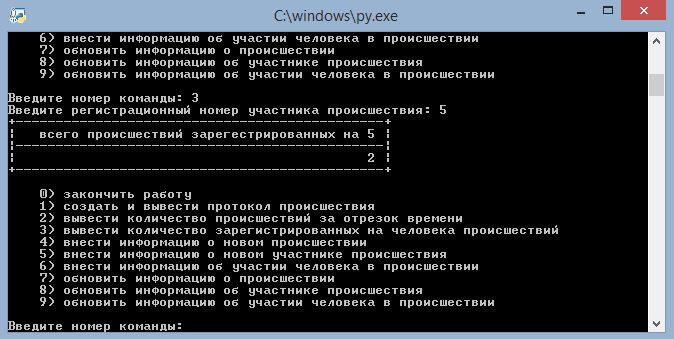


Рисунок 6 – вывод результата

Для того чтобы вывести количество зарегистрированных на человека происшествий, нажмите 3 и введите регистрационный номер человека (рисунок 7):

 Рисунок 7 – вывод результата

Для того чтобы внести информацию о новом происшествии, нажмите 4 и введите требуемые данные о происшествии (рисунок 8):

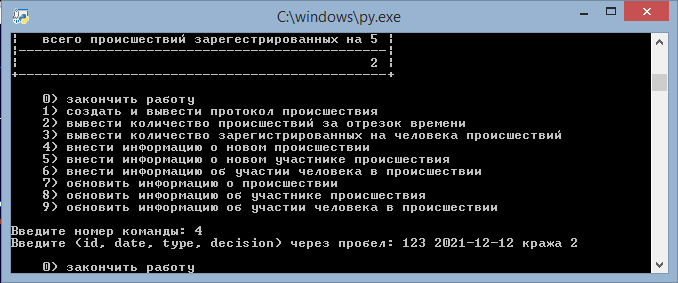


Рисунок 8 – вставка данных

Для того чтобы внести информацию о новом участнике происшествия, нажмите 5 и введите требуемые данные о человеке (рисунок 9):

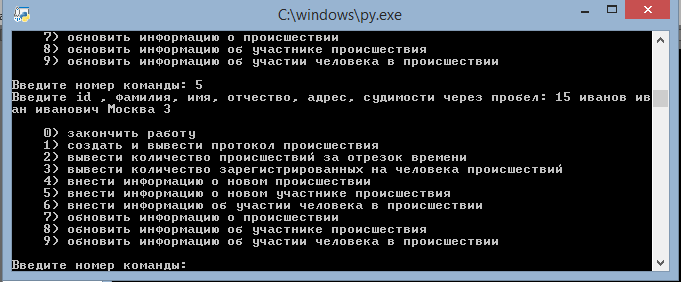


Рисунок 9 – вставка данных

Для того чтобы внести информацию об участии человека в происшествии, нажмите 6 и введите требуемые данные (рисунок 10):

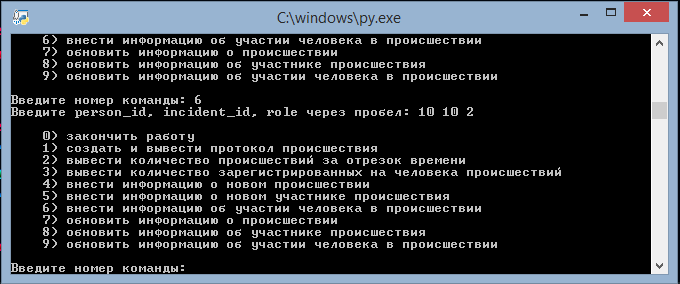


Рисунок 10 – вставка данных

Для того чтобы обновить данные о происшествии нажмите 7 и введите новые данные (рисунок 11):

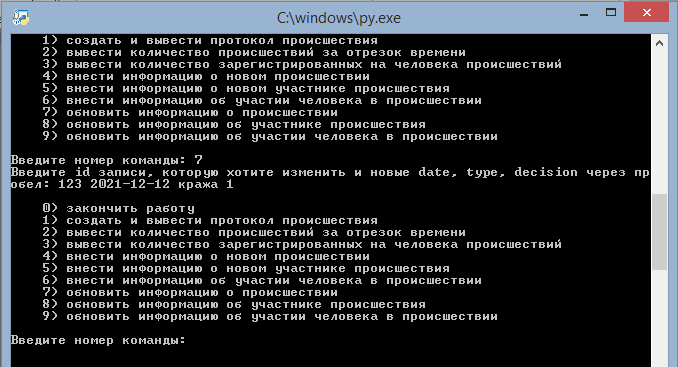


Рисунок 11 – обновление данных

Для того чтобы обновить данные о человеке нажмите 8 и введите новые данные (рисунок 12):

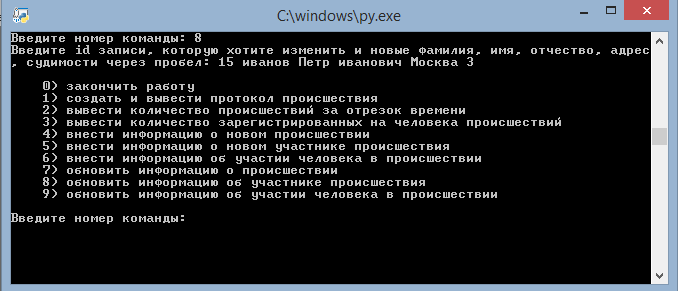


Рисунок 12 – обновление данных

Для того чтобы обновить данные об участии человека в происшествии нажмите 9 и введите новые данные (рисунок 13):

