Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа интеллектуальных систем и суперкомпьютерных технологий

Базы данных

Отчёт по лабораторным работам

Работу выполнили:
Назаров Д. Ю.
Петров В. Д.
Телепова М. П.
Группа:
3530901/80201
Преподаватель:
Мяснов А. В.

Санкт-Петербург
2021

Содержание

1. Лабораторная работа 1

1.1. Цели работы

Познакомиться с основами проектирования схемы БД, способами организации данных в SQL-БД.

1.2. Программа работы

1. Создание проекта для работы в `GitLab`.
2. Выбор задания (предметной области), описание набора данных и требований к хранимым данным в свободном формате в `wiki` своего проекта в `GitLab`.
3. Формирование в свободном формате (предпочтительно в виде графической схемы) схемы БД, соответствующей заданию. Должно получиться около 21 таблицы (7 таблиц на человека).
4. Согласование с преподавателем схемы БД. Обоснование принятых решений и соответствия требованиям выбранного задания.
5. Выкладывание схемы БД в свой проект в `GitLab`.
6. Демонстрация результатов преподавателю.

1.3. Выполнение работы

1.3.1. Создание проекта

Рис. 1.1. Список репозиториев на `GitLab`

Было создано несколько репозиториев на `GitLab`:

- Проект для SQL_DDL и SQL_DML скриптов;
- Проект для генератора тестовых данных;
- Проект для тестового приложения трех уровней транзакций;
- Проект для реализации rest-сервера для курсового проекта;

- Проект для веб-приложения;
- Проект для мобильного приложения Андроид;
- Проект для хранения отчетности о выполнении работы и ссылок на версии схем проекта.

1.3.2. Предметная область

В качестве задания для курсовой работы и предметной области для лабораторных работ было выбрано персональное расписание для студента Политеха с возможностью добавления персональных дедлайнов.

1.3.3. Описание таблиц

- `Faculty` - таблица институтов
  - `faculty_id` - id института
  - `faculty_name` - имя института
- `Department` - таблица кафедр
  - `department_id` - id кафедры
  - `department_name` - имя кафедры
  - `faculty_id` - внешний ключ на институт, к которому относится кафедра
- `Subject` - таблица предметов
  - `subject_id` - id предмета
  - `subject_name` - название предмета
  - `spec_id` - внешний ключ на направление
- `Specialization` - таблица направлений обучения
  - `spec_id` - id направления
  - `spec_name` - название направления
- `Lesson` - таблица занятий
  - `lesson_id` - id занятия
  - `lesson_date` - дата проведения занятия
  - `lesson_place` - место проведения занятия
  - `lesson_type_id` - внешний ключ на вид занятия

  - `subject_id` - внешний ключ на предмет
- `Lesson_type` - таблица видов занятий
  - `lesson_type_id` - id вида занятий
  - `lesson_type_name` - название вида занятий
- `Teacher` - таблица преподавателей
  - `teacher_id` - id преподавателя
  - `teacher_name` - имя преподавателя
- `Lesson_teacher` - таблица связи между занятиями и преподавателями
  - `teacher_id` - внешний ключ на преподавателя
  - `lesson_id` - внешний ключ на занятие
- `Group` - таблица студенческих групп
  - `group_id` - id группы
  - `group_name` - имя группы
  - `spec_id` - внешний ключ на направление
- `Student` - таблица студентов
  - `student_id` - id студента
  - `student_name` - имя студента
  - `group_id` - id группы
- `Deadline`
  - `deadline_id` - id дедлайна
  - `deadline_title` - заголовок дедлайна
  - `deadline_description` - описание дедлайна
  - `deadline_notification_time` - время оповещения о дедлайне
  - `deadline_date` - дата дедлайна
  - `deadline_type` - тип дедлайна
  - `lesson_id` - внешний ключ на занятие
  - `media_url` - ссылка на дополнительные материалы

1.3.4. Изначальная схема базы данных

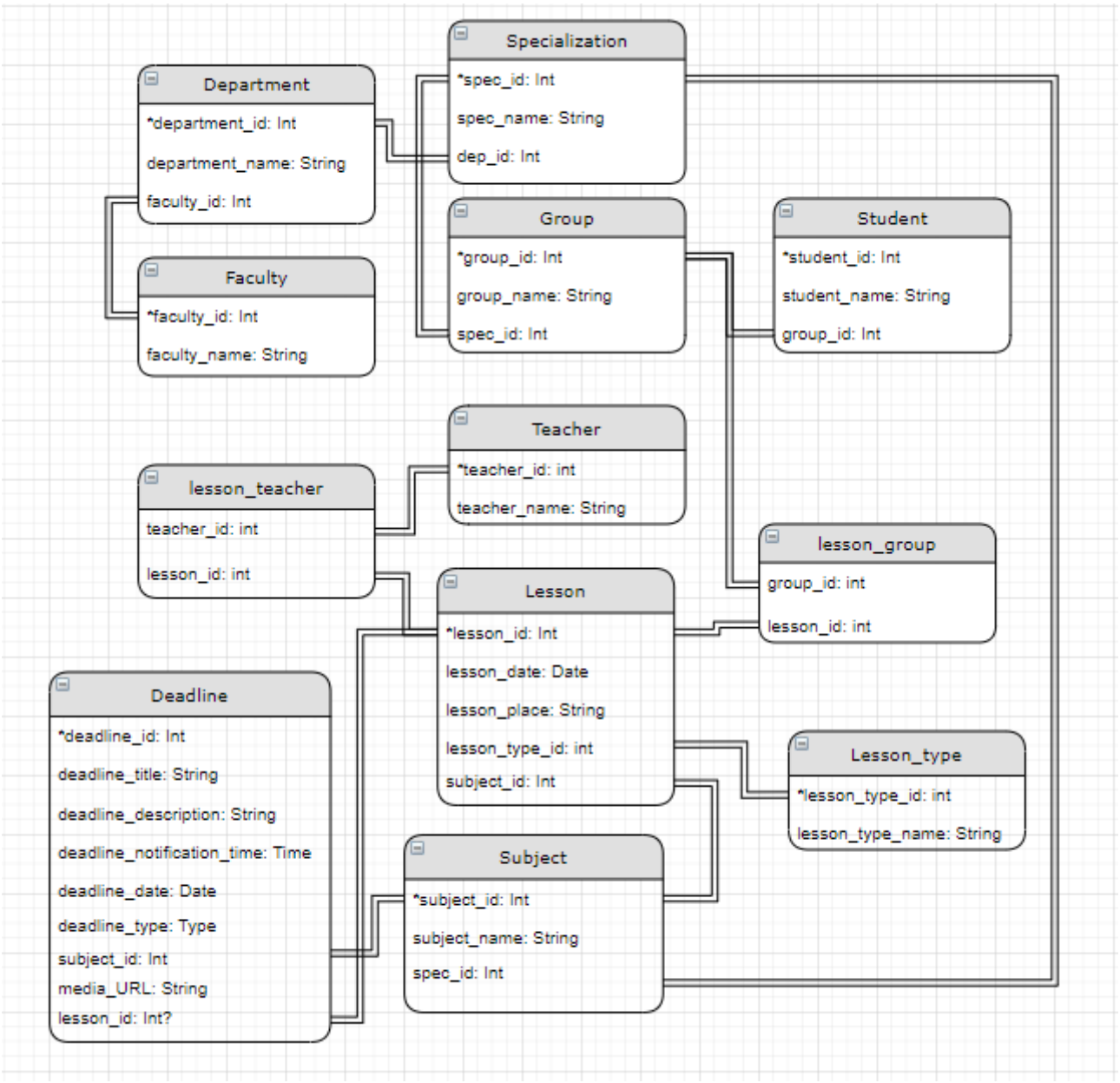


Рис. 1.2. Схема базы данных

1.3.5. Скрипт создания базы данных

Скрипт создает саму базу данных с помощью команды `CREATE`.

```
CREATE SCHEMA IF NOT EXISTS schema_schedule;
```

После чего создает таблицы, удаляя ранее созданные, если они существуют, вместе с связанными с ней таблицами. У каждого поля указываетсятип данных, у первичного ключа указывается, что он `PRIMARY KEY`. Для связей между полями используется конструкция `REFERENCES`. Ниже представлен код создания таблицы `Lesson`:

```
DROP TABLE IF EXISTS LESSON CASCADE;
CREATE TABLE LESSON (
    ID SERIAL PRIMARY KEY,
    LESSON_DATE DATE NOT NULL,
    PLACE TEXT,
    TYPE INTEGER REFERENCES LESSON_TYPE (ID) ON DELETE CASCADE
     ↪  NOT NULL,
    SUBJECT_ID INTEGER REFERENCES SUBJECT (ID) ON DELETE
     ↪  CASCADE NOT NULL
);
```

Полный код скрипта можно найти по ссылке.

1.3.6. Изменение базы данных

Список изменений базы данных:

1. Изменить тип связи между направлениями подготовки и высшими школами таким образом, чтобы каждая ВШ могла выпускать студентов по нескольким направлениям, а каждое направление могло быть реализовано в нескольких ВШ. При изменении структуры БД данные не должны быть потеряны.

2. Реализовать учет профилей подготовки.

3. Реализовать учет учебных планов: профилю соответствует набор дисциплин, практик, ГИА. В каждом элементе могут быть разные виды занятий с разной нагрузкой.

4. Реализовать учет успеваемости студентов по предметам. Нужно учитывать текущую успеваемость: контрольные, коллоквиумы, рефераты и промежуточную: КП, КР, зачеты, экзамены.

Ниже представлены добавленные таблицы, согласно необходимым изменениям для реализации:

- `Department_specialization` - таблица связи между кафедрой и направлением
  - `dep_spec_id` - id связи
  - `department_id` - внешний ключ на кафедру
  - `spec_id` - внешний ключ на направление
- `Profile` - таблица профиля специализации

  - profile_id - id профиля
  - spec_id - внешний ключ на направление обучения
  - profile_name - название профиля специализации
- Subject_type - таблица видов практики по предмету
  - subject_type_id - id вида практики по предмету
  - subject_type_name - название вида практики по предмету
- Profile_subject - связь между профилем специализации и предметом
  - profile_subject_id - id связи
  - subject_id - внешний ключ на предмет
  - profile_id - внешний ключ на профиль специализации
- Lesson_group - таблица связи между занятиями и группами
  - group_id - внешний ключ на группу
  - lesson_id - внешний ключ на занятие
- Semester- - таблица семестров
  - semester_id - id семестра
  - semester_level - номер семестра
  - student_id - внешний ключ на студента
- Progress - таблица для учета успеваемости студентов по семестрам
  - progress_id - id учета успеваемости
  - semester_id - внешний ключ на семестр
  - subject_id - внешний ключ на предмет
- Task - таблица видов контроля успеваемости
  - task_id - id вида контроля
  - task_type_id - внешний ключ на название вида контроля
  - task_result - полученная оценка
  - progress_id - внешний ключ на учет успеваемости
- Task_type - таблица названий видов контроля
  - task_type_id - id названия вида контроля
  - task_type_name - название вида контроля

1.3.7. Окончательная схема базы данных

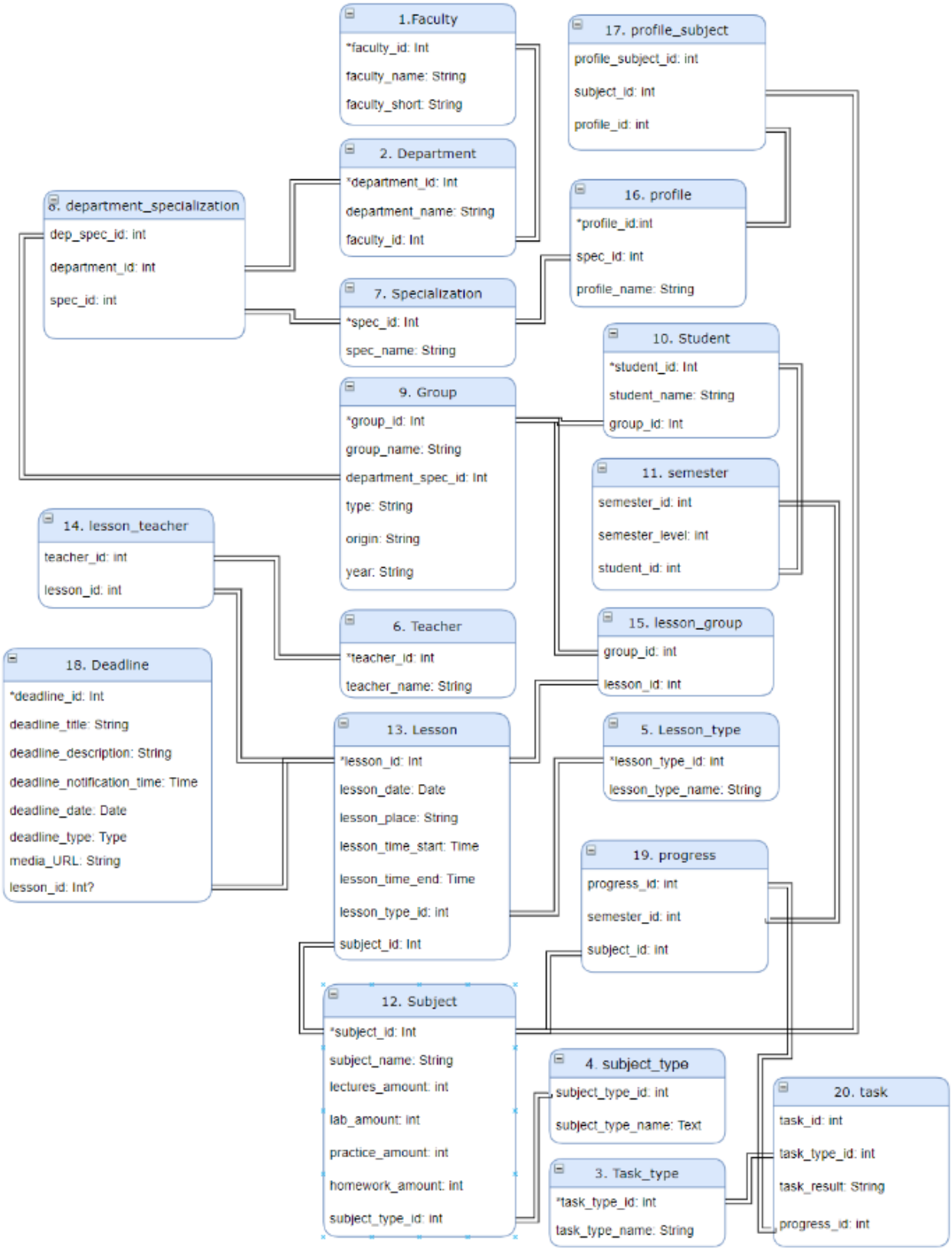


Рис. 1.3. Измененная схема базы данных

1.3.8. Скрипт для внесения изменений

Для добавление полей в уже существующие таблицы мы воспользовались командой `ALTER TABLE`:

```
ALTER TABLE subject ADD IF NOT EXISTS SUBJECT_TYPE_ID INTEGER
 ↪  REFERENCES SUBJECT_TYPE (ID) ON DELETE CASCADE NOT NULL
 ↪  DEFAULT 0;
```

Новые таблицы были добавлены с помощью уже использованной команды `CREATE TABLE`:

```
DROP TABLE IF EXISTS PROFILE CASCADE;
CREATE TABLE PROFILE (
    ID SERIAL PRIMARY KEY,
    SPECIALIZATION_ID INTEGER REFERENCES SPECIALIZATION (ID) ON
     ↪  DELETE CASCADE NOT NULL,
    NAME TEXT NOT NULL
);
```

Полный код скрипта можно найти по ссылке.

1.4. Выводы

В данной лабораторной работе был создан проект на `GitLab`, выбрана предметная область базы данных, описан набор данных, сформирована схема БД в графическом формате, а также был изучен язык SQL, с помощью которого были разработаны скрипты для создания таблиц согласно схеме и изменения таблиц согласно заданию преподавателя.

2. Лабораторная работа 2

2.1. Цели работы

Сформировать набор данных, позволяющий производить операции на реальных объемах данных.

