Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Кафедра «Информатика»
кафедра

ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ № 3

Использование локальной базы данных
Тема

Преподаватель _____ И. В. Ковалев
Подпись, дата — Инициалы, Фамилия

Студент КИ19-17/1Б, №031939174 _____ А. К. Никитин
Номер группы, зачетной книжки — Подпись, дата — Инициалы, Фамилия

Красноярск 2022

1 Задачи

Подключить локальную базу данных.

2 Ход работы

Использовалась готовая локальная база данных покемонов расширения .sqlite. База данных содержит в себе полный список покемонов, их описание, типы, поколение, приемы и множество другой полезной информации.

Также была создана таблица, хранящая отмеченных пользователем покемонов.

3 Листинг программы

Ниже представлены листинги со всем элементами системы.

Листинг 1 – Класс базы данных

```
class PokemonDatabase {

    static int get _version => 1;

    static Future<Database> getDatabase() async {
        return await getDatabasesPath().then((path) async {
            String androidDbPath = join(path, 'veekun-pokedex.sqlite');
            if (FileSystemEntity.typeSync(path) == FileSystemEntityType.notFound)
{
                String srcDbPath = join('databases', 'veekun-pokedex.sqlite');
                ByteData data = await rootBundle.load(srcDbPath);
                List<int>  bytes  =  data.buffer.asUint8List(data.offsetInBytes,
data.lengthInBytes);
                await File(androidDbPath).writeAsBytes(bytes);
            }
            return openDatabase(androidDbPath, version: _version, onCreate: (db,
version) =>
            db.execute('CREATE TABLE favorites (id INTEGER PRIMARY KEY)'));
        });
    }
}
Future<Database> database = PokemonDatabase.getDatabase();
```

Листинг 2 – Класс модели покемона

```
class Pokemon {
  int id;
  String name;
  Gen gen;
```

```
  Type type1;
  Type type2;
  bool isFavorite;

  Pokemon(this.id, this.name, this.gen, this.type1, this.type2, [this.isFavorite
= false]);

  Map<String, dynamic> toMap() {
    return {
      'id': id,
      'name': name,
      'gen': genToString(gen),
      'type1': typeToString(type1),
      'type2': typeToString(type2),
    };
  }
}

Future<List<Pokemon>> getPokemons() async {
  final db = await database;

  db.execute(
      'CREATE TABLE IF NOT EXISTS favorites AS SELECT species_id as id, count(case
when 0 then 0 end) as is_favorite FROM pokemon GROUP BY species_id;');

  String sqlQuery = '''
    WITH RECURSIVE pokemon_gen_data AS (
        SELECT
        species_id as id,
        identifier as name,
        min(pokemon_form_generations.generation_id) as gen
        FROM pokemon
        JOIN pokemon_form_generations ON pokemon_form_generations.pokemon_form_id
= pokemon.species_id
        WHERE is_default = 1
        GROUP BY name
    ), pokemon_types_gen_data AS (
        SELECT
        pokemon_gen_data.id as id,
        name,
        gen,
        group_concat(types.identifier) as types
```

```
        FROM pokemon_gen_data
        JOIN pokemon_types ON pokemon_gen_data.id = pokemon_types.pokemon_id
        JOIN types ON types.id = pokemon_types.type_id
        GROUP BY name
    )
    SELECT
    id,
    gen,
    name,
    types,
    is_favorite
    FROM pokemon_types_gen_data
    NATURAL JOIN favorites
    ORDER BY is_favorite DESC, id ASC;
    ''';

  final List<Map<String, dynamic>> maps = await db.rawQuery(sqlQuery);
  return List.generate(maps.length, (i) {
    List<String> splitResult = maps[i]['types'].split(',');
    String type1 = splitResult.first;
    String type2 = splitResult.last;

    return Pokemon(
      maps[i]['id'],
      capitalize(maps[i]['name']),
      numberToGen(maps[i]['gen']),
      stringToType(type1),
      stringToType(type2),
      maps[i]['is_favorite'] == 0 ? false : true,
    );
  });
}

void updateFavorite(Pokemon pokemon) async {
  final db = await database;

  db.update('favorites', {'is_favorite': pokemon.isFavorite ? 1 : 0}, where: "id
= ${pokemon.id}");
}
```