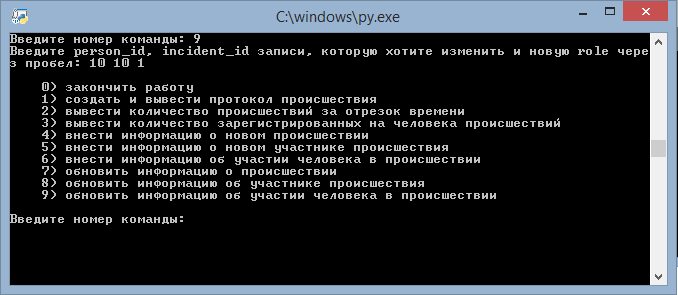


Рисунок 13 – обновление данных

Чтобы закончить работу с базой данных нажмите 0, чтобы закрыть программу нажмите enter (рисунок 14):

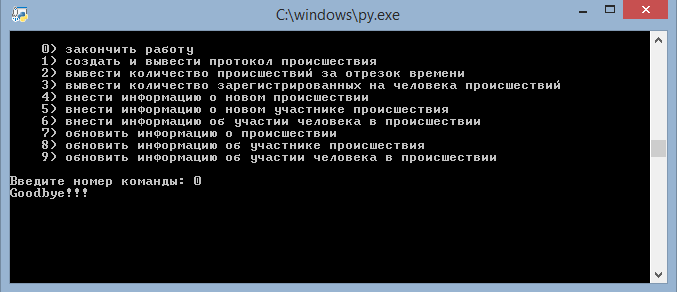


Рисунок 14 – завершение работы

Если Вы авторизованы как администратор, то вам будет предоставлен другой интерфейс, в котором можно обращаться к БД напрямую через SQL-запросы (рисунок 15):

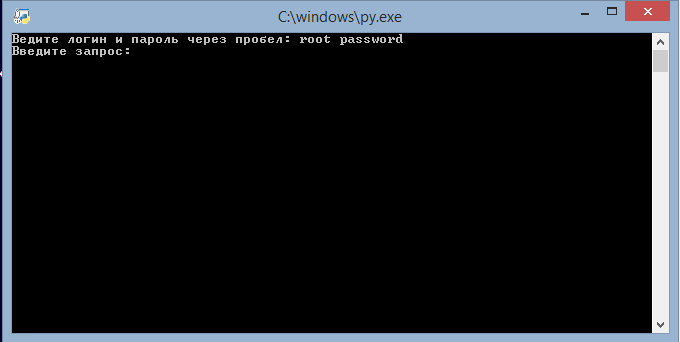


Рисунок 15 – администратор

## ПРИЛОЖЕНИЕ А «ТЕКСТ ПРОГРАММЫ ПАРСЕРА (Python)»

import os  
import subprocess  
from datetime import datetime  
from typing import Tuple  
  
"""  
This module is used to get all information from filenames  
you need file called 'all\_firmwares'  
you wil get file called 'parsed'  
we get commit, branch, project, platform from filename  
in commonpocket we get author name and email by parsing "git show pretty=fuller hash"  
"""  
  
  
def **branch\_hash**(name: str) -> Tuple[str, str, str]:  
 *"""  
 Use this function with "> file" to get file  
 with structure filename-branch-hash  
 """* b = ("testing", "master", "release\_21", "release\_22", "release\_20", "HEAD")  
 rx\_tx = ("rx", "tx", "azurit", "UAV", "0818", "mt501", "lab", "0818\_UavField", "Ground", "2640\_UavField",  
 "chrome", "um", "973\_Dubrava\_RD", "2640", "0826", "0826\_RtkNavi", "0826\_OrlanUM", "0826\_Dubrava",  
 "field")  
 c = ("6250Hz", "comint", "rev2", "1ch")  
 *# name = "FileGenSG200-ubuntu18-259-f28f214a.tar.xz"* if "Model\_RTR" in name:  
 commit = name.**split**("-")[4].**split**(".")[0]  
 branch = name.**split**("-")[3]  
 return name.**strip**(), branch, commit  
 elif "FileGenSG200" in name:  
 branch = "master"  
 commit = name.**split**("-")[3].**split**(".")[0]  
 return name.**strip**(), branch, commit  
 else:  
 branch = name.**split**("-")[2]  
 if len(name.**split**("-")) > 4:  
 commit = name.**split**("-")[4].**split**("\_")[0]  
 else:  
 commit = name.**split**("-")[3].**split**("\_")[0]  
 if branch in rx\_tx:  
 branch = name.**split**("-")[3]  
 if len(name.**split**("-")) > 5:  
 commit = name.**split**("-")[5].**split**("\_")[0]  
 else:  
 commit = name.**split**("-")[4].**split**("\_")[0]  
 if branch in c:  
 branch = name.**split**("-")[4]  
 if len(name.**split**("-")) > 6:  
 commit = name.**split**("-")[6].**split**("\_")[0]  
 else:  
 commit = name.**split**("-")[5].**split**("\_")[0]  
  