2.2. Программа работы

1. Реализация в виде программы параметризуемого генератора, который позволит сформировать набор связанных данных в каждой таблице.
2. Частные требования к генератору, набору данных и результирующему набору данных:
    - количество записей в справочных таблицах должно соответствовать ограничениям предметной области
    - количество записей в таблицах, хранящих информацию об объектах или субъектах должно быть параметром генерации
    - значения для внешних ключей необходимо брать из связанных таблиц
    - сохранение уже имеющихся данных в базе данных

2.3. Выполнение работы

Для выполнения данной лабораторной работы было написано приложение на языке `Kotlin` с использованием библиотеки `jdbc` для исполнения `sql` запросов внутри функций на языке `Kotlin` и получения возвращаемых из базы данных значений. Для генерации осмысленных имен студентов и преподавателей используется библиотека `Faker`, для обращения к `API` - библиотека `okhttp3`, а для конвертации данных из `API` в `Java` классы используется библиотека `Gson`. В функции `main` используется `measure_time` для того, чтобы посчитать время, за которое мы заполняем данными таблицу, а также функция `insertAllTables`, которая и будет исполнять весь код генератора.

```
@ExperimentalTime
fun main() {
    val connection = createConnection() ?: return

    val duration = measureTime {
        insertAllTables(connection)
    }
```

```
    logger.log("====================================")
    logger.log(duration)
    logger.close()
}
```

Внутри функции `insertAllTables` вызываются методы для заполнения каждой таблицы в строгом порядке - начиная с таблицы `faculty` и заканчивая таблицей `tasks` для того, чтобы брать значения для внешних ключей из связанных таблиц, для каждого метода используется подсчет затраченного времени.

```
@ExperimentalTime
fun insertAllTables(connection: Connection) {

    var duration = measureTime { insertFaculty(connection) }
    logger.log("[FACULTY] DURATION = $duration\n====")

    duration = measureTime { insertDepartment(connection) }
    logger.log("[DEPARTMENT] DURATION = $duration\n====")

    duration = measureTime { insertTaskType(connection) }
    logger.log("[TASK_TYPE] DURATION = $duration\n====")

    duration = measureTime { insertSubjectType(connection) }
    logger.log("[SUBJECT_TYPE] DURATION = $duration\n====")

    duration = measureTime { insertLessonType(connection) }
    logger.log("[LESSON_TYPE] DURATION = $duration\n====")

    duration = measureTime { insertTeacher(connection) }
    logger.log("[TEACHER] DURATION = $duration\n====")

    duration = measureTime {
     ↪  insertGroupSpecializationAndGroupSpecialization(connection)
     ↪  }
    logger.log("[GROUP SPECIALIZATION GROUP_SPECIALIZATION]
     ↪  DURATION = $duration\n====")

    duration = measureTime { insertStudent(connection) }
    logger.log("[STUDENTS] DURATION = $duration\n====")

    duration = measureTime { insertSemester(connection) }
```

```
    logger.log("[SEMESTERS] DURATION = $duration\n====")

    duration = measureTime {
     ↪  insertSubjectLessonLessonTeacherLessonGroup(connection)
     ↪  }
    logger.log("[SUBJECT LESSON LESSON_TEACHER LESSON_GROUP]
     ↪  DURATION = $duration\n====")

    duration = measureTime { insertProfile(connection) }
    logger.log("[PROFILE] DURATION = $duration\n====")

    duration = measureTime { insertProfileSubject(connection) }
    logger.log("[PROFILE_SUBJECT] DURATION = $duration\n====")

    duration = measureTime { insertDeadline(connection) }
    logger.log("[DEADLINE] DURATION = $duration\n====")

    duration = measureTime { insertProgress(connection) }
    logger.log("[PROGRESS] DURATION = $duration\n====")

    duration = measureTime { insertTask(connection) }
    logger.log("[TASK] DURATION = $duration\n====")
}
```

Рассмотрим использование API для генерации правдивых данных на примере метода insertDepartments, который добавляет в таблицу department новые записи, попутно связывая их с правильными записями из таблицы faculty, так, например, кафедра ВШИСиСТ будет правильно связана с институтом ИКНТ.

```
fun insertDepartment(connection: Connection) {

    val departmentBatch = connection.createStatement()
    val facultyResultSet =
     ↪  connection.createStatement().executeQuery("SELECT id,
     ↪  name FROM schema_schedule.faculty")

    val divisionURL = StringBuilder(ApiURL.DEPARTMENT)
    val divisionLength = divisionURL.length
    var apiPage = 1

    do {
        divisionURL.append("$apiPage")
```

```
        val response = getApiResponse(divisionURL.toString(),
        ↪  true)
        divisionURL.setLength(divisionLength)
        apiPage++

        val divisionContainer = gson.fromJson(response,
        ↪  DivisionContainer::class.java)

        divisionFullList.addAll(divisionContainer.divisionList)

    } while (divisionContainer.next != null)

    for (department in divisionFullList) {
        val departmentParent =
        ↪  facultyApiIdToNameMap[department.parentId] ?:
        ↪  continue

        if (department.closeDate == null) {
            val facultyID = facultyNameToIdMap[departmentParent]
            departmentBatch.addBatch(
                """
                INSERT INTO schema_schedule.department (name,
                ↪  faculty_id)
                SELECT '${department.name}', $facultyID
                WHERE NOT EXISTS(
                SELECT 1
                FROM schema_schedule.department
                WHERE name = '${department.name}'
                AND faculty_id = $facultyID)""".trimMargin()
            )
        }
    }

    executeBatch(departmentBatch, tableName)
}
```

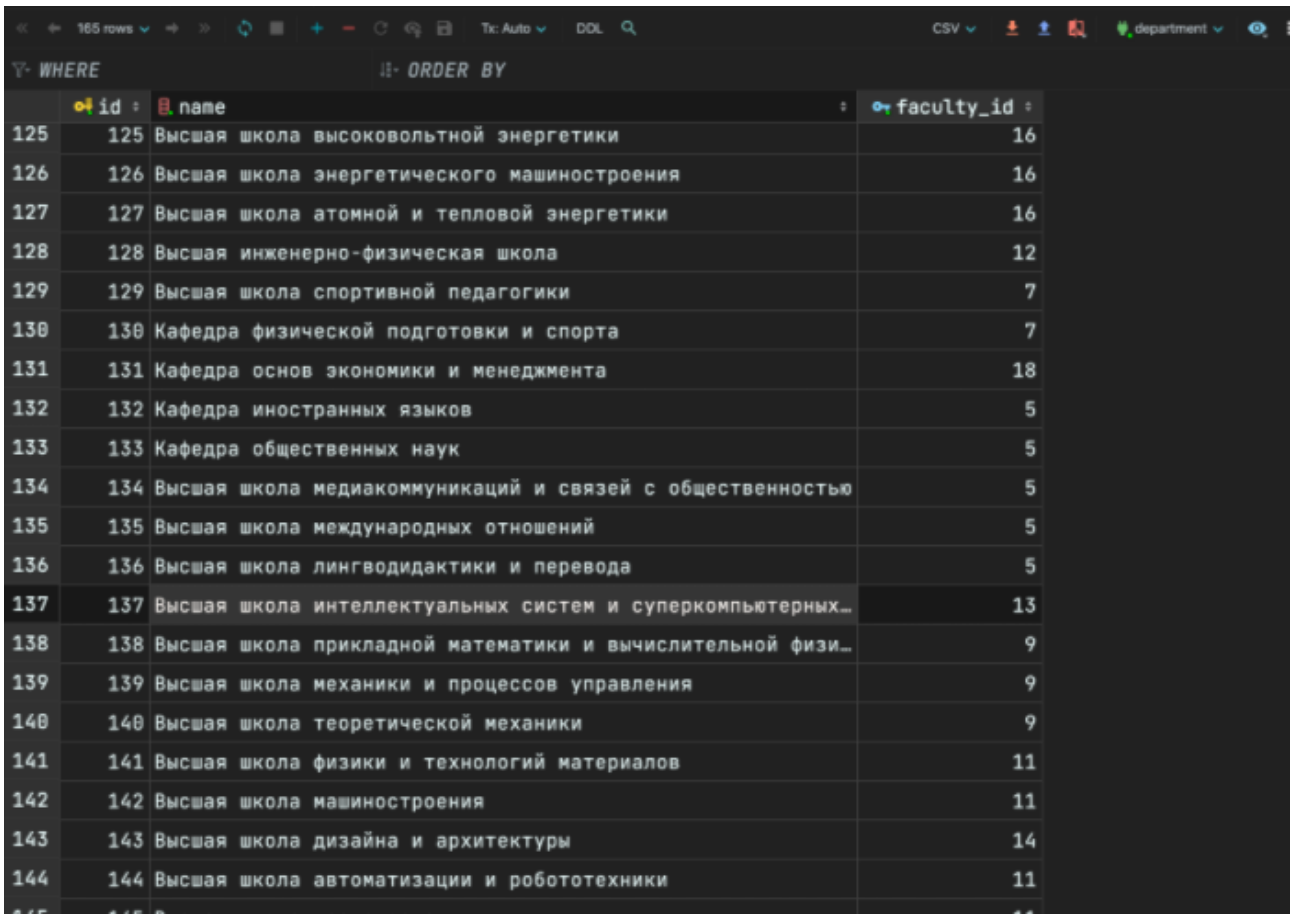
В данном методе мы используем `SELECT` для получения всех `id` из таблицы `faculty`, после чего обращаемся к `API` и сопоставляем полученные значения с тем, что у нас в таблице `faculty`, после чего добавляем новые записи в `department`, используя сопоставленные значения для `facultyID`.

Теперь взглянем на значение `id` у ИКНТ из таблицы `faculty` и убедимся, что оно соответствует значению `faculty_id` у ВШИСиСТ из таблицы `department`.

	id	name	short	api_id
1	1	Институт ядерной энергетики (филиал ФГАОУ ВО СПбПУ) г. Сосновый…	ИЯЭ	120
2	2	Институт кибербезопасности и защиты информации	ИКиЗИ	122
3	3	Университетский политехнический колледж	УПКР	117
4	4	Высшая школа международных образовательных программ	ВШМОП	116
5	5	Гуманитарный институт	ГИ	101
6	6	Институт биомедицинских систем и биотехнологий	ИБСиБ	119
7	7	Институт физической культуры, спорта и туризма	ИФКСиТ	121
8	8	Институт передовых производственных технологий	ИППТ	111
9	9	Институт прикладной математики и механики	ИПММ	99
10	10	Институт дополнительного образования	ИДО	114
11	11	Институт машиностроения, материалов и транспорта	ИММиТ	94
12	12	Институт физики, нанотехнологий и телекоммуникаций	ИФНиТ	98
13	13	Институт компьютерных наук и технологий	ИКНТ	95
14	14	Инженерно-строительный институт	ИСИ	92
15	15	Высшая школа техносферной безопасности	ВШТБ	96
16	16	Институт энергетики	ИЭ	93
17	17	Высшая школа биотехнологии и пищевых технологий	ВШБТиПТ	112
18	18	Институт промышленного менеджмента, экономики и торговли	ИПМЭиТ	100

Рис. 2.1. Часть таблицы faculty

Как мы можем видеть, `id` у ИКНТ равен 13. Теперь обратимся к таблице `department` и убедимся, что в ней у ВШИСиСТ `faculty_id` равен 13.



WHERE ORDER BY

	id	name	faculty_id
125	125	Высшая школа высоковольтной энергетики	16
126	126	Высшая школа энергетического машиностроения	16
127	127	Высшая школа атомной и тепловой энергетики	16
128	128	Высшая инженерно-физическая школа	12
129	129	Высшая школа спортивной педагогики	7
130	130	Кафедра физической подготовки и спорта	7
131	131	Кафедра основ экономики и менеджмента	18
132	132	Кафедра иностранных языков	5
133	133	Кафедра общественных наук	5
134	134	Высшая школа медиакоммуникаций и связей с общественностью	5
135	135	Высшая школа международных отношений	5
136	136	Высшая школа лингводидактики и перевода	5
137	137	Высшая школа интеллектуальных систем и суперкомпьютерных…	13
138	138	Высшая школа прикладной математики и вычислительной физи…	9
139	139	Высшая школа механики и процессов управления	9
140	140	Высшая школа теоретической механики	9
141	141	Высшая школа физики и технологий материалов	11
142	142	Высшая школа машиностроения	11
143	143	Высшая школа дизайна и архитектуры	14
144	144	Высшая школа автоматизации и робототехники	11

Рис. 2.2. Часть таблицы department

Как и ожидалось, после генерации таблицы `department` все связанные `id` из таблицы `faculty` были правильно взяты для значений внешних ключей.

Теперь обратимся к методу `insertStudents`, чтобы продемонстрировать работу справочника для генерации различных имен.

```
fun insertStudent(connection: Connection) {

    val studentBatch = connection.createStatement()

    for (i in 0..GENERATED_STUDENT_AMOUNT) {

        val fakeName = faker.name
        val studentName = "${fakeName.firstName()}
         ↪  ${fakeName.lastName()}".replace("\'", "")

        val groupIds = fastIdSelecting(connection, "\"GROUP\"")
        val groupId = groupIds.random()

        studentBatch.addBatch(
            """
```

```
            INSERT INTO schema_schedule.student(name, group_id)
                SELECT '$studentName', $groupId
                WHERE NOT EXISTS(
                SELECT 1
                FROM schema_schedule.student
                WHERE name = '$studentName'
                AND group_id = $groupId
                )"""
        )
    }

    executeBatch(studentBatch, tableName)
}
```

Здесь мы генерируем случайное имя и фамилию для студента, получаем рандомный `id` и вставляем полученного студента в таблицу. Посмотрим на получившиеся значения.

	id	name	group_id	password	role
336	336	Brianne Bogisich	1193	<null>	STUDENT
337	337	Evita Schinner	1399	<null>	STUDENT
338	338	Roland Reilly	317	<null>	STUDENT
339	339	Jerald Schimmel	195	<null>	STUDENT
340	340	Chet Pfeffer	1256	<null>	STUDENT
341	341	Trent MacGyver	425	<null>	STUDENT
342	342	Marty Powlowski	1235	<null>	STUDENT
343	343	Jonah Schmeler	972	<null>	STUDENT
344	344	Orval Ryan	1427	<null>	STUDENT
345	345	Ute Rogahn	794	<null>	STUDENT
346	346	Imelda Christiansen	1386	<null>	STUDENT
347	347	Ping Emard	1409	<null>	STUDENT
348	348	Art Okuneva	547	<null>	STUDENT
349	349	Lorrine Gorczany	514	<null>	STUDENT
350	350	Keila Bernhard	80	<null>	STUDENT
351	351	Lavera Hackett	1010	<null>	STUDENT
352	352	Dominque Cormier	1153	<null>	STUDENT
353	353	Leslie Fahey	607	<null>	STUDENT
354	354	Darwin Rodriguez	1655	<null>	STUDENT
355	355	Melba Pouros	183	<null>	STUDENT
356	356	Raphael Flatley	472	<null>	STUDENT

Рис. 2.3. Часть таблицы студентов

Как мы можем видеть, имен из справочника достаточно, чтобы данные из таблицы не повторяли друг друга. Теперь посмотрим на различные таблицы после генерации и посмотрим, что данные сгенерированы правильно. Так, таблица `lesson` использует `API`.

	id	lesson_date	place	type	subject_id	time_start	time_end
1774	1774	2021-05-17	1120	1	498	16:00	17:40
1775	1775	2021-05-21	1121	6	499	10:00	13:40
1776	1776	2021-05-17	255	0	4	10:00	11:40
1777	1777	2021-05-17	255	0	4	12:00	13:40
1778	1778	2021-05-17	255	0	4	14:00	15:40
1779	1779	2021-05-17	Дистанционно	2	500	10:00	11:40
1780	1780	2021-05-17	Дистанционно	0	500	16:00	17:40
1781	1781	2021-05-19	Дистанционно	6	98	16:00	17:40
1782	1782	2021-05-20	513	6	311	10:00	11:45
1783	1783	2021-05-20	513	6	311	12:00	13:45
1784	1784	2021-05-17	Лаб. компрессоростроения	0	130	10:00	11:40
1785	1785	2021-05-17	Лаб. компрессоростроения	0	130	12:00	13:40
1786	1786	2021-05-18	Лаб. компрессоростроения	6	242	10:00	13:00
1787	1787	2021-05-18	Лаб. компрессоростроения	6	130	13:00	16:00
1788	1788	2021-05-19	Лаб. компрессоростроения	6	130	10:00	13:00
1789	1789	2021-05-20	Лаб. компрессоростроения	6	130	10:00	13:00
1790	1790	2021-05-20	Лаб. компрессоростроения	6	130	13:00	16:00
1791	1791	2021-05-21	Лаб. компрессоростроения	6	242	10:00	14:00
1792	1792	2021-05-17	111	0	501	10:00	11:40
1793	1793	2021-05-17	111	1	501	12:00	13:40
1794	1794	2021-05-17	111	0	501	16:00	17:40

Рис. 2.4. Часть таблицы занятий

Также `API` использует таблица `subject`.

	id	name	lec...	lab_...	practic...	hom...	subject_type_id
1	1	Экономический анализ	22	38	33	39	4
2	2	Промышленные ТЭС	25	21	36	40	4
3	3	Бизнес-планирование и анализ инвести…	34	22	29	35	5
4	4	Высшая математика	22	35	40	32	5
5	5	Релейная защита электроэнергетически…	31	21	40	35	7
6	6	Общая электротехника	38	24	22	21	7
7	7	Теория автоматического регулирования…	24	26	40	32	9
8	8	Теоретическая механика	22	39	27	22	6
9	9	Метрология	27	22	39	35	6
10	10	Практикум по вычислительной математи…	27	22	39	35	6
11	11	Практикум по теории вероятностей и м…	27	22	39	35	6
12	12	Семинар по УТС	23	30	24	33	7
13	13	Теория и практика педагогического об…	26	32	30	26	8
14	14	Качественные и количественные методы…	26	32	30	26	8
15	15	Управленческий учет	30	23	33	27	5
16	16	Основы компьютерного проектирования …	29	40	36	36	7
17	17	Лабораторный практикум по системам м…	40	27	25	39	1
18	18	Лабораторный практикум по схемотехни…	40	27	25	39	1
19	19	Экономика	32	33	22	38	6
20	20	Макроэкономика	23	26	37	28	6
21	21	Основы проектной деятельности	23	26	37	28	6

Рис. 2.5. Часть таблицы предметов

В свою очередь таблица `deadline` использует только получившиеся значения `id` из таблиц `lesson` и `subject`.

	id	title	description	no...	deadl...	type	subject_id
40982	40982	Jog in the morning	Will jog every morning to prepare for …	22:52:43	2021-05-05	Goal	636
40983	40983	Math class postponed	Today math class is postponed due to t…	02:50:49	2021-05-24	Notifica…	1129
40984	40984	Math class postponed	Today math class is postponed due to t…	15:49:51	2021-05-20	Notifica…	1024
40985	40985	Teacher birthday today	Today is our teacher birthday, we need…	06:49:41	2021-05-06	Reminder	1466
40986	40986	Teacher birthday today	Today is our teacher birthday, we need…	09:36:50	2021-05-10	Reminder	559
40987	40987	Teacher birthday today	Today is our teacher birthday, we need…	16:32:46	2021-05-06	Reminder	782
40988	40988	Jog in the morning	Will jog every morning to prepare for …	17:09:37	2021-05-16	Goal	795
40989	40989	Teacher birthday today	Today is our teacher birthday, we need…	21:13:30	2021-05-02	Reminder	1632
40990	40990	Teacher birthday today	Today is our teacher birthday, we need…	22:51:56	2021-05-05	Reminder	1630
40991	40991	Math class postponed	Today math class is postponed due to t…	06:27:33	2021-05-19	Notifica…	1343
40992	40992	OpenEdu test deadline	Should do the OpenEdu test before it e…	10:49:55	2021-05-13	Deadline	1320
40993	40993	Do the test before sunday	Need to do the test before sunday beca…	01:02:29	2021-05-21	Task	1480
40994	40994	OpenEdu test deadline	Should do the OpenEdu test before it expires	3:44	2021-05-01	Deadline	468
40995	40995	Do the test before sunday	Need to do the test before sunday beca…	22:13:37	2021-05-29	Task	779
40996	40996	Databases lab deadline	Need to send report for database lab b…	05:33:22	2021-05-13	Deadline	1095
40997	40997	ModelSim lab deadline	Send a report and files for project of…	17:32:25	2021-05-14	Deadline	234
40998	40998	Databases lab deadline	Need to send report for database lab b…	22:22:08	2021-05-08	Deadline	819
40999	40999	ModelSim lab deadline	Send a report and files for project of…	03:06:50	2021-05-22	Deadline	894
41000	41000	Do the test before sunday	Need to do the test before sunday beca…	22:53:08	2021-05-04	Task	1402
41001	41001	English test deadline	Should do the english class test on SP…	15:06:12	2021-05-28	Deadline	343

Рис. 2.6. Часть таблицы дедлайнов

Нерассмотренные методы для генерации таблиц аналогичны рассмотренным по структуре и функциональности. Полный код для проекта доступен по ссылке.