 if branch in b:  
 res = (name.**strip**(), branch, commit)  
 elif any(list(map(lambda x: x in branch, b))):  
 commit = name.**split**("-")[name.**split**("-").**index**(branch) + 1].**split**("\_")[0]  
 res = (name.**strip**(), b[b.**index**(max(b))], commit)  
 else:  
 print(name)  
 raise IndexError  
 return res  
  
  
def **author**(line: Tuple[str, str, str]) -> Tuple[str, str, str, str, str, str]:  
 *"""  
 You'll need a commonpocket in the root dir of project  
 filename;;branch;;hash;;author\_name;;author\_email;;date\_time  
 """* if "FileGenSG200" in line[0]:  
 os.**chdir**("../appforsg200")  
 if "Model\_RTR" in line[0]:  
 os.**chdir**("../python-model")  
 try:  
 subprocess.**check\_output**(["rm", "./.git/index.lock", "./.git/HEAD.lock"],  
 encoding='utf-8',  
 stderr=subprocess.STDOUT,  
 universal\_newlines=True)  
 except subprocess.CalledProcessError:  
 pass  
 *# print(f"Status : {exc.returncode}, \n{exc.output}")  
 # raise Exception(f"Status : {exc.returncode}, \n{exc.output}")  
  
 # line = "Emulator-Aniva-master-6512-8731e6\_RTR.tar.xz;;master;;8731e6"* branch = line[1].**strip**()  
 commit = line[2].**strip**()  
  
 try:  
 subprocess.**check\_output**(['git',  
 'checkout',  
 branch,  
 ],  
 encoding='utf-8',  
 stderr=subprocess.STDOUT,  
 universal\_newlines=True)  
 output\_git\_log = subprocess.**check\_output**(["git",  
 "show",  
 "--pretty=fuller",  
 commit,  
 ],  
 encoding='utf-8',  
 stderr=subprocess.STDOUT,  
 universal\_newlines=True)  
 if "Merge" in output\_git\_log:  
 a = output\_git\_log.**split**("**\n**")[4][8:].**strip**()  
 res = (line[0].**strip**(), line[1].**strip**(), line[2].**strip**(),  
 a[:a.**index**("<")].**strip**(),  
 a[a.**index**("<") + 1: -1].**strip**(),  
 **time\_to\_iso**(output\_git\_log.**split**("**\n**")[5][12:].**strip**()))  
 else:  
 a = output\_git\_log.**split**("**\n**")[3][8:].**strip**()  
 res = (line[0].**strip**(), line[1].**strip**(), line[2].**strip**(),  
 a[:a.**index**("<")].**strip**(),  
 a[a.**index**("<") + 1: -1].**strip**(),  
 **time\_to\_iso**(output\_git\_log.**split**("**\n**")[4][12:].**strip**()))  
  
 except subprocess.CalledProcessError as exc:  
 print(f"Status : **{**exc.returncode**}**, **\n{**exc.output**}**")  
 raise Exception(f"Status : **{**exc.returncode**}**, **\n{**exc.output**}**")  
 return res  
  
  
def **projects\_platforms**(line: str) -> Tuple[str, str]:  
 *"""  
  
 """* if "FileGenSG200-ubuntu18" in line:  
 return "ubuntu18", "SG200"  
 elif "FileGenSG200-win10" in line:  
 return "win10", "SG200"  
 elif "Model\_RTR-u18-field" in line:  
 return "u18-field", "Model\_RTR"  
 elif "Model\_RTR-u20-field" in line:  
 return "u20-field", "Model\_RTR"  
 branches = ("testing", "master", "release\_21", "release\_22", "release\_20", "HEAD")  
  
 line\_split = line.**split**("-")  
 platform = line\_split[0]  
 k = 2  
 try:  
 while True:  
 if any(list(map(lambda x: x in line\_split[k], branches))):  
 break  
 k += 1  
 except Exception:  
 print(line)  
 raise Exception  
  
 project = "-".**join**(line\_split[1:k])  
 return project, platform  
  
  
def **time\_to\_iso**(wrong\_date: str) -> str:  
 *"""  
 from "Wed Oct 12 14:25:09 2022 +0300"  
 to "2022-10-12 14:25:09"  
 """* commit\_date = str(datetime.**strptime**(" ".**join**(wrong\_date.**split**(" ")[:-1]), "%a %b %d %H:%M:%S %Y"))  
 return commit\_date  
  
  
def **firmware\_parser**(arg\_line: str):  
 redundant = ('Emulator-AnivaTest', 'Emulator-Baul', 'Emulator-Bo4ka\_Zynq', 'Emulator-Ecm\_zynq\_dummy',  
 'Emulator-Emulator', 'Emulator-EmulatorRR', 'Emulator-EmulatorRTR', 'Emulator-ImitatorP18-rx',  
 'Emulator-ImitatorP18-tx', 'Emulator-PictureZynq', 'Emulator-PictureZynq\_Test',  
 'Emulator-Seno\_Storage\_Polygon\_RLS\_I', 'Emulator-Seno\_Storage\_Polygon\_RLS\_R',  
 'Emulator-Seno\_Storage\_Polygon\_RLS\_SI', 'Emulator-SmallRtrF1400\_02\_18', 'Emulator-ZynqSar',  
 'Emulator-flea\_ks', 'Emulator-flea\_ks\_biu', 'Emulator-hay\_storage\_handheld', 'Emulator-o10-azurit',  
 'Emulator-o10\_azurit', 'Emulator-o10\_flock', 'Emulator-o10\_flock\_k4\_rev2', 'Emulator-o10ks-0818',  
 'Emulator-o25\_field', 'Emulator-o25ks-0818\_UavField', 'Emulator-tits-chrome', 'Pocket-BIU\_ublox\_nav',  
 'Pocket-Saturn\_Barrel', 'Pocket-azest', 'Pocket-dummy', 'Pocket-full\_switch',  
 'Pocket-o10-0826\_RtkNavi', 'Pocket-o10-um', 'Pocket-shevron', 'Pocket-skb\_1ch', 'Pocket-uav-lab',  
 'PocketKS-AnivaTest', 'PocketKS-Bo4ka\_STC\_Saturn\_SKR\_Test', 'PocketKS-Emulator', 'PocketKS-EmulatorRR',  
 'PocketKS-EmulatorRTR', 'PocketKS-ErhanRotatorWooden', 'PocketKS-Erhan\_Ch\_Sprut',  
 'PocketKS-SmallRtrF1400\_02\_18', 'PocketKS-dummy', 'PocketKS-flea\_ks\_biu', 'PocketKS-k4\_rev2',  
 'PocketKS-pocketK2', 'Zynq-Bo4ka\_Zynq', 'Zynq-Ecm\_zynq\_dummy', 'Zynq-Mosquito\_Elint\_Modern',  
 'Zynq-PictureZynq\_Test', 'Zynq-Seno\_Storage\_Polygon\_RLS\_SI\_G', 'Zynq-Seno\_Storage\_Polygon\_RLS\_SI\_P',  
 'Zynq-ZynqSar', 'ats-lin', 'ats-win', 'Zynq-Adaptation', 'Zynq-ErhanChe', 'noarm')  
  
 if any(list(map(lambda x: x in arg\_line, redundant))):  
 return  
  
 with open("./parsed", "a") as f2:  
 file = arg\_line.**strip**()  
 f\_b\_h = **branch\_hash**(file)  
 os.**chdir**("../commonpocket")  
 f\_b\_h\_ana\_aem\_t = **author**(f\_b\_h)  
 os.**chdir**("../firmwares")  
 pr\_pl = **projects\_platforms**(file)  
 f2.**write**(";;".**join**([\*f\_b\_h\_ana\_aem\_t, \*pr\_pl]) + "**\n**")  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 with open("./all\_firmwares", "r") as f:  
 for firmware in f:  
 **firmware\_parser**(firmware)

## ПРИЛОЖЕНИЕ Б «ТЕКСТ ПРОЦЕДУР ЗАПОЛНЕНИЯ И ОБНОВЛЕНИЯ (plpgsql)»

DROP PROCEDURE IF EXISTS insert\_payload;  
DROP PROCEDURE IF EXISTS update\_payload;  
DROP PROCEDURE IF EXISTS insert\_firmware;  
DROP PROCEDURE IF EXISTS insert\_platform;  
DROP PROCEDURE IF EXISTS insert\_project;  
DROP PROCEDURE IF EXISTS insert\_author;  
DROP PROCEDURE IF EXISTS insert\_branch;  
DROP PROCEDURE IF EXISTS insert\_project\_platform;  
DROP PROCEDURE IF EXISTS insert\_project\_firmware;  
  
  
/\* добавить нагрузку \*/  
*--DROP PROCEDURE IF EXISTS insert\_payload;*CREATE PROCEDURE insert\_payload(platform\_name\_ text, payload\_name\_ text, project\_name\_ text, decimal\_id\_ text = '', zavod\_id\_ text = '',  
 characteristics\_path\_ text = '', rtr\_ bool = FALSE, rsa\_ bool = FALSE, rc\_ bool = FALSE,  
 rtc\_ bool = FALSE, rp\_ bool = FALSE, rlo\_ bool = FALSE, fc\_ bool = FALSE, description\_ text = '')  
LANGUAGE 'plpgsql'  
AS $BODY$  
BEGIN  
 INSERT INTO payload (payload\_name, decimal\_id, zavod\_id, platform\_id, project\_id, characteristics\_path, rtr, rsa, rc, rtc, rp, rlo, fc, description) VALUES (  
 payload\_name\_,  
 decimal\_id\_,  
 zavod\_id\_,  
 (SELECT platform\_id FROM platform WHERE platform\_name = platform\_name\_),  
 (SELECT project\_id FROM project WHERE project\_name = project\_name\_),  
 characteristics\_path\_,  
 rtr\_,  
 rsa\_,  
 rc\_,  
 rtc\_,  
 rp\_,  
 rlo\_,  
 fc\_,  
 description\_) ON CONFLICT (payload\_name, project\_id, platform\_id) DO NOTHING;  
EXCEPTION  
 WHEN others THEN  
 RAISE NOTICE 'SQLSTATE: %', SQLSTATE;  
 ROLLBACK;  
 RAISE;  
END;  
$BODY$;  
  