2.4. Выводы

В данной лабораторной работы мы создали приложение для генерации тестовых данных в нашу таблицу, протестировали его работу и убедились в правдивости полученных данных. Для внешних ключей были задействованы связанные таблицы, обращение к базе данных производилось с помощью библиотеки `jdbc`, а обращение к `API` - с помощью `okhttp3`.

3. Лабораторная работа 3

3.1. Цели работы

Познакомиться с языком создания запросов управления данными SQL-DML.

3.2. Программа работы

1. Изучение SQL-DML.
2. Выполнение всех запросов из списка стандартных запросов. Демонстрация результатов преподавателю.
3. Получение у преподавателя и реализация SQL-запросов в соответствии с индивидуальным заданием. Демонстрация результатов преподавателю.
4. Сохранение в БД выполненных запросов `SELECT` в виде представлений, запросов `INSERT`, `UPDATE` или `DELETE` -- в виде ХП. Выкладывание скрипта в `GitLab`.

3.3. Выполнение работы

3.3.1. Стандартные запросы

```
CREATE SCHEMA IF NOT EXISTS schema_schedule;
SET search_path = "schema_schedule";


/*
Общее задание 1.

Сделайте выборку всех данных из каждой таблицы
*/
SELECT *
FROM faculty;

SELECT *
FROM department;

SELECT *
FROM task_type;

SELECT *
```

```
FROM subject_type;

SELECT *
FROM lesson_type;

SELECT *
FROM teacher;

SELECT *
FROM specialization;

SELECT *
FROM department_specialization;

SELECT *
FROM "GROUP";

SELECT *
FROM student;

SELECT *
FROM semester;

SELECT *
FROM subject;

SELECT *
FROM lesson;

SELECT *
FROM lesson_teacher;

SELECT *
FROM lesson_group;

SELECT *
FROM profile;

SELECT *
FROM profile_subject;

SELECT *
```

```sql
FROM progress;

SELECT *
FROM task;

SELECT *
FROM deadline;

/*
Общее задание 2.

Сделайте выборку данных из одной таблицы при нескольких
 ↪  условиях, с использованием логических операций, LIKE,
 ↪  BETWEEN, IN
*/
SELECT profile.id, profile.name
FROM profile
WHERE profile.name LIKE '09.03.01%';

SELECT subject.lab_amount, subject.name
FROM subject
WHERE subject.lab_amount BETWEEN 27 AND 29;

SELECT l_subject_name__good_task_results__task_type.name,
task_type.name,
l_subject_name__good_task_results__task_type.result
FROM (SELECT subject.name,
 ↪  l_subjec_id__task_type_result.result,
 ↪  l_subjec_id__task_type_result.type_id
FROM (SELECT progress.subject_id, task.type_id, task.result
FROM progress
INNER JOIN task
ON progress.id = task.progress_id
) AS l_subjec_id__task_type_result
INNER JOIN subject
ON l_subjec_id__task_type_result.subject_id = subject.id
WHERE l_subjec_id__task_type_result.result IN ('Отлично',
 ↪  'Хорошо')
) AS l_subject_name__good_task_results__task_type
INNER JOIN task_type
ON l_subject_name__good_task_results__task_type.type_id =
 ↪  task_type.id;
```

```sql
/*
Общее задание 3.

Создайте в запросе вычисляемое поле.
*/
SELECT name, (lectures_amount + lab_amount + practice_amount +
 ↪  homework_amount) AS work_amount
FROM subject;


/*
Общее задание 4.

Сделайте выборку всех данных с сортировкой по нескольким полям.
*/
SELECT *
FROM "GROUP"
ORDER BY "GROUP".level, "GROUP".origin;

/*
Общее задание 5.

Создайте запрос, вычисляющий несколько совокупных характеристик
 ↪  таблиц.
*/
SELECT COUNT(*) AS number_of_subjects,
(AVG(lectures_amount) + AVG(lab_amount) + AVG(practice_amount)
 ↪  + AVG(homework_amount)) AS avg_work_amount,
MIN(homework_amount) AS minimum_homework,
MAX(lectures_amount) AS maximum_lectures
FROM subject;


/*
Общее задание 6.

Сделайте выборку данных из связанных таблиц (не менее двух
 ↪  примеров).
*/
SELECT student.name, "GROUP".name
```

```
FROM student
LEFT OUTER JOIN "GROUP"
ON student.group_id = "GROUP".id;


SELECT subject.name,
subject.lectures_amount,
subject.lab_amount,
subject.practice_amount,
subject.homework_amount,
subject_type.name
FROM subject
INNER JOIN subject_type
ON subject.subject_type_id = subject_type.id;

/*
Общее задание 7.

Создайте запрос, рассчитывающий совокупную характеристику с
 ↪  использованием группировки, наложите ограничение на
 ↪  результат группировки.
*/
SELECT faculty.name, COUNT(*) AS number_of_departments
FROM department
INNER JOIN faculty ON faculty.id = department.faculty_id
GROUP BY faculty.name
HAVING COUNT(*) < 8
ORDER BY faculty.name;

/*
Общее задание 8.

Придумайте и реализуйте пример использования вложенного запроса.
*/
SELECT "GROUP".name, l_lesson__group_id.lesson_date,
 ↪  l_lesson__group_id.time_start, l_lesson__group_id.time_end
FROM (SELECT lesson.*, lesson_group."GROUP" AS group_id
FROM lesson
LEFT JOIN lesson_group
ON lesson.id = lesson_group.lesson
) AS l_lesson__group_id
INNER JOIN "GROUP"
```

```
ON l_lesson__group_id.group_id = "GROUP".id
ORDER BY "GROUP".name, l_lesson__group_id.lesson_date,
 ↪  l_lesson__group_id.time_start;

/*
Общее задание 9.

С помощью оператора INSERT добавьте в каждую таблицу по одной
 ↪  записи
*/
DO
$$
DECLARE
faculty_id      faculty.id%TYPE;
department_id   department.id%TYPE;
task_type_id    task_type.id%TYPE;
subject_type_id subject_type.id%TYPE;
lesson_type_id  lesson_type.id%TYPE;
teacher_id      teacher.id%TYPE;
spec_id         specialization.id%TYPE;
dep_spec_id     department_specialization.id%TYPE;
group_id        "GROUP".id%TYPE;
student_id      student.id%TYPE;
semester_id     semester.id%TYPE;
subject_id      subject.id%TYPE;
lesson_id       lesson.id%TYPE;
profile_id      profile.id%TYPE;
progress_id     progress.id%TYPE;
BEGIN

INSERT INTO faculty(name, short)
VALUES ('Институт программистов', 'ИП')
RETURNING id INTO faculty_id;

INSERT INTO department(name, faculty_id)
VALUES ('Кафедра программистов', faculty_id)
RETURNING id INTO department_id;

INSERT INTO task_type(id, name)
VALUES ((CASE
WHEN (SELECT MAX(id) FROM task_type) IS NOT NULL THEN (SELECT
 ↪  MAX(id) FROM task_type) + 1
```

```
ELSE 1 END),
'Задание')
RETURNING id INTO task_type_id;

INSERT INTO subject_type(id, name)
VALUES (DEFAULT, 'Внеучебная Практика')
RETURNING id INTO subject_type_id;

INSERT INTO lesson_type(id, name)
VALUES ((CASE
WHEN (SELECT MAX(id) FROM lesson_type) IS NOT NULL THEN (SELECT
 ↪  MAX(id) FROM lesson_type) + 1
ELSE 1 END),
'Занятие')
RETURNING id INTO lesson_type_id;

INSERT INTO teacher(name, api_id)
VALUES ('Петров Виталий Дмитриевич', '96024')
RETURNING id INTO teacher_id;

INSERT INTO specialization(name)
VALUES ('01.01.01 Образование')
RETURNING id INTO spec_id;

INSERT INTO department_specialization(department_id,
 ↪  specialization_id)
VALUES (department_id, spec_id)
RETURNING id INTO dep_spec_id;

INSERT INTO "GROUP"(name, api_id, level, type, origin, year,
 ↪  department_specialization_id)
VALUES ('3530901/80201_Fake', '96024', 3, 'common', 0, 2020,
 ↪  dep_spec_id)
RETURNING id INTO group_id;

INSERT INTO student(name, group_id)
VALUES ('Петров Виталий Дмитриевич', group_id)
RETURNING id INTO student_id;

INSERT INTO semester(student_id, level)
VALUES (student_id, 6)
RETURNING id INTO semester_id;
```

```
INSERT INTO subject(id, name, lectures_amount, lab_amount,
 ↪  practice_amount, homework_amount, subject_type_id)
VALUES ((CASE WHEN (SELECT MAX(id) FROM subject) IS NOT NULL
 ↪  THEN (SELECT MAX(id) FROM subject) + 1 ELSE 1 END),
'Предмет', 100, 100, 100, 100, subject_type_id)
RETURNING id INTO subject_id;

INSERT INTO lesson(lesson_date, place, type, subject_id,
 ↪  time_start, time_end)
VALUES ('2021-05-26', 'Дистанционно', lesson_type_id,
 ↪  subject_id, '10:00', '20:00')
RETURNING id INTO lesson_id;

INSERT INTO lesson_teacher(lesson, teacher)
VALUES (lesson_id, teacher_id);

INSERT INTO lesson_group(lesson, "GROUP")
VALUES (lesson_id, group_id);

INSERT INTO profile(specialization_id, name)
VALUES (spec_id, '01.01.01_01 Программисты')
RETURNING id INTO profile_id;

INSERT INTO profile_subject(profile, subject)
VALUES (profile_id, subject_id);

INSERT INTO deadline(title, description, notification_time,
 ↪  deadline_date, type, subject_id, lesson_id,
media_url)
VALUES ('Работа', 'Сделать работу', '10:00:00', '2021-05-26',
 ↪  'Deadline', subject_id, lesson_id, 'vk.com');

INSERT INTO progress(semester_id, subject_id)
VALUES (semester_id, subject_id)
RETURNING id INTO progress_id;

INSERT INTO task(progress_id, type_id, result)
VALUES (progress_id, task_type_id, 'Отчислен');
END
$$;
```

```
/*
Общее задание 10.

С помощью оператора UPDATE измените значения нескольких полей у
 ↪  всех записей, отвечающих заданному условию.
*/
UPDATE deadline
SET deadline.description = CONCAT('!!Сделать срочно!! ',
 ↪  deadline.description),
deadline.media_URL    =
 ↪  'https://dl.spbstu.ru/mod/assign/view.php?id=123456'
WHERE deadline.id = 2;

/*
Общее задание 11.

С помощью оператора DELETE удалите запись, имеющую максимальное
 ↪  (минимальное) значение некоторой совокупной характеристики.
*/
DELETE
FROM deadline
WHERE deadline.deadline_date = (SELECT
 ↪  MAX(deadline.deadline_date)
FROM deadline);

/*
Общее задание 12.

С помощью оператора DELETE удалите записи в главной таблице, на
 ↪  которые не ссылается подчиненная таблица (используя
 ↪  вложенный запрос).
*/
DELETE
FROM faculty
WHERE faculty.id NOT IN (SELECT department.faculty_id
FROM department)
```

3.3.2. Индивидуальные задания

Оринальный файл можно посмотреть по этой ссылке.

1. Вывести 5 наиболее сложных институтов. Сложность института считать

как среднее значение произведения количества дисциплин в семестре на количество заданий по каждой дисциплине.

```
SELECT l_faculty_name__multiplied_tasks_and_semesters.facu
 ↪  lty_name            AS
 ↪  институт,
ROUND(AVG(l_faculty_name__multiplied_tasks_and_semesters.m
 ↪  uptiply), 1) AS
 ↪  сложность
FROM (SELECT fac_name_sem_id_progr_cnt.faculty_name,
SUM(l_tasks_amount__semester_id.tasks_amount) * /*
 ↪  Количество дисциплин в семестре */
COUNT(fac_name_sem_id_progr_cnt.progress_amount_per_semest
 ↪  er) AS
 ↪  muptiply
FROM (SELECT l_tasks_amount__progress_id.tasks_amount,
 ↪  progress.semester_id
FROM (SELECT COUNT(*) AS tasks_amount, progress_id
FROM task
GROUP BY progress_id) AS l_tasks_amount__progress_id
INNER JOIN progress ON
 ↪  l_tasks_amount__progress_id.progress_id = progress.id)
 ↪  AS l_tasks_amount__semester_id
INNER JOIN
(SELECT l_faculty_name_semester_id.faculty_name,
progress.semester_id,
COUNT(*) AS progress_amount_per_semester /* Количество
 ↪  дисциплин */
FROM (SELECT DISTINCT ON (semester.student_id)
 ↪  l_faculty_name_student_id.faculty_name, semester.id
FROM (SELECT DISTINCT ON (l_faculty_name_group_id.id)
 ↪  l_faculty_name_group_id.faculty_name, student.id
FROM (SELECT l_faculty_name__dep_spec_id.faculty_name,
 ↪  "GROUP".id
FROM (SELECT l_faculty_name__department_id.faculty_name,
dep_spec.id
FROM (SELECT faculty.name AS faculty_name, department.id
FROM faculty
INNER JOIN department ON faculty.id = department.faculty_id
) AS l_faculty_name__department_id
INNER JOIN department_specialization AS dep_spec
ON l_faculty_name__department_id.id =
dep_spec.department_id
```

```
) AS l_faculty_name__dep_spec_id
INNER JOIN "GROUP"
ON l_faculty_name__dep_spec_id.id =
"GROUP".department_specialization_id
) AS l_faculty_name_group_id
INNER JOIN student ON student.group_id =
 ↪  l_faculty_name_group_id.id
) AS l_faculty_name_student_id
INNER JOIN semester ON semester.student_id =
 ↪  l_faculty_name_student_id.id
) AS l_faculty_name_semester_id
INNER JOIN progress ON progress.semester_id =
 ↪  l_faculty_name_semester_id.id
GROUP BY progress.semester_id,
 ↪  l_faculty_name_semester_id.faculty_name
) AS fac_name_sem_id_progr_cnt
ON fac_name_sem_id_progr_cnt.semester_id =
 ↪  l_tasks_amount__semester_id.semester_id
GROUP BY l_tasks_amount__semester_id.semester_id,
 ↪  fac_name_sem_id_progr_cnt.faculty_name
) AS l_faculty_name__multiplied_tasks_and_semesters
GROUP BY l_faculty_name__multiplied_tasks_and_semesters.fa
 ↪  culty_name
ORDER BY сложность DESC
LIMIT 5;
```

	институт	сложность
1	Институт компьютерных наук и технологий	173.8
2	Институт передовых производственных технологий	152.2
3	Институт промышленного менеджмента, экономики и торговли	150.4
4	Гуманитарный институт	141.9
5	Институт энергетики	141.1

Рис. 3.1. Результат выполнения индивидуального задания 1

2. Вывести преподавателей, которые каждый семестр с момента начала работы увеличивают количество дисциплин, которые преподают в рамках одного семестра.

```
CREATE OR REPLACE FUNCTION COUNT_SUBJECTS_OF_TEACHER_PER_S
 ↪  EMESTER(semester_month_start DATE, semester_month_end
 ↪  DATE)
```

```
RETURNS TABLE
(
teacher_id    INTEGER,
teacher_name  TEXT,
subject_count INTEGER
)
AS
'SELECT l_teacher_id_name__subject_id.teacher_id,
l_teacher_id_name__subject_id.name,
COUNT(*)
FROM (SELECT l_teacher_id_name__lesson_id.teacher_id,
l_teacher_id_name__lesson_id.name,
lesson.subject_id
FROM lesson
INNER JOIN
(SELECT teacher.id            AS teacher_id,
teacher.name,
lesson_teacher.lesson AS lesson_id
FROM teacher
INNER JOIN lesson_teacher ON teacher.id =
 ↪  lesson_teacher.teacher
) AS l_teacher_id_name__lesson_id
ON l_teacher_id_name__lesson_id.lesson_id = lesson.id
WHERE (lesson.lesson_date BETWEEN semester_month_start AND
 ↪  semester_month_end)
GROUP BY l_teacher_id_name__lesson_id.teacher_id,
 ↪  l_teacher_id_name__lesson_id.name, lesson.subject_id
) AS l_teacher_id_name__subject_id
GROUP BY l_teacher_id_name__subject_id.teacher_id,
 ↪  l_teacher_id_name__subject_id.name'
LANGUAGE sql
IMMUTABLE
RETURNS NULL ON NULL INPUT;

SELECT l_semester_1.teacher_name  AS преподаватель,
l_semester_1.subject_count AS количество_в_первом_семестре,
l_semester_2.subject_count AS количество_во_втором_семестре
FROM (SELECT *
FROM COUNT_SUBJECTS_OF_TEACHER_PER_SEMESTER(CURRENT_SETTIN
 ↪  G('schedule_vars.first_semester_start_date')::DATE,
CURRENT_SETTING('schedule_vars.first_semester_end_date')::
 ↪  DATE)) AS
 ↪  l_semester_1
```

```
INNER JOIN
(SELECT *
FROM COUNT_SUBJECTS_OF_TEACHER_PER_SEMESTER(CURRENT_SETTIN
 ↪  G('schedule_vars.second_semester_start_date')::DATE,
CURRENT_SETTING('schedule_vars.second_semester_end_date'):
 ↪  :DATE)) AS
 ↪  l_semester_2
ON l_semester_1.teacher_id = l_semester_2.teacher_id AND
 ↪  l_semester_1.subject_count <
 ↪  l_semester_2.subject_count;
```

490 rows

	преподаватель	количество_в_первом_семестре	количество_во_втором_семестре
1	Авдуевская Е.А.	1	2
2	Балашов Евгений Владимирович	1	3
3	Рожанский Владимир Александрович	3	4
4	Божокин Сергей Валентинович	1	2
5	Бурова Екатерина Валерьевна	2	4
6	Куликов Николай Викторович	2	3
7	Ливинцова Мария Геннадьевна	2	3
8	Силиненко Александр Витальевич	1	3
9	Тихонов Дмитрий Владимирович	1	3
10	Долотова Наталья Леонидовна	1	4
11	Воронова Лариса Сергеевна	1	2
12	Попов Аркадий Николаевич	2	3
13	Дуболазов Анатолий Александрович	1	2
14	Наумов Антон Алексеевич	1	3
15	Бурцева Александра Вячеславовна	5	7
16	Землинская Татьяна Евгеньевна	3	6
17	Бурдаков Сергей Федорович	1	2
18	Дмитриева Татьяна Александровна	2	5
19	Никитина Татьяна Александровна	2	3
20	Андрианова Екатерина Евгеньевна	1	2
21	Мокеева Татьяна Васильевна	2	3
22	Шевченко Сергей Михайлович	1	3
23	Сафонова Анна Сергеевна	2	8
24	Шарова Наталья Юрьевна	1	2
25	Шелейко Виктория Анатольевна	3	9
26	Ванина Полина Юрьевна	2	3
27	Хлопков Елисей Алексеевич	1	3
28	Иванников Никита Сергеевич	3	6
29	Измайлов Максим Кириллович	1	4
30	Шульгин Анатолий Анатольевич	1	2
31	Колесниченко-Янушев Сергей Леонидович	3	4
32	Колесник Ирина Ивановна	3	6
33	Тихомиров Виктор Васильевич	1	2
34	Лукин Алексей Вячеславович	1	2
35	Сланов Валерий Павлович	3	5

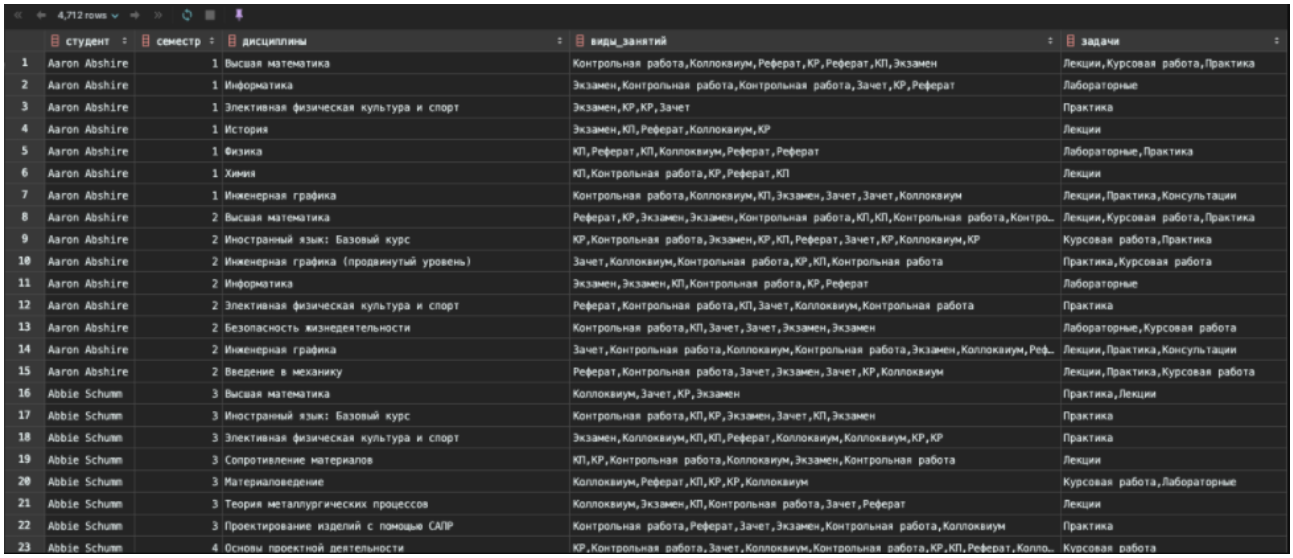
Рис. 3.2. Результат выполнения индивидуального задания 2

3. Вывести агрегированный учебный план заданного студента: по каждому семестру обучения вывести дисциплины, виды занятий по каждой, задачи по каждой.

```
SELECT l_student_name__level__subject_name_id__task_type.s
 ↪  tudent_name  AS
 ↪  студент,
l_student_name__level__subject_name_id__task_type.level
 ↪       AS семестр,
```

```
l_student_name__level__subject_name_id__task_type.subject_
 ↪  name  AS
 ↪  дисциплины,
l_student_name__level__subject_name_id__task_type.task_typ
 ↪  e_seq AS
 ↪  виды_занятий,
l_student_name__subject_type__lesson_type.lesson_type_seq
 ↪       AS задачи
FROM (SELECT l_student_name__lesson_type_id__subject_id.st
 ↪  udent_name,
l_student_name__lesson_type_id__subject_id.subject_id,
STRING_AGG(lesson_type.name, ',') AS lesson_type_seq
FROM (SELECT l_student_name__lesson_id.student_name,
 ↪  lesson.type, lesson.subject_id
FROM (SELECT l_student_name__group_id.student_name,
 ↪  lesson_group.lesson
FROM (SELECT student.name AS student_name, "GROUP".id
FROM student
INNER JOIN "GROUP" ON "GROUP".id = student.group_id
) AS l_student_name__group_id
INNER JOIN lesson_group ON lesson_group."GROUP" =
 ↪  l_student_name__group_id.id
) AS l_student_name__lesson_id
INNER JOIN lesson ON lesson.id =
 ↪  l_student_name__lesson_id.lesson
GROUP BY l_student_name__lesson_id.student_name,
 ↪  lesson.subject_id, lesson.type
) AS l_student_name__lesson_type_id__subject_id
INNER JOIN lesson_type
ON l_student_name__lesson_type_id__subject_id.type =
 ↪  lesson_type.id
GROUP BY l_student_name__lesson_type_id__subject_id.studen
 ↪  t_name,
l_student_name__lesson_type_id__subject_id.subject_id
) AS l_student_name__subject_type__lesson_type
INNER JOIN
(SELECT l_student_name__level__subject_name_id__task_type.
 ↪  student_name,
l_student_name__level__subject_name_id__task_type.level,
l_student_name__level__subject_name_id__task_type.subject_
 ↪  name,
l_student_name__level__subject_name_id__task_type.subject_
 ↪  id,
```

```
STRING_AGG(task_type.name, ',') AS task_type_seq
FROM (SELECT l_student_name__level__subject_id__progress_i
 ↪  d.student_name,
l_student_name__level__subject_id__progress_id.level,
subject.name AS subject_name,
subject.id   AS subject_id,
task.type_id
FROM (SELECT
 ↪  l_student_name__semester_id_level.student_name,
l_student_name__semester_id_level.level,
progress.subject_id,
progress.id AS progress_id
FROM (SELECT student.name AS student_name, semester.id,
 ↪  semester.level
FROM student
INNER JOIN semester ON student.id = semester.student_id
) AS l_student_name__semester_id_level
INNER JOIN progress ON progress.semester_id =
 ↪  l_student_name__semester_id_level.id
) AS l_student_name__level__subject_id__progress_id
INNER JOIN subject ON subject.id = l_student_name__level__
 ↪  subject_id__progress_id.subject_id
INNER JOIN task ON task.progress_id = l_student_name__leve
 ↪  l__subject_id__progress_id.progress_id
) AS l_student_name__level__subject_name_id__task_type
INNER JOIN task_type
ON task_type.id = l_student_name__level__subject_name_id__
 ↪  task_type.type_id
GROUP BY l_student_name__level__subject_name_id__task_type
 ↪  .student_name,
l_student_name__level__subject_name_id__task_type.level,
l_student_name__level__subject_name_id__task_type.subject_
 ↪  name,
l_student_name__level__subject_name_id__task_type.subject_
 ↪  id
) AS l_student_name__level__subject_name_id__task_type
ON l_student_name__level__subject_name_id__task_type.stude
 ↪  nt_name
 ↪  =
l_student_name__subject_type__lesson_type.student_name
AND l_student_name__subject_type__lesson_type.subject_id =
l_student_name__level__subject_name_id__task_type.subject_
 ↪  id;
```



	студент	семестр	дисциплины	виды_занятий	задачи
1	Aaron Abshire	1	Высшая математика	Контрольная работа,Коллоквиум,Реферат,КР,Реферат,КП,Экзамен	Лекции,Курсовая работа,Практика
2	Aaron Abshire	1	Информатика	Экзамен,Контрольная работа,Контрольная работа,Зачет,КР,Реферат	Лабораторные
3	Aaron Abshire	1	Элективная физическая культура и спорт	Экзамен,КР,КР,Зачет	Практика
4	Aaron Abshire	1	История	Экзамен,КП,Реферат,Коллоквиум,КР	Лекции
5	Aaron Abshire	1	Физика	КП,Реферат,КП,Коллоквиум,Реферат,Реферат	Лабораторные,Практика
6	Aaron Abshire	1	Химия	КП,Контрольная работа,КР,Реферат,КП	Лекции
7	Aaron Abshire	1	Инженерная графика	Контрольная работа,Коллоквиум,КП,Экзамен,Зачет,Зачет,Коллоквиум	Лекции,Практика,Консультации
8	Aaron Abshire	2	Высшая математика	Реферат,КР,Экзамен,Экзамен,Контрольная работа,КП,КП,Контрольная работа,Контро…	Лекции,Курсовая работа,Практика
9	Aaron Abshire	2	Иностранный язык: Базовый курс	КР,Контрольная работа,Экзамен,КР,КП,Реферат,Зачет,КР,Коллоквиум,КР	Курсовая работа,Практика
10	Aaron Abshire	2	Инженерная графика (продвинутый уровень)	Зачет,Коллоквиум,Контрольная работа,КР,КП,Контрольная работа	Практика,Курсовая работа
11	Aaron Abshire	2	Информатика	Экзамен,Экзамен,КП,Контрольная работа,КР,Реферат	Лабораторные
12	Aaron Abshire	2	Элективная физическая культура и спорт	Реферат,Контрольная работа,КП,Зачет,Коллоквиум,Контрольная работа	Практика
13	Aaron Abshire	2	Безопасность жизнедеятельности	Контрольная работа,КП,Зачет,Зачет,Экзамен,Экзамен	Лабораторные,Курсовая работа
14	Aaron Abshire	2	Инженерная графика	Зачет,Контрольная работа,Коллоквиум,Контрольная работа,Экзамен,Коллоквиум,Реф…	Лекции,Практика,Консультации
15	Aaron Abshire	2	Введение в механику	Реферат,Контрольная работа,Зачет,Экзамен,Зачет,КР,Коллоквиум	Лекции,Практика,Курсовая работа
16	Abbie Schumm	3	Высшая математика	Коллоквиум,Зачет,КР,Экзамен	Практика,Лекции
17	Abbie Schumm	3	Иностранный язык: Базовый курс	Контрольная работа,КП,КР,Экзамен,Зачет,КП,Экзамен	Практика
18	Abbie Schumm	3	Элективная физическая культура и спорт	Экзамен,Коллоквиум,КП,КП,Реферат,Коллоквиум,Коллоквиум,КР,КР	Практика
19	Abbie Schumm	3	Сопротивление материалов	КП,КР,Контрольная работа,Коллоквиум,Экзамен,Контрольная работа	Лекции
20	Abbie Schumm	3	Материаловедение	Коллоквиум,Реферат,КП,КР,КР,Коллоквиум	Курсовая работа,Лабораторные
21	Abbie Schumm	3	Теория металлургических процессов	Коллоквиум,Экзамен,КП,Контрольная работа,Зачет,Реферат	Лекции
22	Abbie Schumm	3	Проектирование изделий с помощью САПР	Контрольная работа,Реферат,Зачет,Экзамен,Контрольная работа,Коллоквиум	Практика
23	Abbie Schumm	4	Основы проектной деятельности	КР,Контрольная работа,Зачет,Коллоквиум,Контрольная работа,КР,КП,Реферат,Колло…	Курсовая работа

Рис. 3.3. Результат выполнения индивидуального задания 3

4. Вывести расписание дедлайнов заданного студента по неделям заданного семестра.

```
CREATE OR REPLACE FUNCTION
 ↪  WEEK_SCHEDULE_OF_DEADLINES(semester_start DATE,
 ↪  semester_end DATE, studentID INTEGER)
RETURNS TABLE
(
student_id        INTEGER,
student_name      TEXT,
week_start        DATE,
deadline_date     DATE,
notification_time TIME,
title             TEXT,
description       TEXT,
type              TEXT,
subject_id        INTEGER
)
AS
'SELECT l_existing_deadlines.studentID,
l_existing_deadlines.name,
l_all_semester_weeks.all_weeks,
l_existing_deadlines.deadline_date,
l_existing_deadlines.notification_time,
l_existing_deadlines.title,
l_existing_deadlines.description,
```

```
l_existing_deadlines.type,
l_existing_deadlines.subject_id
FROM (SELECT *
FROM GENERATE_SERIES(semester_start, semester_end, ''1
 ↪  week'') AS all_weeks) AS l_all_semester_weeks
LEFT JOIN
(SELECT studentID,
student.name,
week_start,
week_order_deadlines.deadline_date,
week_order_deadlines.notification_time,
week_order_deadlines.title,
week_order_deadlines.description,
week_order_deadlines.type,
week_order_deadlines.subject_id
FROM (SELECT DATE_TRUNC(''week'',
 ↪  deadline.deadline_date::DATE)::DATE AS week_start,
deadline.deadline_date,
deadline.notification_time,
deadline.title,
deadline.description,
deadline.type,
deadline.subject_id,
deadline.lesson_id,
deadline.student_id
FROM deadline
WHERE deadline.student_id = studentID
AND deadline_date BETWEEN semester_start AND semester_end
ORDER BY week_start, deadline_date, notification_time) AS
 ↪  week_order_deadlines
INNER JOIN student ON student.id =
 ↪  week_order_deadlines.student_id) AS
 ↪  l_existing_deadlines
ON l_existing_deadlines.week_start =
 ↪  l_all_semester_weeks.all_weeks'
LANGUAGE sql
IMMUTABLE
RETURNS NULL ON NULL INPUT;

SELECT *
FROM WEEK_SCHEDULE_OF_DEADLINES(CURRENT_SETTING('schedule_
 ↪  vars.second_semester_start_date')::DATE,
```

```
CURRENT_SETTING('schedule_vars.second_semester_end_date'):
 ↪  :DATE,
 ↪  1500);
```

71 rows

	student_id	student_name	week_start	deadline_date	notification_time	title	description	...	subject_id
1	<null>	<null>	2021-02-01	<null>	<null>	<null>	<null>	<null>	<null>
2	<null>	<null>	2021-02-08	<null>	<null>	<null>	<null>	<null>	<null>
3	<null>	<null>	2021-02-15	<null>	<null>	<null>	<null>	<null>	<null>
4	<null>	<null>	2021-02-22	<null>	<null>	<null>	<null>	<null>	<null>
5	<null>	<null>	2021-03-01	<null>	<null>	<null>	<null>	<null>	<null>
6	<null>	<null>	2021-03-08	<null>	<null>	<null>	<null>	<null>	<null>
7	<null>	<null>	2021-03-15	<null>	<null>	<null>	<null>	<null>	<null>
8	<null>	<null>	2021-03-22	<null>	<null>	<null>	<null>	<null>	<null>
9	<null>	<null>	2021-03-29	<null>	<null>	<null>	<null>	<null>	<null>
10	<null>	<null>	2021-04-05	<null>	<null>	<null>	<null>	<null>	<null>
11	<null>	<null>	2021-04-12	<null>	<null>	<null>	<null>	<null>	<null>
12	<null>	<null>	2021-04-19	<null>	<null>	<null>	<null>	<null>	<null>
13	100	Jesusa Brown	2021-04-26	2021-05-01	10:23:39	English test deadline	Should do the english c…	Deadline	105
14	100	Jesusa Brown	2021-04-26	2021-05-01	17:55:39	Databases lab deadline	Need to send report for…	Deadline	283
15	100	Jesusa Brown	2021-04-26	2021-05-02	03:43:09	Math class postponed	Today math class is pos…	Notifica…	1061
16	100	Jesusa Brown	2021-04-26	2021-05-02	16:07:33	ModelSim lab deadline	Send a report and files…	Deadline	2207
17	100	Jesusa Brown	2021-05-03	2021-05-03	04:23:36	English test deadline	Should do the english c…	Deadline	363
18	100	Jesusa Brown	2021-05-03	2021-05-03	06:54:03	Do the test before sun…	Need to do the test bef…	Task	118
19	100	Jesusa Brown	2021-05-03	2021-05-03	17:41:20	Jog in the morning	Will jog every morning …	Goal	1024
20	100	Jesusa Brown	2021-05-03	2021-05-03	17:49:42	English test deadline	Should do the english c…	Deadline	2424
21	100	Jesusa Brown	2021-05-03	2021-05-04	09:51:21	OpenEdu test deadline	Should do the OpenEdu t…	Deadline	129
22	100	Jesusa Brown	2021-05-03	2021-05-05	09:03:12	Teacher birthday today	Today is our teacher bi…	Reminder	568

Рис. 3.4. Результат выполнения индивидуального задания 4

5. Вывести наиболее простой и наиболее трудный (по количеству зачетов) семестр для заданной группы.

```
SELECT l_group_id_name__rank_max_min__level__sum_of_tasks.
 ↪  name AS
 ↪  группа,
(CASE
WHEN rank_max = 1 AND rank_min =/= 1 THEN 'наиболее трудный'
WHEN rank_max =/= 1 AND rank_min = 1 THEN 'наиболее простой'
WHEN rank_max = 1 AND rank_min = 1 THEN 'один семестр'
END)                                                    AS
 ↪  сложность_семестра,
level                                                   AS
 ↪  семестр,
sum_of_tasks_per_semester                               AS
 ↪  количество_зачётов
FROM (SELECT l_group_id_name__level__sum_of_tasks.group_id,
l_group_id_name__level__sum_of_tasks.group_name,
DENSE_RANK() OVER (PARTITION BY
 ↪  l_group_id_name__level__sum_of_tasks.group_name
ORDER BY l_group_id_name__level__sum_of_tasks.sum_of_tasks
 ↪  _per_semester DESC) AS
 ↪  rank_max,
```

```
DENSE_RANK() OVER (PARTITION BY
 ↪  l_group_id_name__level__sum_of_tasks.group_name
ORDER BY l_group_id_name__level__sum_of_tasks.sum_of_tasks
 ↪  _per_semester)      AS
 ↪  rank_min,
l_group_id_name__level__sum_of_tasks.level,
l_group_id_name__level__sum_of_tasks.sum_of_tasks_per_seme
 ↪  ster
FROM (SELECT l_group_id_name__level__progress_id.group_id,
l_group_id_name__level__progress_id.group_name,
l_group_id_name__level__progress_id.level,
COUNT(*) AS sum_of_tasks_per_semester
FROM (SELECT l_group_id_name__semester_id_level.group_id,
l_group_id_name__semester_id_level.group_name,
l_group_id_name__semester_id_level.level,
progress.id AS progress_id
FROM (SELECT l_group_id_name__student_id.group_id,
l_group_id_name__student_id.group_name,
semester.id AS semester_id,
semester.level
FROM (SELECT DISTINCT ON ("GROUP".id) "GROUP".id   AS
 ↪  group_id,
"GROUP".name AS group_name,
student.id   AS student_id
FROM "GROUP"
INNER JOIN student ON "GROUP".id = student.group_id
) AS l_group_id_name__student_id
INNER JOIN semester ON semester.student_id =
 ↪  l_group_id_name__student_id.student_id
) AS l_group_id_name__semester_id_level
INNER JOIN progress ON progress.semester_id =
 ↪  l_group_id_name__semester_id_level.semester_id
) AS l_group_id_name__level__progress_id
INNER JOIN task
ON task.progress_id =
 ↪  l_group_id_name__level__progress_id.progress_id AND
task.type_id = 1 /* ID ЗАЧЁТА  */
GROUP BY l_group_id_name__level__progress_id.group_id,
 ↪  l_group_id_name__level__progress_id.group_name,
l_group_id_name__level__progress_id.level
) AS l_group_id_name__level__sum_of_tasks
) AS l_group_id_name__rank_max_min__level__sum_of_tasks
```

```
WHERE /*l_group_id_name__rank_max_min__level__sum_of_tasks
 ↪  .name
in ('3430302/90001','3530904/70103','3531201/80201','36301
 ↪  02/80401','3631503/00001','3834101/00001','в3135401/80
 ↪  301','в3743805/90501')
/* Группы, для  которых созданы два семестра*/
AND*/ (l_group_id_name__rank_max_min__level__sum_of_tasks.
 ↪  rank_max = 1
 ↪  OR
l_group_id_name__rank_max_min__level__sum_of_tasks.rank_mi
 ↪  n =
 ↪  1)
ORDER BY group_id, сложность_семестра;
```