/\* по id нагрузки изменить ее данные \*/  
*--DROP PROCEDURE IF EXISTS update\_payload;*CREATE PROCEDURE update\_payload(platform\_name\_ text, project\_name\_ text, decimal\_id\_ text = '', zavod\_id\_ text = '',  
 characteristics\_path\_ text = '', rtr\_ bool = FALSE, rsa\_ bool = FALSE, rc\_ bool = FALSE,  
 rtc\_ bool = FALSE, rp\_ bool = FALSE, rlo\_ bool = FALSE, fc\_ bool = FALSE, description\_ text = '')  
LANGUAGE 'plpgsql'  
AS $BODY$  
BEGIN  
 UPDATE payload SET  
 decimal\_id = decimal\_id\_,  
 zavod\_id = zavod\_id\_,  
 platform\_id = (SELECT platform\_id FROM platform WHERE platform\_name = platform\_name\_),  
 project\_id = (SELECT project\_id FROM project WHERE project\_name = project\_name\_),  
 characteristics\_path = characteristics\_path\_,  
 rtr = rtr\_,  
 rsa = rsa\_,  
 rc = rc\_,  
 rtc = rtc\_,  
 rp = rp\_,  
 rlo = rlo\_,  
 fc = fc\_,  
 description = description\_  
 WHERE payload\_name = payload\_name\_  
 AND project\_id = project\_id\_  
 AND platform\_id = platform\_id\_;  
EXCEPTION  
 WHEN others THEN  
 RAISE NOTICE 'SQLSTATE: %', SQLSTATE;  
 ROLLBACK;  
*-- RAISE;*END;  
$BODY$;  
  
  
  
/\* добавить прошивку \*/  
*--DROP PROCEDURE IF EXISTS insert\_firmware;*CREATE PROCEDURE insert\_firmware(platform\_ text, branch\_ text, author\_ text, email\_ text,  
 build\_date\_ TIMESTAMP(5), hash\_ text, path\_ text, tests\_ bool = FALSE)  
LANGUAGE 'plpgsql'  
AS $BODY$  
BEGIN  
 INSERT INTO firmware (platform\_id, branch\_id, author\_id, tests\_passed, build\_date, hash, path) VALUES (  
 (SELECT platform\_id FROM platform WHERE platform\_name = platform\_),  
 (SELECT branch\_id FROM branch WHERE branch\_name = branch\_),  
 (SELECT author\_id FROM author WHERE author\_name = author\_ AND email = email\_),  
 tests\_,  
 build\_date\_,  
 hash\_,  
 path\_)  
 ON CONFLICT (path) DO NOTHING;  
EXCEPTION  
 WHEN others THEN  
 RAISE NOTICE 'SQLSTATE: %', SQLSTATE;  
 RAISE;  
 ROLLBACK;  
END;  
$BODY$;  
  
  
/\* добавить платформу \*/  
*--DROP PROCEDURE IF EXISTS insert\_platform;*CREATE PROCEDURE insert\_platform(name text)  
LANGUAGE 'plpgsql'  
AS $BODY$  
BEGIN  
 INSERT INTO platform (platform\_name) VALUES (name)  
 ON CONFLICT (platform\_name) DO NOTHING;  
EXCEPTION  
 WHEN others THEN  
 RAISE NOTICE 'SQLSTATE: %', SQLSTATE;  
*-- RAISE;* ROLLBACK;  
END;  
$BODY$;  
  
  
/\* добавить проект \*/  
*--DROP PROCEDURE IF EXISTS insert\_project;*CREATE PROCEDURE insert\_project(name text)  
LANGUAGE 'plpgsql'  
AS $BODY$  
BEGIN  
 INSERT INTO project (project\_name) VALUES (name)  
 ON CONFLICT (project\_name) DO NOTHING;  
EXCEPTION  
 WHEN others THEN  
 RAISE NOTICE 'SQLSTATE: %', SQLSTATE;  
*-- RAISE;* ROLLBACK;  
END;  
$BODY$;  
  
/\* добавить автора \*/  
*--DROP PROCEDURE IF EXISTS insert\_author;*CREATE PROCEDURE insert\_author(name text, author\_email text)  
LANGUAGE 'plpgsql'  
AS $BODY$  
BEGIN  
 INSERT INTO author (author\_name, email) VALUES (name, author\_email)  
 ON CONFLICT (author\_name, email) DO NOTHING;  
EXCEPTION  
 WHEN others THEN  
 RAISE NOTICE 'SQLSTATE: %', SQLSTATE;  
*-- RAISE;* ROLLBACK;  
END;  
$BODY$;  
  
/\* добавить ветку \*/  
*--DROP PROCEDURE IF EXISTS insert\_branch;*CREATE PROCEDURE insert\_branch(name text)  
LANGUAGE 'plpgsql'  
AS $BODY$  
BEGIN  
 INSERT INTO branch (branch\_name) VALUES (name)  
 ON CONFLICT (branch\_name) DO NOTHING;  
EXCEPTION  
 WHEN others THEN  
 RAISE NOTICE 'SQLSTATE: %', SQLSTATE;  
*-- RAISE;* ROLLBACK;  
END;  
$BODY$;  
  
/\* добавить связь проекта с платформой \*/  
*--DROP PROCEDURE IF EXISTS insert\_project\_platform;*CREATE PROCEDURE insert\_project\_platform(project\_ text, platform\_ text)  
LANGUAGE 'plpgsql'  
AS $BODY$  
BEGIN  
 INSERT INTO platform\_project (platform\_id, project\_id) VALUES (  
 (SELECT platform\_id FROM platform WHERE platform\_name = platform\_),  
 (SELECT project\_id FROM project WHERE project\_name = project\_))  
 ON CONFLICT (platform\_id, project\_id) DO NOTHING;  
EXCEPTION  
 WHEN others THEN  
 RAISE NOTICE 'SQLSTATE: %', SQLSTATE;  
*-- RAISE;* ROLLBACK;  
END;  
$BODY$;  
  