	группа	сложность_семестра	семестр	количество_зачётов
1	4341503/90701	один семестр	3	1
2	3733802/80301	один семестр	6	3
3	4851003/00002	наиболее простой	1	2
4	4851003/00002	наиболее трудный	2	3
5	3140801/01201	один семестр	2	1
6	в3743801/01401	один семестр	2	3
7	3743803/00301	один семестр	2	4
8	3530904/00006	наиболее простой	1	6
9	3530904/00006	наиболее трудный	2	9
10	3431104/90001	наиболее простой	4	4
11	3431104/90001	наиболее трудный	3	6
12	3331506/70101	один семестр	7	4
13	3140801/02301	один семестр	1	1
14	3140801/02301	один семестр	2	1
15	3530903/90001	наиболее простой	4	3
16	3530903/90001	наиболее трудный	3	10
17	3241302/90101	один семестр	3	1
18	3844502/90101	один семестр	3	3
19	3733802/80501	наиболее простой	5	1
20	3733802/80501	наиболее трудный	6	2
21	3844101/90101	один семестр	4	1
22	3241302/00101	один семестр	2	1
23	3743804/90501	наиболее простой	4	1
24	3743804/90501	наиболее трудный	3	4
25	3540901/91501	один семестр	3	2
26	3431101/80102	наиболее простой	6	5
27	3431101/80102	наиболее трудный	5	7
28	3640102/90101	один семестр	3	2

Рис. 3.5. Результат выполнения индивидуального задания 5

6. Вывести преподавателей, у которых занятия есть только в осеннем или весеннем семестре, но при этом суммарная нагрузка у них выше, чем средняя нагрузка преподавателей, которые реализуют те же профили.

```
CREATE OR REPLACE FUNCTION GET_TEACHING_DATES(study_year
 ↪  INTEGER)
RETURNS TABLE
(
```

```
teacher_id        INTEGER,
teacher_name      TEXT,
lesson_date       DATE,
lesson_subject_id INTEGER
)
AS
'SELECT l_teacher_id_name__lesson_id.teacher_id,
l_teacher_id_name__lesson_id.name,
lesson_date,
subject_id
FROM lesson
INNER JOIN
(SELECT teacher.id            AS teacher_id,
teacher.name,
lesson_teacher.lesson AS lesson_id
FROM teacher
INNER JOIN lesson_teacher ON teacher.id =
 ↪  lesson_teacher.teacher) AS l_teacher_id_name__lesson_id
ON l_teacher_id_name__lesson_id.lesson_id = lesson.id
WHERE lesson_date BETWEEN MAKE_DATE(study_year, 09, 01)
 ↪  AND MAKE_DATE(study_year + 1, 08, 31)'
LANGUAGE sql
IMMUTABLE
RETURNS NULL ON NULL INPUT;

CREATE OR REPLACE FUNCTION
 ↪  GET_BUSIER_TEACHER_THAN_AVERAGE(study_year INTEGER)
RETURNS TABLE
(
преподаватель        TEXT,
нагрузка             FLOAT,
средняя_нагрузка     FLOAT,
профиль_преподавания TEXT
)
AS
'SELECT teacher_name,
l_single_semester_teachers.lesson_amount,
ROUND(l_teaching_profile__profile_average_lesson_amount.le
 ↪  sson_amount,
 ↪  1),
ARRAY_TO_STRING(
ARRAY(SELECT name
```

```
FROM subject
WHERE id = ANY
 ↪  (l_single_semester_teachers.teaching_profile::int[])),
'', '')::TEXT AS teacher_profile
FROM (SELECT teacher_id,
teacher_name, /*first_lesson_date, last_lesson_date,*/
lesson_amount,
teaching_profile
FROM (SELECT teacher_id,
teacher_name,
MIN(lesson_date)                              AS
 ↪  first_lesson_date,
MAX(lesson_date)                              AS
 ↪  last_lesson_date,
COUNT(*)                                      AS
 ↪  lesson_amount,
ARRAY_AGG(DISTINCT lesson_subject_id::TEXT) AS
 ↪  teaching_profile
FROM GET_TEACHING_DATES(study_year)
GROUP BY teacher_id, teacher_name) AS
 ↪  l_teacher_id_name__first_last_lesson_date
WHERE EXTRACT(MONTH FROM first_lesson_date) NOT IN (9, 10,
 ↪  11, 12, 1)
OR EXTRACT(MONTH FROM last_lesson_date) NOT IN (2, 3, 4,
 ↪  5, 6, 7, 8)) AS l_single_semester_teachers
INNER JOIN
(SELECT teaching_profile,
AVG(lesson_amount) AS lesson_amount
FROM (SELECT COUNT(*)
 ↪  AS lesson_amount,
ARRAY_AGG(DISTINCT lesson_subject_id::TEXT) AS
 ↪  teaching_profile
FROM GET_TEACHING_DATES(study_year)
GROUP BY teacher_id, teacher_name) AS
 ↪  l_teaching_profile__lesson_amount
GROUP BY teaching_profile) AS
 ↪  l_teaching_profile__profile_average_lesson_amount
ON l_single_semester_teachers.teaching_profile =
l_teaching_profile__profile_average_lesson_amount.teaching
 ↪  _profile
AND
l_single_semester_teachers.lesson_amount > l_teaching_prof
 ↪  ile__profile_average_lesson_amount.lesson_amount'
```

```
LANGUAGE sql
IMMUTABLE
RETURNS NULL ON NULL INPUT;

SELECT *
FROM GET_BUSIER_TEACHER_THAN_AVERAGE(
EXTRACT(YEAR FROM current_setting('schedule_vars.first_sem
↪  ester_start_date')::DATE)::INTEGER)
```

	преподаватель	нагрузка	средняя_нагрузка	профиль_преподавания
1	Каштанова Станислава Викторовна	16	15.6	Теоретическая механика
2	Михель Екатерина Алексеевна	50	27.5	Экономика
3	Королева Юлиана Викторовна	41	24	Иностранный язык: Базовый курс
4	Терешко Е.К.	23	12.5	Управление проектами
5	Бушма Татьяна Валерьевна	51	27.3	Элективная физическая культура и спорт
6	Рыбин Валерий Васильевич	11	8	Иностранный язык в профессиональной коммуникации, Английский
7	Кузькин Виталий Андреевич	22	13.5	Курсовое проектирование по теоретической механике
8	Сарамаха Софья Игоревна	19	8	Иностранный язык в профессиональной коммуникации, Английский
9	Мяснов Александр Владимирович	26	14	Базы данных
10	Склярова Анастасия Сергеевна	50	37	Физика
11	Пылькин Александр Александрович	24	19.3	Философия
12	Гусейнов Тимур Мухтарович	29	27.3	Элективная физическая культура и спорт
13	Ананьевский Михаил Сергеевич	11	6.5	Объектно-ориентированное программирование
14	Леньшин Алексей Иванович	18	12.5	Компьютерные, сетевые и информационные технологии в электроэнергетике и электротехнике
15	Хомицевич Нина Анатольевна	2	1.5	Электрические машины
16	Дьяченко Галина Брониславовна	43	27.3	Элективная физическая культура и спорт
17	Попова Ольга Васильевна	43	38	Статистика

Рис. 3.6. Результат выполнения индивидуального задания 6

3.4. Выводы

В данной лабораторной работе нами были изучены возможности создания запросов и получения различных данных из таблицы; был изучен язык SQL_DML, с помощью которого были написаны 12 базовых запросов и 6 сложных запросов согласно заданию преподавателя.

4. Лабораторная работа 4

4.1. Цели работы

Сформировать тестовое приложение для трех уровней изолированности транзакций и поставить эксперимент.

4.2. Программа работы

Для БД с предварительно сгенерированным набором тестовых данных (генерация должна быть осуществлена с помощью приложения, реализованного в рамках работы №2) запустить 3 потока для работы с БД:

- поток №1 выполняет выборку данных из одной из таблиц БД
- поток №2 добавляет по одно записи в ту же таблицу БД
- поток №3 изменяет записи в той же таблице БД, условие для изменения должно затрагивать и данные потока №1 и запись, добавляемую потоком №2

Каждый поток должен выполнить серию (нужно иметь возможность указать ее размер) запросов к БД, для каждого запроса должно быть измерено среднее время выполнения. Для каждого потока для каждого уровня изолированности необходимо вывести среднее время выполнения запроса. Проанализировать полученные результаты и продемонстрировать преподавателю.

4.3. Выполнение работы

В данной лабораторной работе используется библиотека `org.postgresql` для обращения к базе данных и библиотека `jetbrains.lets-plot` для отрисовки графиков, используя язык `Kotlin`.

В функции `main` происходит запуск метода `testAllTransactions` и отрисовка графиков для полученных данных - для 6 запусков по 200 запросов для `SELECT`, `INSERT` и `UPDATE` для разных уровней изоляции.

```
@ExperimentalTime
fun main() {

    if (isTestTransactionsEnabled) {
        val connection = createConnection() ?: return

        repeat(configProperties
        .getProperty("config.fullTestsAmount")
```

```
        .toInt()) {
            testAllTransactions(connection)
        }
    } else {
        createPlots()
    }
}
```

Метод `testAllTransactions` параллельно запускает запросы `SELECT`, `INSERT` и `UPDATE` сначала для уровня изоляции `READ COMMITTED`, затем для уровня `REPEATABLE READ` и наконец для уровня `SERIALIZABLE`.

```
@ExperimentalTime
fun testAllTransactions(connection: Connection) {
val threadsAmount = RequestType.values().size
var lastDeadlines = 0
if (configProperties
.getProperty("config.isReadCommittedTestEnabled").toBoolean()) {
val readCommittedExecutor =
 ↪  Executors.newFixedThreadPool(threadsAmount)
testTransactions(connection, readCommittedExecutor, `READ
 ↪  COMMITTED`, lastDeadlines)
while (!readCommittedExecutor.isTerminated) {
}
lastDeadlines += insertRequestsAmount
}
if (configProperties
.getProperty("config.isRepeatableReadTestEnabled").toBoolean())
 ↪  {
val repeatableReadExecutor =
 ↪  Executors.newFixedThreadPool(threadsAmount)
testTransactions(connection, repeatableReadExecutor, `REPEATABLE
 ↪  READ`, lastDeadlines)
while (!repeatableReadExecutor.isTerminated) {
}
lastDeadlines += insertRequestsAmount
}
if (configProperties
.getProperty("config.isSerializableTestEnabled").toBoolean()) {
val serializableExecutor =
 ↪  Executors.newFixedThreadPool(threadsAmount)
testTransactions(connection, serializableExecutor, SERIALIZABLE,
 ↪  lastDeadlines)
```

```
while (!serializableExecutor.isTerminated) {
}
}
}
```

Для каждого из процессов создается `threadPool` с 3 потоками, в каждом из них исполняется свой запрос и замеряется среднее время исполнения для построения графика.

```
private fun testTransactions(
connection: Connection,
executor: ExecutorService,
transactionType: TransactionType,
lastDeadlines: Int
) {
val selectTransactionWorker = Runnable {
startTransaction(connection, SELECT, transactionType,
 ↪  selectRequestsAmount, lastDeadlines)
}
val insertTransactionWorker = Runnable {
startTransaction(connection, INSERT, transactionType,
 ↪  insertRequestsAmount, lastDeadlines)
}
val updateTransactionWorker = Runnable {
startTransaction(connection, UPDATE, transactionType,
 ↪  updateRequestsAmount, lastDeadlines)
}
with(executor) {
execute(selectTransactionWorker)
execute(insertTransactionWorker)
execute(updateTransactionWorker)
}
executor.shutdown()
}
```

После выполнения программы получаются следующие графики: Для `SELECT`

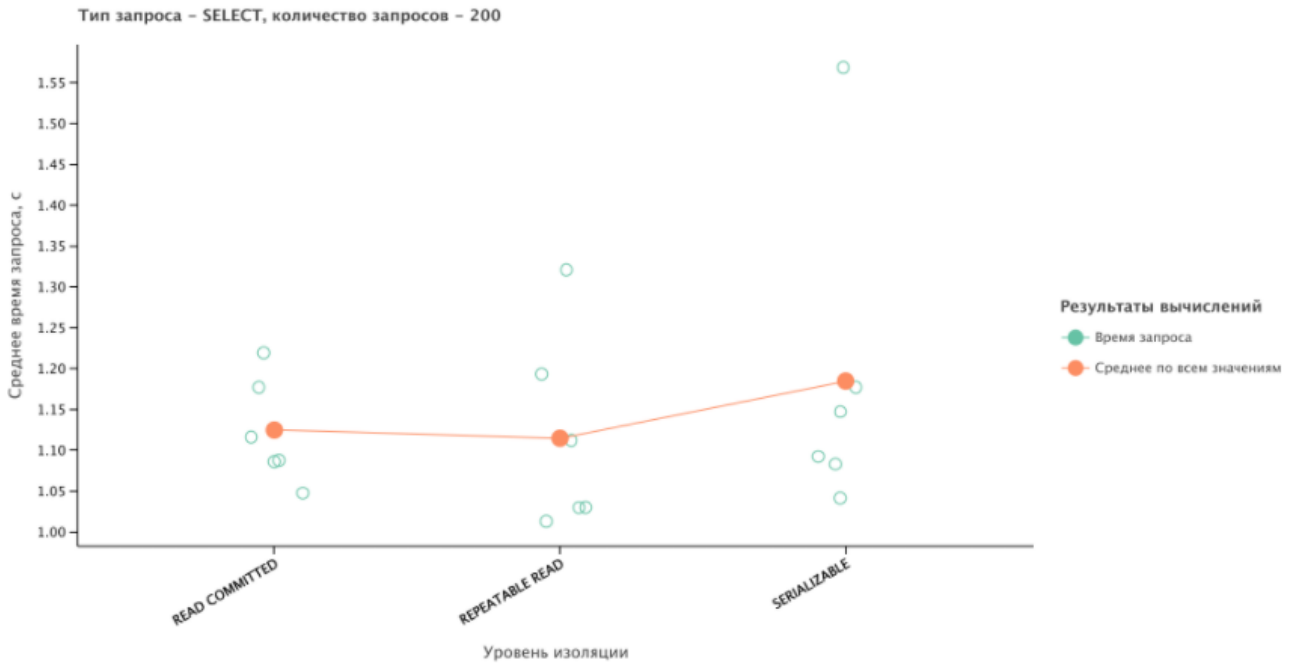


Рис. 4.1. График средней длительности SELECT для разных уровней изоляции

Для `INSERT`

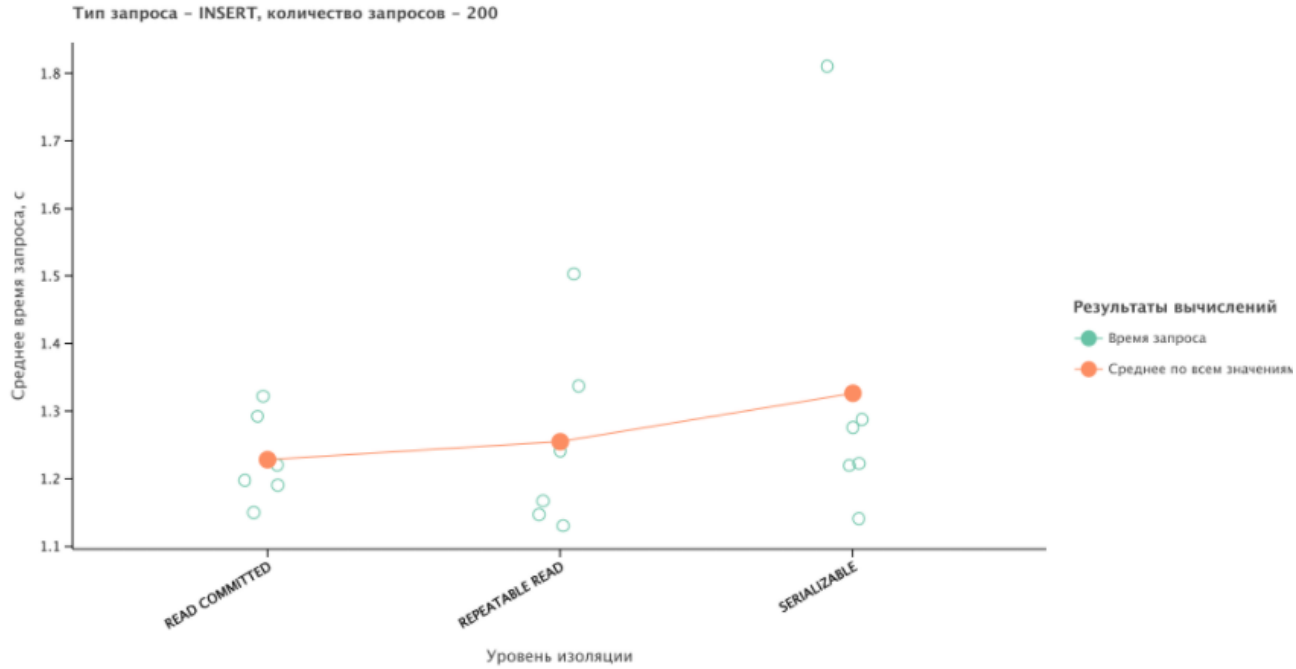


Рис. 4.2. График средней длительности INSERT для разных уровней изоляции

И для `UPDATE`

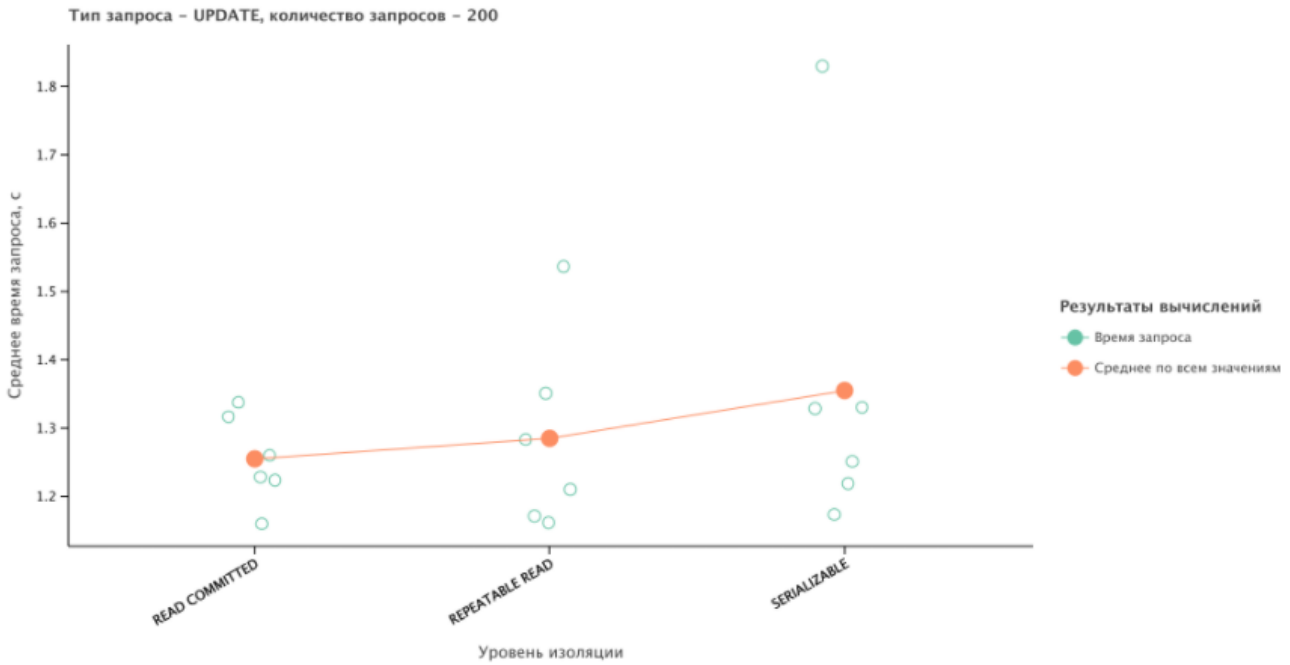


Рис. 4.3. График средней длительности UPDATE для разных уровней изоляции

Как можно видеть из графиков, с увеличением уровня изолированности транзакций увеличивается среднее время исполнения запроса, однако увеличение это не в несколько раз, а на несколько процентов.