  
/\* добавить связь проекта с прошивкой \*/  
*--DROP PROCEDURE IF EXISTS insert\_project\_firmware;*CREATE PROCEDURE insert\_project\_firmware(project\_ text, firmware\_ text)  
LANGUAGE 'plpgsql'  
AS $BODY$  
BEGIN  
 INSERT INTO firmware\_project (firmware\_id, project\_id) VALUES (  
 (SELECT firmware\_id FROM firmware WHERE path = firmware\_),  
 (SELECT project\_id FROM project WHERE project\_name = project\_))  
 ON CONFLICT (firmware\_id, project\_id) DO NOTHING;  
EXCEPTION  
 WHEN others THEN  
 RAISE NOTICE 'SQLSTATE: %', SQLSTATE;  
*-- RAISE;* ROLLBACK;  
END;  
$BODY$;

## ПРИЛОЖЕНИЕ В «ТЕКСТ ПРОГРАММЫ ЗАПОЛНЕНИЯ БД (Python)»

import time  
import psycopg2  
  
conn = psycopg2.**connect**("dbname=firmwares user=buildbot password=qwe")  
rule = {  
 'Emulator-Aniva': {'Анива': ['РТК']},  
 'Emulator-BarrierT\_RLR': {'Барьер\_Т': ['РЛР']},  
 'Emulator-Bo4ka\_STC\_Saturn\_SKR': {'СК-Р': ['МПО РТС версия 2']},  
 'Emulator-Bo4ka\_STC\_Saturn\_SKR\_FastDF': {'СК-Р': ['МПО РТС версия 3']},  
 'Emulator-Demagog\_DRTR': {'Демагог': ['ДРТР']},  
 'Emulator-Erhan': {'Ирхан': ['РПРЛС ПрмПрд "Радиан"']},  
 'Emulator-ErhanRotator': {'Ирхан': ['РПРЛС ПрмПрд "Радиан" на ПУ']},  
 'Emulator-Erhan\_v2\_5': {'Ирхан': ['РПРЛС Прд "Радиан", Прм "Сатурн"']},  
 'Emulator-HayStorage\_RTR': {'Сенохранилище': ['РТР']},  
 'Emulator-NeutralityZynq': {'Нейтралитет': ['РПРЛС']},  
 'Emulator-PacerZynq\_RTR500': {'Иноходец': ['ЦН РТР500']},  
 'Emulator-Platforma\_RTR20': {'Платформа': ['РТР40']},  
 'Emulator-Platforma\_RTR400': {'Платформа': ['РТР400']},  
 'Emulator-Polyanka-Ground': {'Полянка': ['РПРЛС']},  
 'Emulator-Polyanka-UAV': {'Полянка': ['БЛА ЦН РПРЛС']},  
 'Emulator-Stamp': {'Штемпель': ['БРЛС']},  
 'Emulator-Tits\_Demagog': {'Демагог': ['РТКВ']},  
 'SG200-ubuntu18': {'SG200': ['Сборка ubuntu18']},  
 'SG200-win10': {'SG200': ['Сборка win10']},  
 'Model\_RTR-u18-field': {'Поле БЛА': ['Тренажер РТР ubuntu18']},  
 'Model\_RTR-u20-field': {'Поле БЛА': ['Тренажер РТР ubuntu20']},  
 'Pocket-BirderRx': {'Птицелов': ['РЛО']},  
 'Pocket-BirderTx': {'Птицелов': ['Активный РЛО']},  
 'Pocket-Bo4ka\_Kursk\_SKR': {'СК-Р': ['МПО РТС версия 1']},  
 'Pocket-Bo4ka\_RadiotehnikaV': {'Радиотехника-В': ['МПО РТС версия 1']},  
 'Pocket-Bo4ka\_RadiotehnikaV\_Ver2': {'Радиотехника-В': ['МПО РТС версия 2']},  
 'Pocket-Bo4ka\_STC\_WeekRx': {'СК-Р': ['МПО РТС версия 2']},  
 'Pocket-Bo4ka\_Zagoryanka': {'Загорянка': ['МПО РТС'], 'Форпост': ['МПО РТС']},  
 'Pocket-DTR\_RTR': {'ДТР-РТР': ['МПО РТС']},  
 'Pocket-Moskit\_RD\_Orlan30': {'Москит': ['ЦН РПРЛС версия 1']},  
 'Pocket-Moskit\_RTR\_Orlan30': {'Москит': ['ЦН РТР версия 1']},  
 'Pocket-Orion\_1ch\_RR': {'Орион': ['ЦН РР слуховой контроль']},  
 'Pocket-Orion\_RR': {'Орион': ['ЦН РР пеленгатор']},  
 'Pocket-Orion\_RTR': {'Орион': ['ЦН РТР']},  
 'Pocket-Tits\_DF\_Retsenzent': {'Рецензент': ['2-МП РТС']},  
 'Pocket-Tits\_Dubrava': {'Дубрава': ['2-МП РТС']},  
 'Pocket-Tits\_SKR': {'СК-Р': ['2-МП РТС версия 1']},  
 'Pocket-Tits\_SKR\_White': {'СК-Р': ['2-МП РТС версия 2']},  
 'Pocket-Tits\_survey': {'СК-Р': ['1-МП РТС Ил-20']},  
 'Pocket-chrome': {'Хром ОКР': ['ЦН РР 30-3000']},  
 'Pocket-chrome\_rlo': {'Хром ОКР': ['РЛО']},  
 'Pocket-flea': {'СК-Р': ['Блоха РР 30-3000']},  
 'Pocket-flea\_rpu6': {'Блоха': ['РР версия РПУ6 can']},  
 'Pocket-moskit\_ais': {'Москит': ['ЦН АИС']},  
 'Pocket-moskit\_rlo': {'Москит': ['ЦН РЛО']},  
 'Pocket-nested\_box': {'Демагог': ['КСМСИ'], 'СК-Р': ['КСМСИ']},  
 'Pocket-o10-0826': {'Радиотехника-В': ['ЦН РТР-1 версия 2'],  
 'ТОРН-8ПМК': ['ЦН РТР'],  
 'Хром': ['ЦН РРТР']},  
 'Pocket-o10-0826\_Dubrava': {'Дубрава': ['ЦН РТР']},  
 'Pocket-o10-0826\_OrlanUM': {'Орлан-УМ': ['ЦН РТР']},  