Полный исходный код проекта доступен по ссылке.

4.4. Выводы

В данной лабораторной работе мы разработали приложение для изучения работы различных уровней изолированности транзакций, протестировали его работу и построили графики, на основе которых сделали вывод о том, что с увеличением уровня изолированности растет среднее время исполнения запроса.

5. Курсовая работа

5.1. Цели работы

Систематизация и углубление полученных знаний, самостоятельное изучение избранных вопросов и применение их на практике.

5.2. Программа работы

1. Выбор способа реализации курсовой работы
2. Написание и согласование технического задания по курсовой работе с подробным описанием реализуемой функциональности
3. Реализация всей требуемой функциональности
4. Тестирование корректности работы
5. Демонстрация результатов преподавателю
6. Оформление отчета по курсовой работе

5.3. Выполнение работы

5.3.1. Выбор способа реализации курсовой работы

В качестве реализации курсовой работы были выбраны веб-сервер, вебсайт и мобильное приложение, в соответствии с количеством участников команды. Веб-сервер взаимодействует с базой данных, написанной в лабораторных работах и предоставляет RESTful API для вебсайта и приложения. Вебсайт и приложение обрабатывают эти данные и предоставляет конечному пользователю заданный функционал.

5.3.2. Написание и согласование технического задания по курсовой работе с подробным описанием реализуемой функциональности

Курсовая работа имеет следующее техническое задание:

- Веб-сервер
    1. Разделение слоя работы с данными и слоя бизнес-логики
    2. Реализация кеширования результатов выборки из БД
    3. Предоставление эндпоинтов для аутентификации, дедлайнов, институтов, групп и занятий для вебсайта/приложения, в формате `JSON`
    4. Возможность добавлять/удалять дедлайны

5. Возможность только для пользователей с ролью `ADMIN` регистрировать новых пользователей
6. Хеширование паролей пользователей при добавлении нового пользователя
7. Генерация для аутентифицированных пользователей `JWT` токена, и его валидация

- Вебсайт/приложение

1. Страница аутентификации, на которой не аутентифицированный пользователь может ввести логин и пароль, и если в базе данных есть пользователь с такими данными, то успешно аутентифицироваться и получить `JWT` токен, чтобы при переходе между страницами сохранялась аутентификация
2. Проверка валидности `JWT` токена при каждом переходе в вебсайте/-приложении
3. Страница дедлайнов аутентифицированного пользователя
4. Страница институтов, на которой представлен список всех институтов и которую может посмотреть каждый пользователь
5. Страница групп, показывающая группы факультета, выбранного на странице факультетов и которую может посмотреть каждый пользователь
6. Страница занятий, показывающая за текущую неделю занятия группы, выбранной на странице групп и которую может посмотреть каждый пользователь. Также возможность переключиться на предыдущую/следующую неделю, чтобы посмотреть расписание группы за выбранную другую неделю.
7. Страница регистрации пользователей для пользователей с ролью `ADMIN`. Для регистрации студента требуется указать логин, пароль, группу и роль пользователя

5.3.3. Реализация всей требуемой функциональности

Веб-сервер был написан на фреймворке spring boot.

С помощью библиотеки `spring-boot-starter-data-jpa` вынесем работу с базой данных в репозитории (напр. `FacultyRepository`), а бизнес-логика будет написана в сервисах (`FacultyService`).

Рис. 5.1. Структура проекта

Кеширование было реализовано с помощью библиотеки `caffeine`, для её подключения к проекту необходимо создать конфигурационный класс, в котором как `CacheManager` будет установлет объект `Caffeine`. И после этого кешируемые данные следует отметить аннотацией `@Cacheable`.

```
@Configuration
public class CacheConfig {

    @Bean
    public Caffeine<Object, Object> caffeineConfig() {
        return Caffeine.newBuilder()
            .maximumSize(1000)
```

```
            .expireAfterAccess(30, TimeUnit.DAYS);
    }

    @Bean
    public CacheManager cacheManager() {
        CaffeineCacheManager caffeineCacheManager = new
        ↪  CaffeineCacheManager();
        caffeineCacheManager.setCaffeine(caffeineConfig());
        return caffeineCacheManager;
    }
}
```

Например, было сделано три запроса институтов, но в базу данных был отправлен только один запрос.

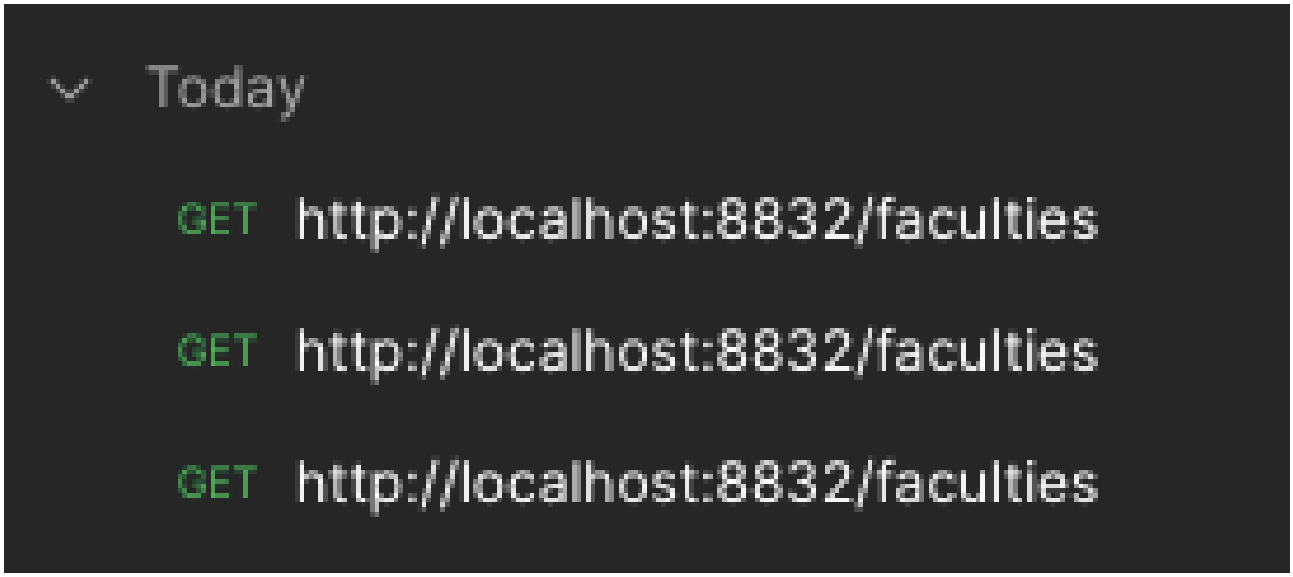


Рис. 5.2. Запросы институтов

```
2021-05-22 19:58:56.320  INFO 46667 --- [nio-8832-exec-1]
 ↪  o.s.web.servlet.DispatcherServlet        : Completed
 ↪  initialization in 1 ms
Hibernate: select faculty0_.id as id1_4_, faculty0_.name as
 ↪  name2_4_, faculty0_.short as short3_4_ from faculty
 ↪  faculty0_ order by faculty0_.id asc
```

В папке `controllers` проекта добавлены контроллеры, которые по заданным адресам предоставляют данные в формате `JSON`. Для того, чтобы получить данные используется метод `GET`, для добавления - `POST`, и для удаления - `DELETE`.

Для проверки ролей пользователя была использована библиотека `Spring Boot Security`, в ней мы можем сделать так, чтобы только пользователи с ролью `ADMIN` могли пользоваться методом добавления пользователей.

```
@Override
protected void configure(HttpSecurity http) throws Exception {
    http
        .csrf()
        .disable()
        .authorizeRequests()
        .antMatchers("/login").permitAll()
        .antMatchers("/registration")
        .hasAuthority("ADMIN")
        .anyRequest().permitAll();
}
```

Пароли хэшируются с помощью библиотеки `BCrypt`

```
@Bean
public PasswordEncoder passwordEncoder() {
    return new BCryptPasswordEncoder();
}
```

```
@Override
protected void configure(AuthenticationManagerBuilder auth)
 ↪  throws Exception {
    auth
        .userDetailsService(studentDetailsService)
        .passwordEncoder(passwordEncoder);
}
```

Для генерации и валидации `JWT` токена используем библиотеку `io.jsonwebtoken:jjwt`. В качестве параметров для токена передаём логин, пароль, группу, роль и идентификатор.

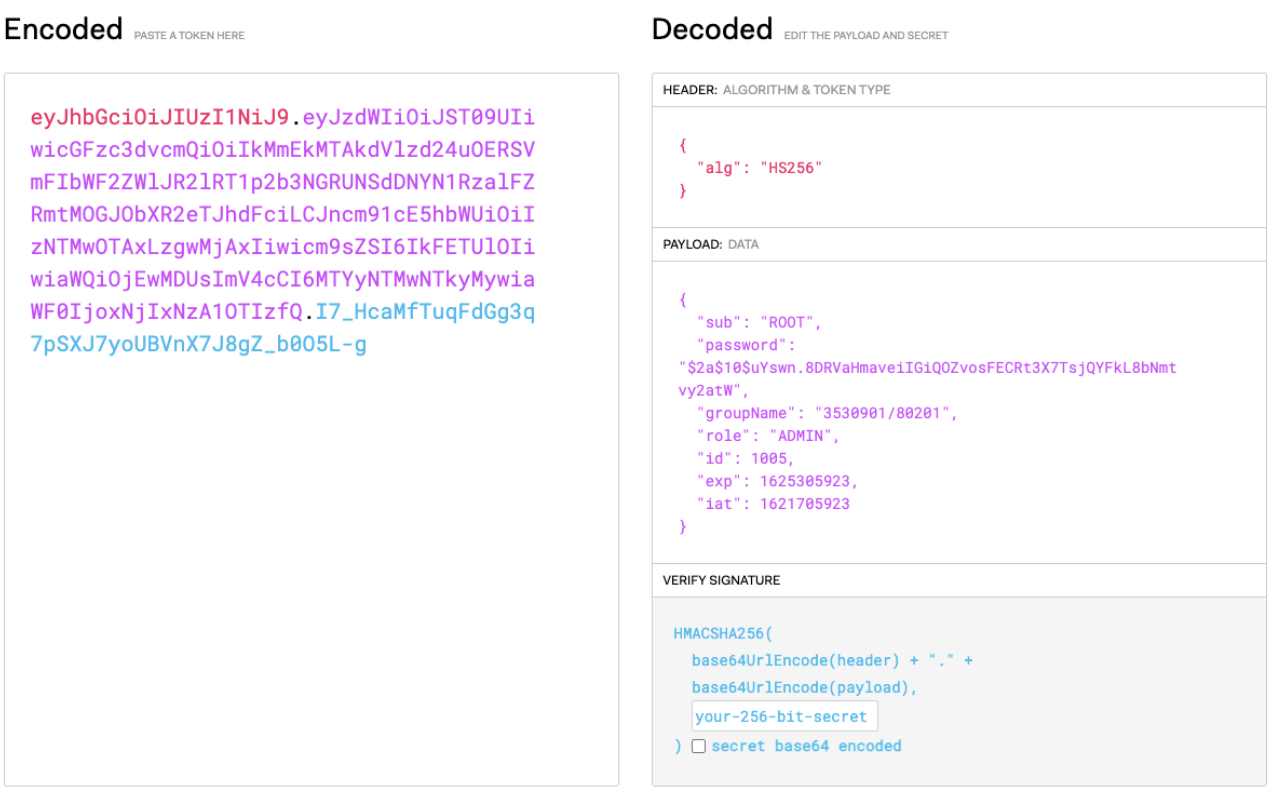


Рис. 5.3. Данные JWT токена

Генерируется токен следующим образом:

```
public String generateToken(StudentDetails userDetails) {
    Map<String, Object> claims = new HashMap<>();

    claims.put("id", userDetails.getStudent().getId());
    claims.put("password", userDetails.getPassword());
    claims.put("groupName",
     ↪  userDetails.getStudent().getGroup().getName());
    claims.put("role", userDetails.getStudent().getRole());

    return createToken(claims, userDetails.getUsername());
}

private String createToken(Map<String, Object> claims, String
 ↪  subject) {
    final long currentTimeMillis = System.currentTimeMillis();
    return Jwts
        .builder()
        .setClaims(claims)
        .setSubject(subject)
```

```
        .setIssuedAt(new Date(currentTimeMillis))
        .setExpiration(new Date(currentTimeMillis +
        ↪  3_600_000_000L))
        .signWith(SignatureAlgorithm.HS256, SECRET_KEY)
        .compact();
}
```

И валидируется следующим образом:

```
public void validateToken(String token, StudentDetails
 ↪  studentDetails) {
    try {
        final Claims claims = extractAllClaims(token);
        final String subject = claims.getSubject();
        final Long id = claims.get("id", Long.class);
        final String groupName = claims.get("groupName",
         ↪  String.class);
        final String password = claims.get("password",
         ↪  String.class);
        final String role = claims.get("role", String.class);

        if (!(studentDetails
            .getStudent()
            .getName()
            .equals(subject)
            && studentDetails
            .getStudent()
            .getId()
            .equals(id)
            && studentDetails
            .getStudent()
            .getGroup()
            .getName()
            .equals(groupName)
            && studentDetails
            .getPassword()
            .equals(password)
            && studentDetails
            .getStudent()
            .getRole()
            .name()
            .equals(role)
```

```
            && !isTokenExpired(token))) {
            throw new ForbiddenApiException("Некорректные поля
             ↪  токена");
        }
    } catch (Exception e) {
        e.printStackTrace();
        throw new ForbiddenApiException("Некорректный токен");
    }
}
```

Вебсайт был написан с помощью библиотеки `React`, приложение было написано на языке `Kotlin`. Для отправки запросов в вебсайте была использована библиотека `Axios`, с помощью неё отправляются `GET` запросы к веб-серверу и, если отправленные данные не соответствуют данным из базы данных, то показывается уведомление о не успешной аутентификации. Для отправки запросов в приложении была использована библиотека `okhttp3`, с помощью нее отправляются `GET` и `POST` запросы к веб-серверу, а также `jwt token` для обращения к данным, уникальным для каждого отдельного пользователя (как дедлайны), при успешной регистрации/логине выдается `jwt token` для обращения к уникальным данным, при обращении к запрашиваемым данным возвращается список из них, а при добавлении нового элемента возвращается ошибка при неудачной попытке добавления.

В вебсайте при каждом переходе вызывается метод `validate`, который проверяет `JWT` токен. Если он невалидный (напр. истёк срок действия), то аутентификация пользователя сбрасывается. Во всех случаях в приложении при возникновении каких-либо ошибок на экране показывается уведомление об ошибке и программа не производит действий, следующих за успешным выполнением запроса, а при невалидном `jwt` токене приложение переходит к экрану авторизации для ввода логина и пароля.

У аутентифицированного пользователя в верхнем левом углу появляется кнопка "Дедлайны", по которой можно перейти на страницу дедлайнов текущего пользователя. В вебсайте это проверяется флагом `isLoggedIn`, находящемся в глобальном состоянии `UserState`. Состояния были добавлены с помощью библиотеки `MobX`. Страница дедлайнов аутенфицированного пользователя в приложении отображает все дедлайны данного пользователя за все время, при нажатии на любой из элементов отображается его полное описание. Также присутствует кнопка добавления нового дедлайна, где заполняются все необходимые поля и после этого жмется кнопка "добавить" для отправки `POST` запроса на сервер.

В вебсайте на страницу институтов можно попасть, нажав на "Главная" в левой верхней части экрана. Страница институтов в приложении реализована с помощью списка произвольного размера, при нажатии на любой из элементов

происходит переход к списку групп, соответствующему данному институту.

В вебсайте при нажатии на какой-либо из институтов можно перейти на страницу групп этого факультета. Для получения групп конкретного факультета есть метод на веб-сервере, идентификатор факультета приходит на страницу групп через `PathVariable` в адресе страницы. Группы имеют разделение по курсу. На странице групп при нажатии на любой из элементов происходит переход к занятиям данной группы.

В вебсайте страница занятий по схожему со страницей групп получает занятия конкретной группы за текущую неделю. Также реализованы ссылки "Предыдущая неделя" и "Следующая неделя", позволяющие получить занятия за предыдущую и следующую неделю соответственно. Дата недели, за которую получается расписание, передаётся через адрес страницы. Страница занятий в приложении реализована также с помощью списка произвольного размера с виджетами произвольного размера, от пустого виджета (нет занятий в этот день) до виджета с 6 элементами, сверху этого списка присутствуют две кнопки, при нажатии на которые осуществляется переход на предыдущую/следующую неделю расписания.

В вебсайте для пользователей с ролью `ADMIN` появляется кнопка "Админка", ведущая на страницу регистрации пользователей. На этой странице есть форма с четырьмя полями: логином, паролем, группой и ролью пользователя. При нажатии кнопки "Регистрация" на веб-сервер приходит `POST` запрос, создающий нового пользователя. Страница регистрации новых пользователей появляется в меню настроек у пользователей с ролью `ADMIN`, в ней указываются все необходимые поля и получаемая роль, после чего отправляется `POST` запрос на добавление этого пользователя.

5.3.4. Тестирование корректности работы

Работа вебсайта была протестирована в браузере.

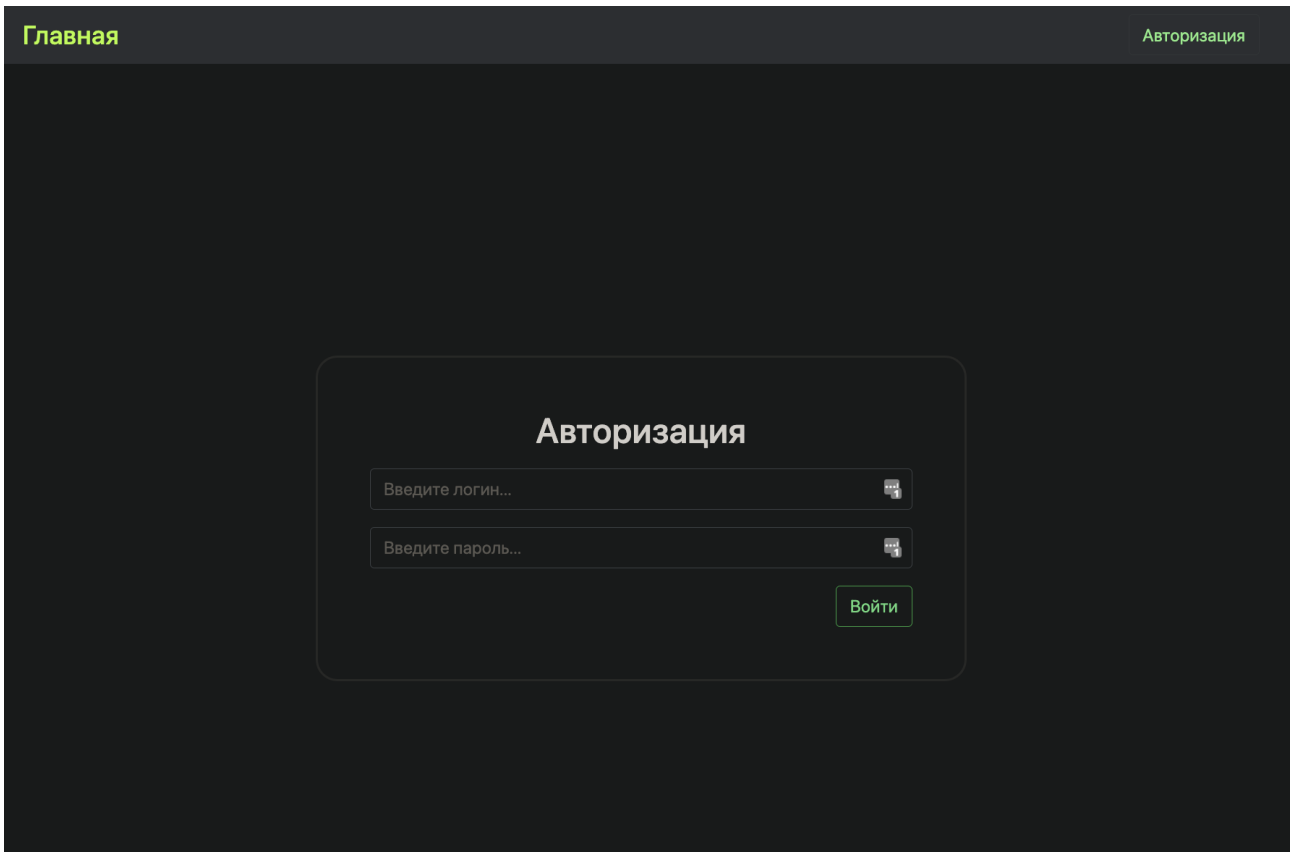


Рис. 5.4. Страница аутентификации пользователя в браузере

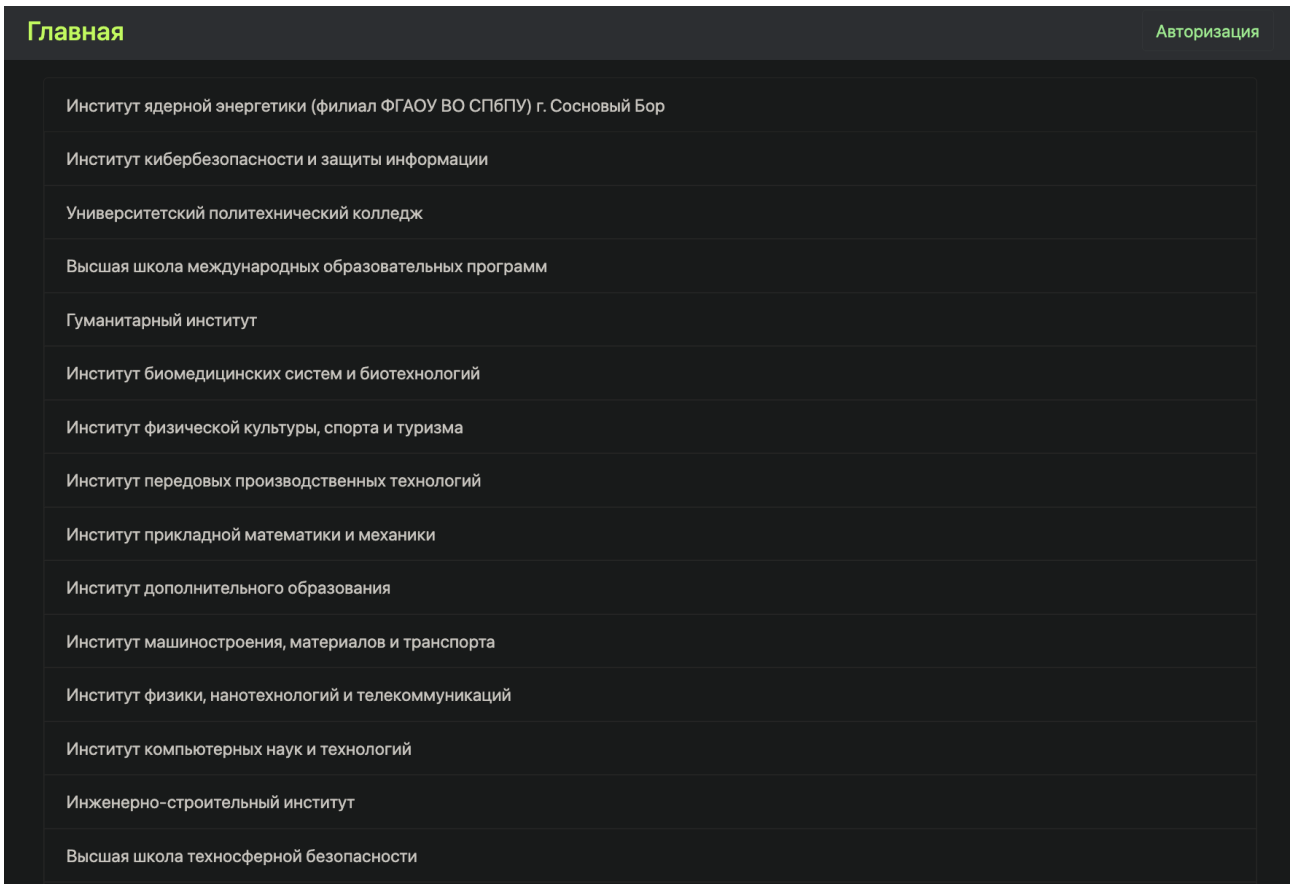


Рис. 5.5. Страница институтов в браузере

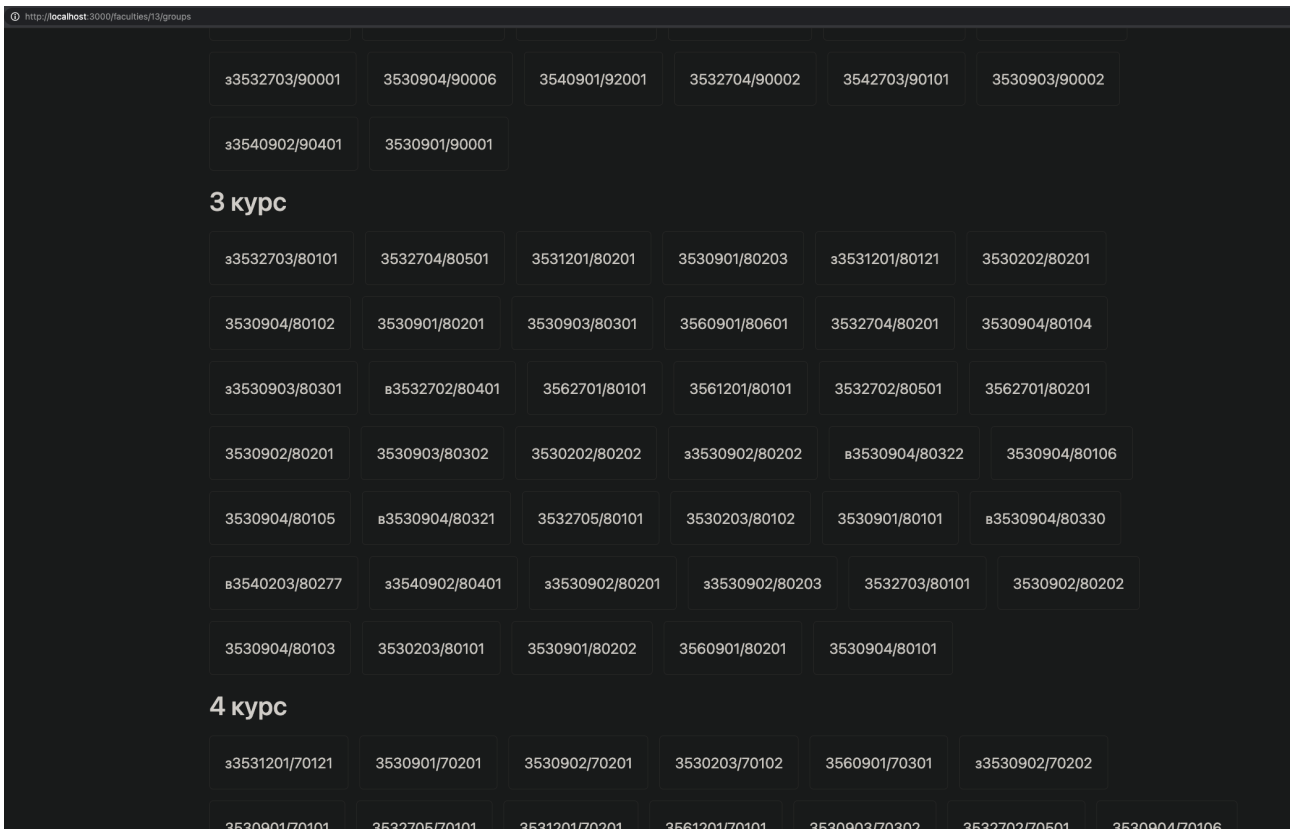


Рис. 5.6. Страница групп в браузере

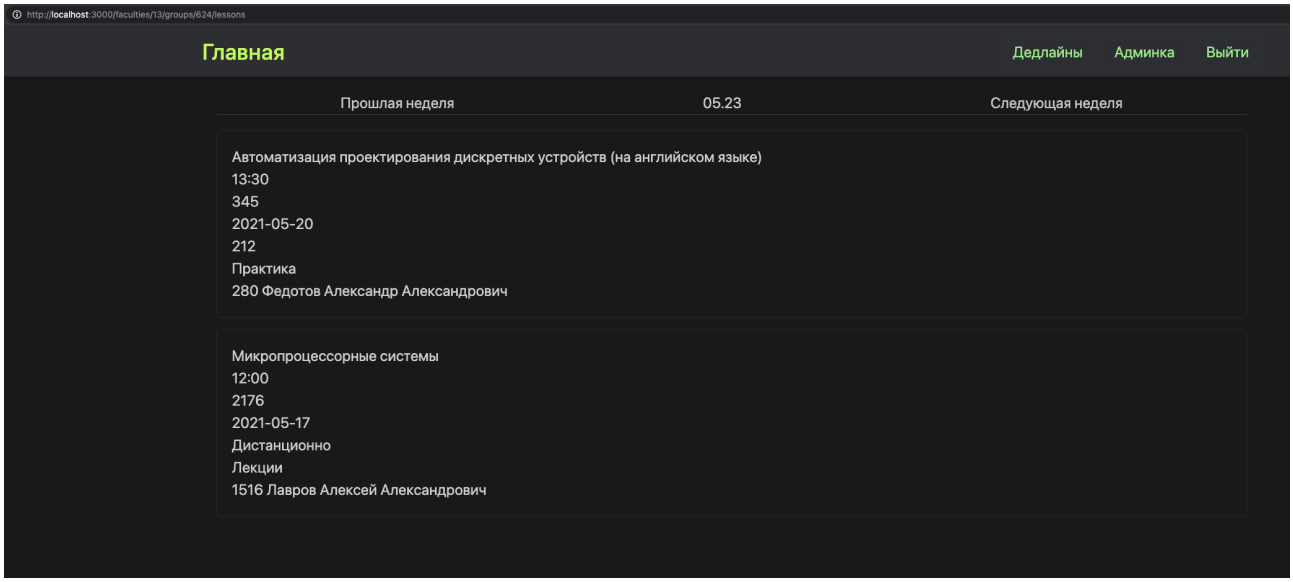


Рис. 5.7. Страница занятий за текущую неделю в браузере