 'Pocket-o10-2640': {'Радиотехника-В': ['ЦН РТР-2 версия 2']},  
 'Pocket-o10-973\_Dubrava\_RD': {'Дубрава': ['ЦН РПРЛС']},  
 'Pocket-o10\_otmetka': {'Отметка': ['ЦН поиска маячков']},  
 'Pocket-o25-2640\_UavField': {'Поле-БЛА ОКР': ['ЦН РТР-2 200МГц']},  
 'Pocket-orlan10\_dubrava': {'Дубрава ОКР': ['ЦН РР Орлан-10 с конвертером']},  
 'Pocket-orlan10\_geos\_navi': {'МФК': ['ЦН РР Орлан-10 30-3000 без ВТО']},  
 'Pocket-orlan10\_stc\_navi': {'МФК': ['ЦН РР Орлан-10 30-3000'],  
 'ТОРН-8ПМК': ['ЦН РР Орлан-10 30-3000']},  
 'Pocket-orlan10\_um\_jetrec': {'Орлан УМ': ['ЦН РР 30-3000']},  
 'Pocket-orlan30\_moskit': {'Москит 501 ОКР': ['ЦН РР 30-3000']},  
 'Pocket-orlan30\_moskit\_fhss': {'Москит 501 ОКР': ['ЦН ППРЧ']},  
 'Pocket-skb\_1ch\_40MHz': {'СКБ': ['Блоха РР РПУ can СКБ']},  
 'Pocket-skb\_1ch\_40MHz\_converter': {'СКБ': ['Блоха РР РПУ can СКБ с '  
 'конвертером СТЦ 1000-6500МГц']},  
 'Pocket-skb\_2ch\_40MHz': {'СКБ': ['РР 2к РПУ can СКБ']},  
 'PocketKS-Aniva': {'Анива': ['РТК']},  
 'PocketKS-Baul': {'Баул': ['РПРЛС']},  
 'PocketKS-Bo4ka\_STC\_Saturn\_SKR': {'СК-Р': ['МПО РТС версия 2']},  
 'PocketKS-Bo4ka\_STC\_Saturn\_SKR\_FastDF': {'Демагог': ['МПО РТС'],  
 'СК-Р': ['МПО РТС версия 3']},  
 'PocketKS-Erhan': {'Ирхан': ['РПРЛС ПрмПрд "Радиан"']},  
 'PocketKS-ErhanRotator': {'Ирхан': ['РПРЛС ПрмПрд "Радиан" на ПУ']},  
 'PocketKS-ErhanRotatorL': {'Ирхан': ['РПРЛС Прд "Радиан", Прм "Сатурн" на '  
 'ПУ']},  
 'PocketKS-Erhan\_v2\_5': {'Ирхан': ['РПРЛС Прд "Радиан", Прм "Сатурн"']},  
 'PocketKS-HayStorage\_RTR': {'Сенохранилище': ['РТР']},  
 'PocketKS-ImitatorP18-rx': {'Имитатор-П18': ['РТР']},  
 'PocketKS-ImitatorP18-tx': {'Имитатор-П18': ['РПРЛС']},  
 'PocketKS-Jupiter': {'Юпитер': ['РТР']},  
 'PocketKS-NeutralityZynq': {'Нейтралитет': ['РПРЛС']},  
 'PocketKS-Platforma\_RTR20': {'Платформа': ['РТР40']},  
 'PocketKS-Platforma\_RTR400': {'Платформа': ['РТР400']},  
 'PocketKS-Polyanka-Ground': {'Полянка': ['РПРЛС наземный']},  
 'PocketKS-Polyanka-UAV': {'Полянка': ['ЦН РПРЛС']},  
 'PocketKS-Sprut': {'Спрут': ['РТР']},  
 'PocketKS-Sprut\_Plus': {'Спрут': ['РТР расширенный диапазон частот']},  
 'PocketKS-Stamp': {'Штемпель': ['БРЛС']},  
 'PocketKS-Tits\_Demagog': {'Демагог': ['РТКВ']},  
 'PocketKS-flea\_ks': {'Блоха': ['РР МРПУ']},  
 'PocketKS-hay\_storage\_handheld': {'Сенохранилище ОКР': ['КАПС ПОЛИГОН-Р-Н-СС '  
 '№14.1 РР+GSM']},  
 'PocketKS-o10-azurit-6250Hz': {'Азурит ОКР': ['ЦН РР МРПУ 1к с фильтром '  
 '6250Гц']},  
 'PocketKS-o10\_azurit': {'Азурит ОКР': ['ЦН РР МРПУ 1к']},  
 'PocketKS-o10\_flock': {'Стая Кейстон': ['ЦН РР МРПУ 3к пеленгатор версия 1']},  
 'PocketKS-o10\_flock\_k4\_rev2': {'Стая Кейстон': ['ЦН РР МРПУ 3к пеленгатор '  
 'версия 2']},  
 'PocketKS-o10ks-0818': {'Чебурашка': ['ЦН РПРЛС']},  
 'PocketKS-o25\_field': {'Поле БЛА': ['ЦН РК']},  
 'PocketKS-o25ks-0818\_UavField': {'Поле БЛА': ['ЦН РТР-1']},  
 'PocketKS-tits-chrome-rev2': {'Хром': ['ЦН РР 1к c фильтром 12500 Гц и полосой 100 '  
 'МГц (РДМ)']},  
 'PocketKS-tits-chrome': {'Хром': ['хрен знает']},  
 'Zynq-Adaptation\_RTR2': {'Адаптация': ['ЦН РТР-2']},  
 'Zynq-BarrierT\_RLR': {'Барьер-Т': ['РЛР']},  
 'Zynq-Demagog\_DRTR': {'Демагог': ['ДРТР']},  
 'Zynq-Moskit\_RP\_Zynq': {'Москит': ['ЦН РПРЛС версия 2']},  
 'Zynq-Mosquito\_Elint\_M': {'Москит': ['РТР версия 2']},  
 'Zynq-PacerZynq\_RTR500': {'Иноходец': ['ЦН РТР500']},  
 'Zynq-PictureZynq': {'Картина': ['ЦН РПРЛС']},  
 'Zynq-Seno\_Storage\_Polygon\_RLS\_I': {'Сенохранилище': ['КАПС-17']},  
 'Zynq-Seno\_Storage\_Polygon\_RLS\_R': {'Сенохранилище': ['КАПС-07']},  
 'Zynq-Seno\_Storage\_Polygon\_RLS\_SI': {'Сенохранилище': ['КАПС-06']},  
 'Zynq-Sich': {'Сыч': ['РПРЛС']},  
 'Zynq-o30-mt501-comint': {'Москит 501 серия': ['ЦН РР']}}  
cur = conn.**cursor**()  
  
  
with open("./parsed", "r") as f:  
 i = 0  
 start = time.**time**()  
 try:  
 for line in f:  
 line\_split = line.**strip**().**split**(";;")  
 i += 1  
 if i % 1000 == 0:  
 print(i, f"**{**int((time.**time**() - start) // 60)**}**:**{**int((time.**time**() - start) % 60)**}**")  
 filename = line\_split[0]  
 branch = line\_split[1]  
 commit\_hash = line\_split[2]  
 author\_name = line\_split[3]  
 author\_email = line\_split[4]  
 commit\_date = line\_split[5]  
 project = line\_split[6]  
 platform = line\_split[7]  
  
 cur.**execute**(f"CALL insert\_author('**{**author\_name**}**', '**{**author\_email**}**');")  
 conn.**commit**()  
 cur.**execute**(f"CALL insert\_platform('**{**platform**}**');")  
 conn.**commit**()  
 project\_lists = []  
 cur.**execute**(f"CALL insert\_branch('**{**branch**}**');")  
 conn.**commit**()  
 platform\_project = platform + "-" + project  
 cur.**execute**(f"CALL insert\_firmware('**{**platform**}**', '**{**branch**}**', '**{**author\_name**}**', '**{**author\_email**}**', '**{**commit\_date**}**', '**{**commit\_hash**}**', '**{**filename**}**');")  
 conn.**commit**()  
 try:  
 rule[platform\_project]  
 except KeyError:  
 print(line)  
 raise KeyError  
 for project\_ru in rule[platform\_project]:  
 project\_lists.**append**(project\_ru)  
 cur.**execute**(f"CALL insert\_project('**{**project\_ru**}**');")  
 conn.**commit**()  
  
 for payload\_ru in rule[platform\_project][project\_ru]:  
 cur.**execute**(f"CALL insert\_payload('**{**platform**}**', '**{**payload\_ru**}**', '**{**project\_ru**}**');")  
 conn.**commit**()  
  
 cur.**execute**(f"CALL insert\_project\_firmware('**{**project\_ru**}**', '**{**filename**}**');")  
 conn.**commit**()  
  
 cur.**execute**(f"CALL insert\_project\_platform('**{**project\_ru**}**', '**{**platform**}**');")  
 conn.**commit**()  
  
 except Exception as exc:  
 raise Exception(f"Status : **{**exc**}**")  
 finally:  
 cur.**close**()  
 conn.**close**()