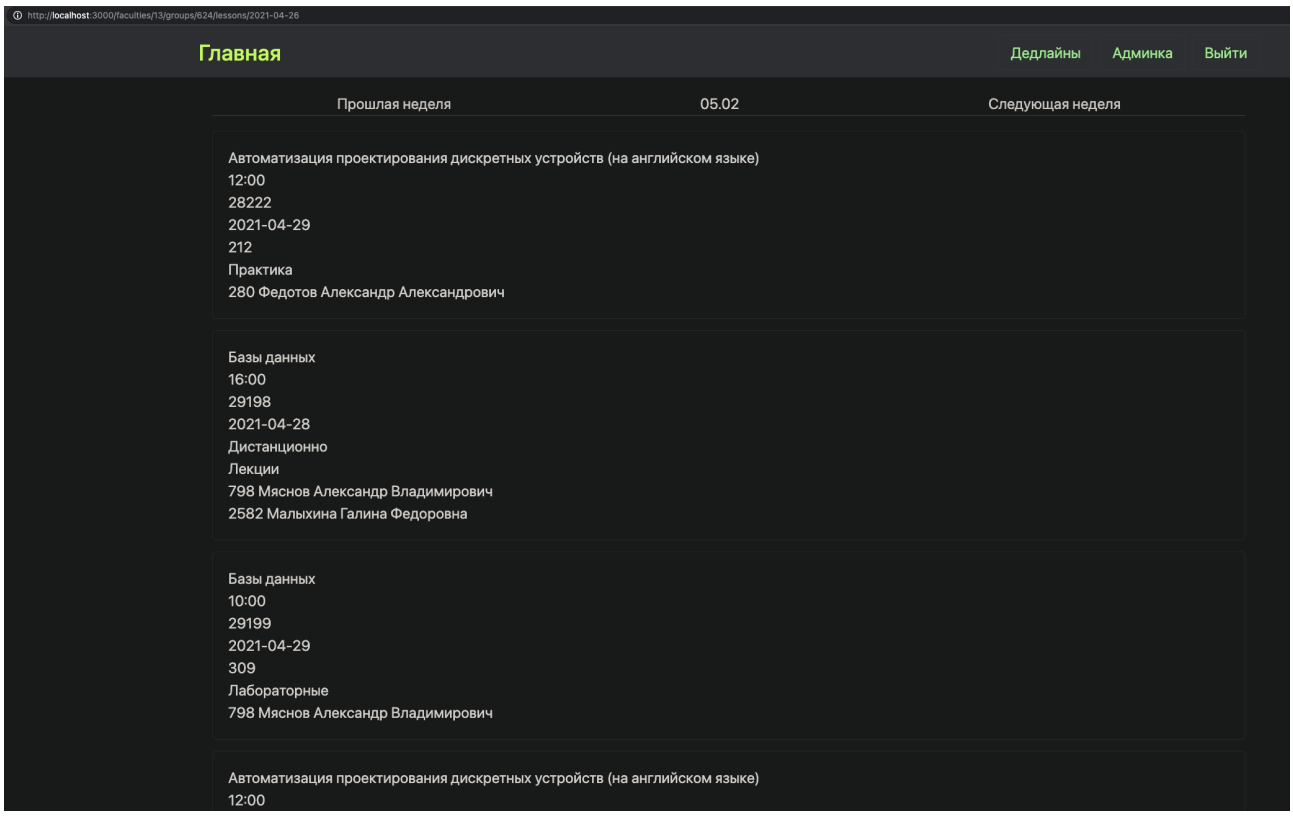


Рис. 5.8. Страница за другую неделю в браузере

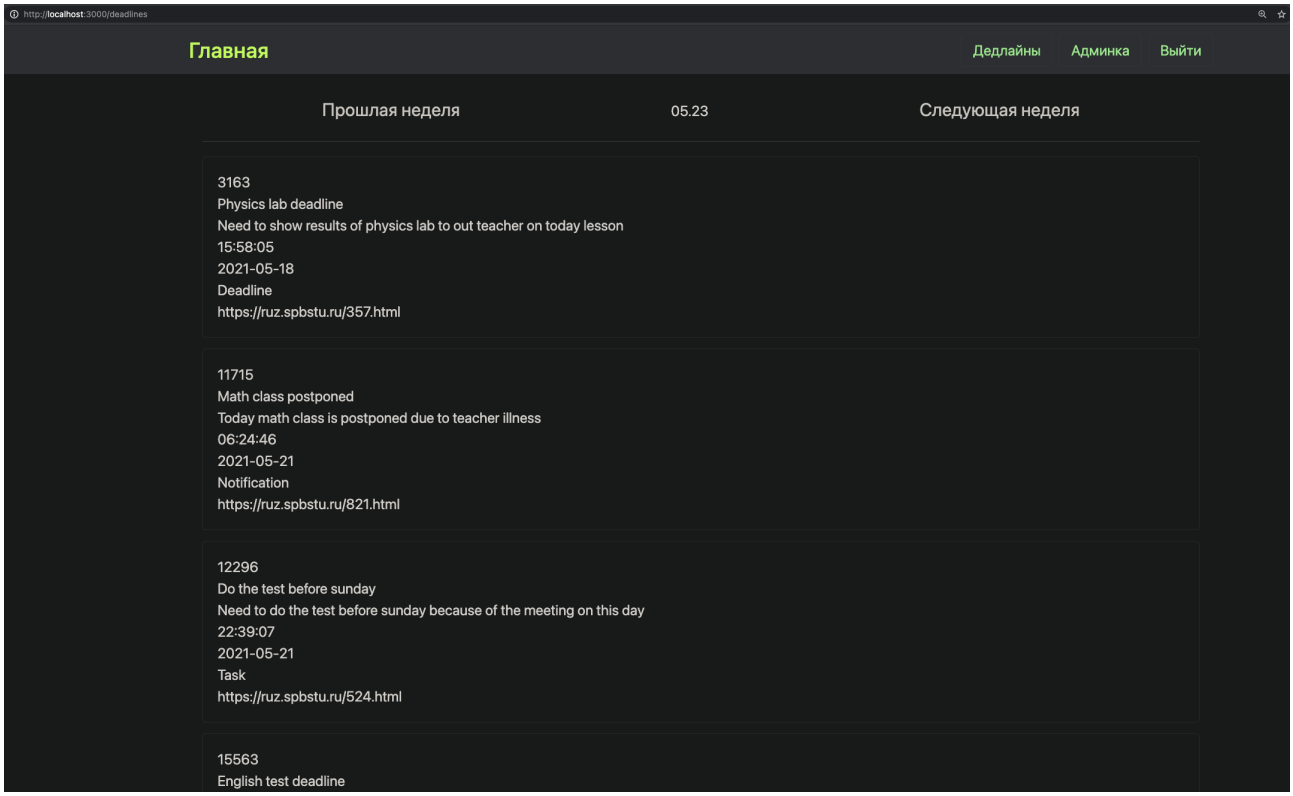


Рис. 5.9. Страница дедлайнов в браузере

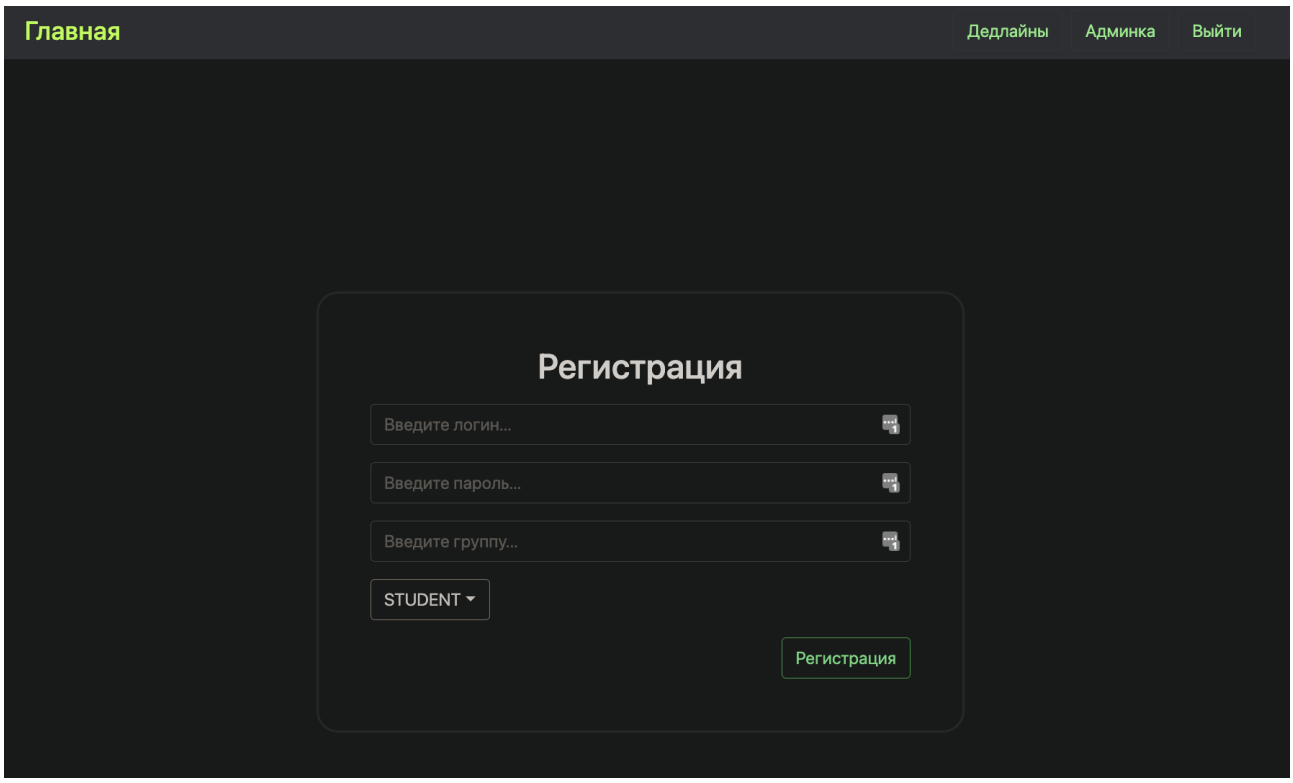
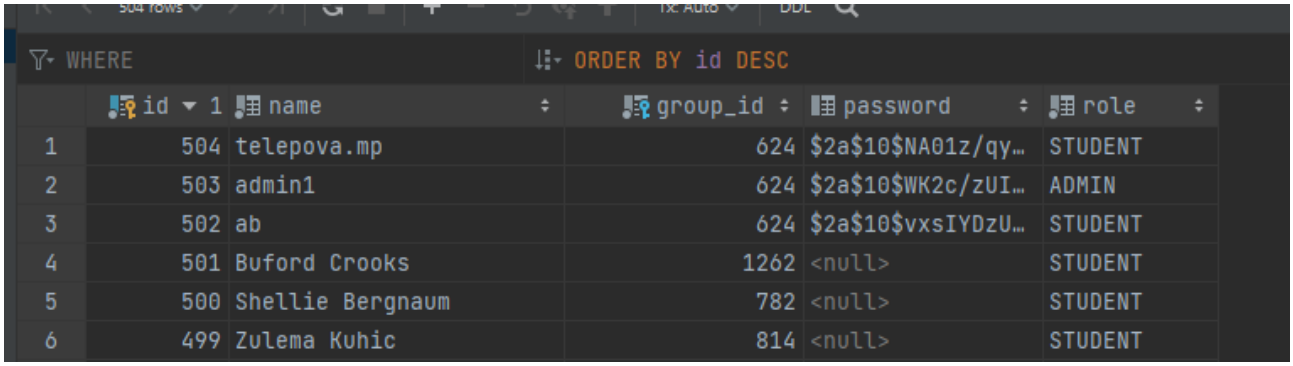


Рис. 5.10. Страница регистрации нового пользователя в браузере

Работа мобильного приложения была протестирована с помощью встроенного Android-эмулятора.

В таблице студентов базы данных находятся несколько уже зарегистрированных пользователей:



WHERE ORDER BY id DESC

	id 1	name	group_id	password	role
1	504	telepova.mp	624	$2a$10$NA01z/qy…	STUDENT
2	503	admin1	624	$2a$10$WK2c/zUI…	ADMIN
3	502	ab	624	$2a$10$vxsIYDzU…	STUDENT
4	501	Buford Crooks	1262	<null>	STUDENT
5	500	Shellie Bergnaum	782	<null>	STUDENT
6	499	Zulema Kuhic	814	<null>	STUDENT

Рис. 5.11. Таблица студентов

Запустим наше приложение. При первом запуске отображается экран авторизации. Воспользуемся учетной записью admin1 и авторизуемся за нее:

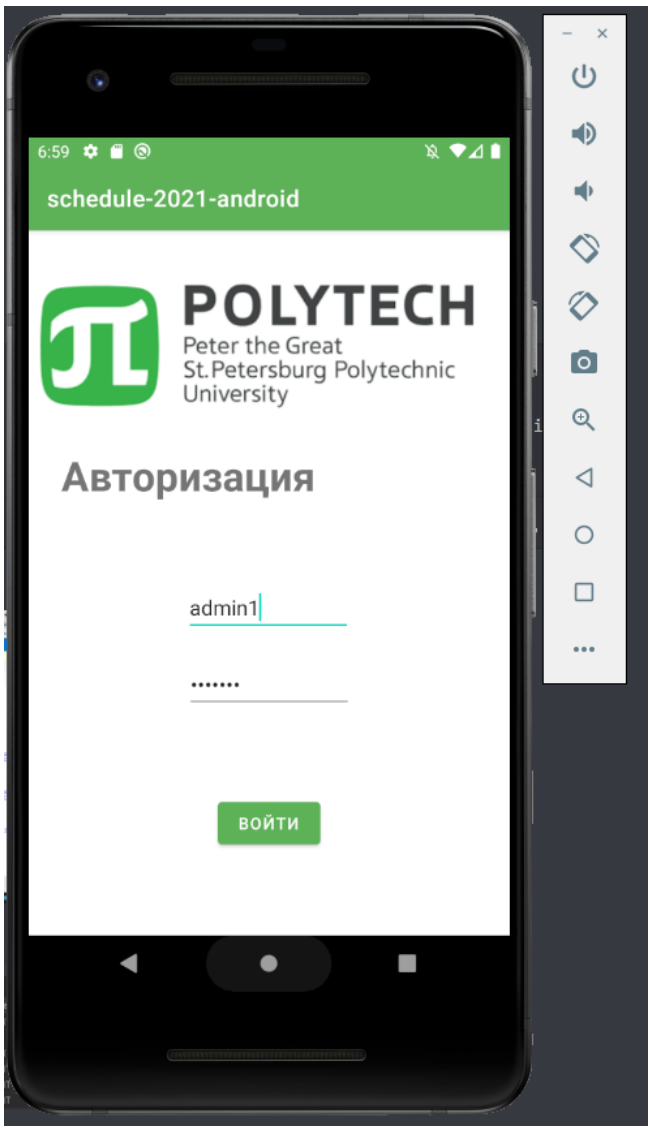


Рис. 5.12. Экран авторизации

После авторизации открывается экран с расписанием:

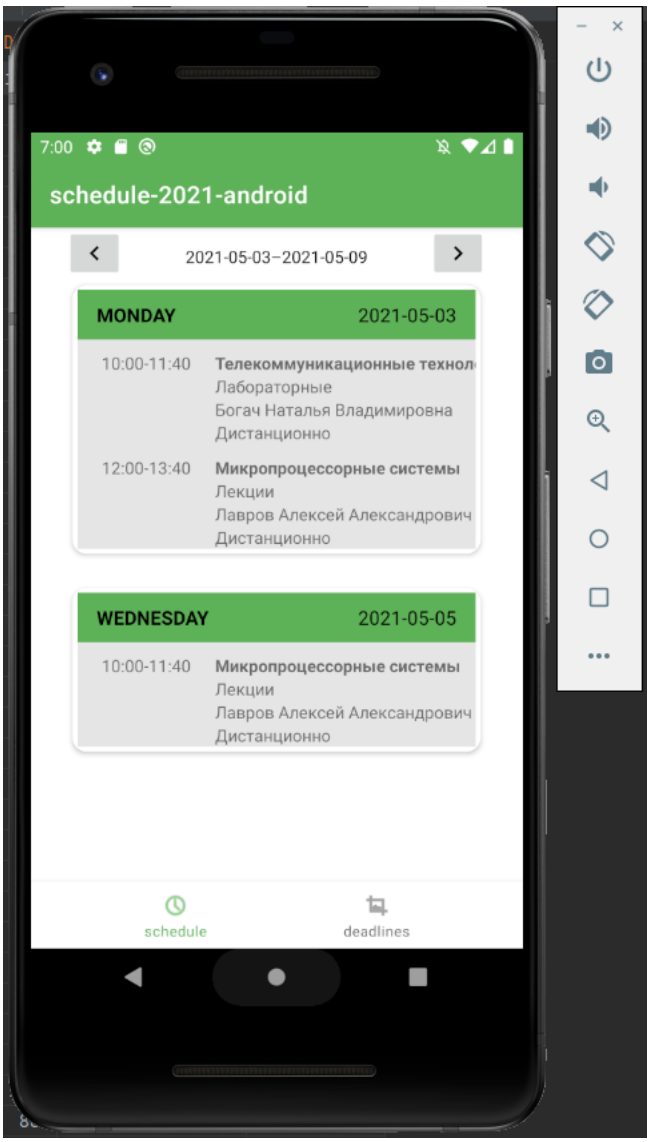


Рис. 5.13. Экран расписания

Нажмем кнопку "deadline" и посмотрим текущий список дедлайнов пользователя admin1:

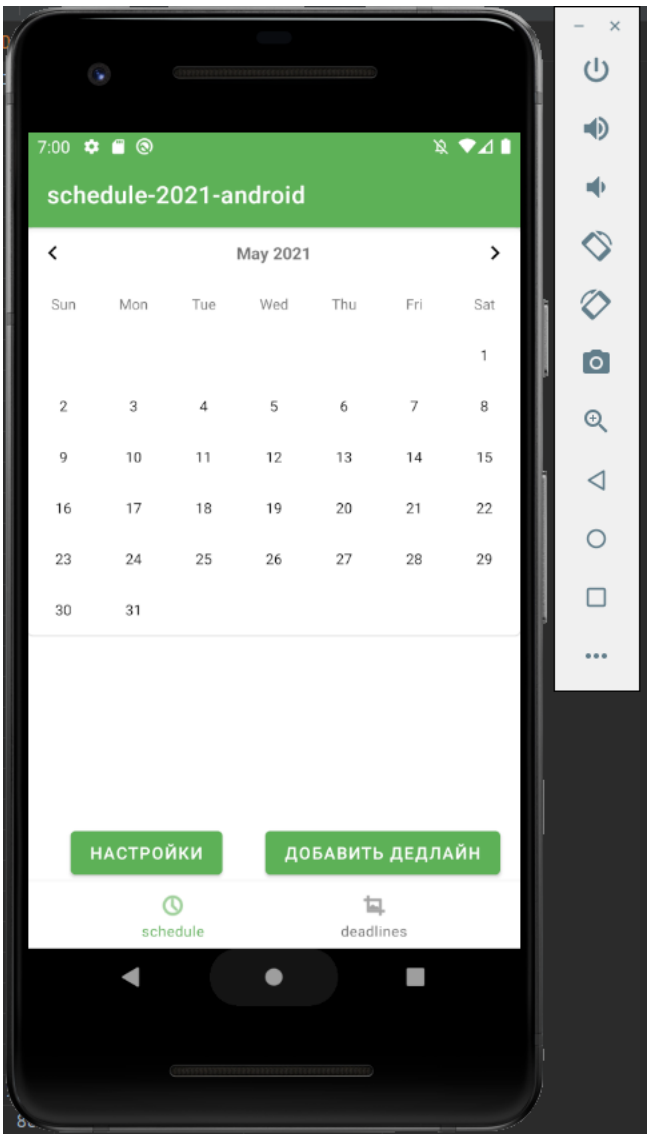


Рис. 5.14. Первоначальный экран дедлайнов

Можно заметить, что он пустой, поскольку дедлайнов у этого пользователя в базе данных еще нет:

Рис. 5.15. Первоначальное количество записей у пользователя admin1

Добавим новый дедлайн пользователю через мобильное приложение:

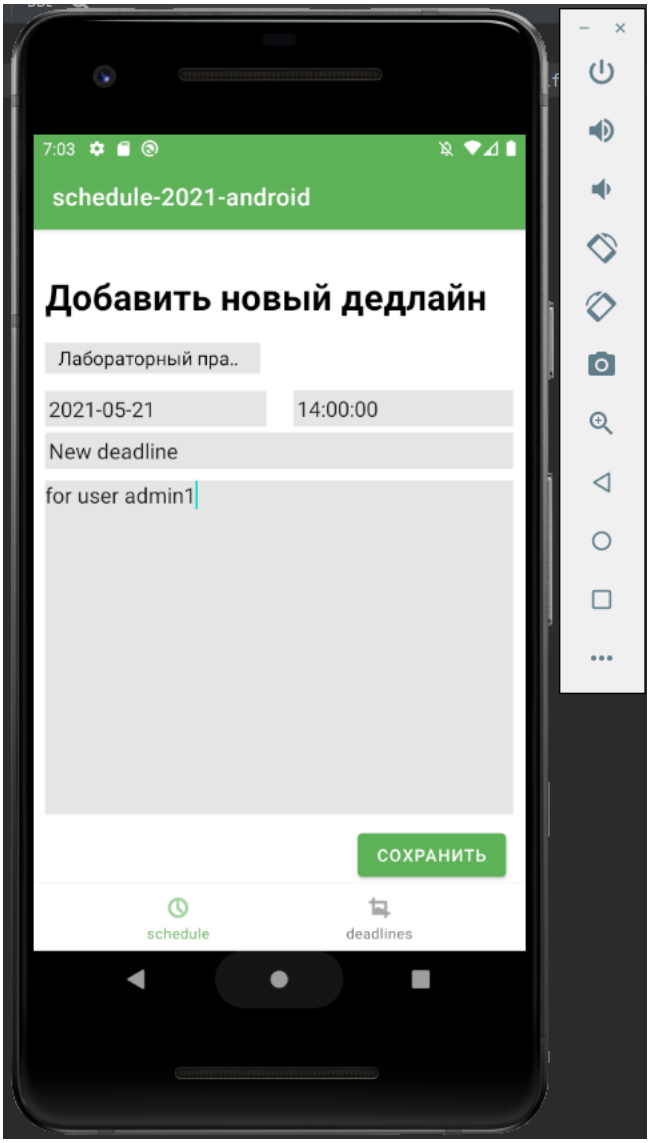


Рис. 5.16. Добавление нового дедлайна

Теперь посмотрим состояние базы данных и экрана дедлайнов:

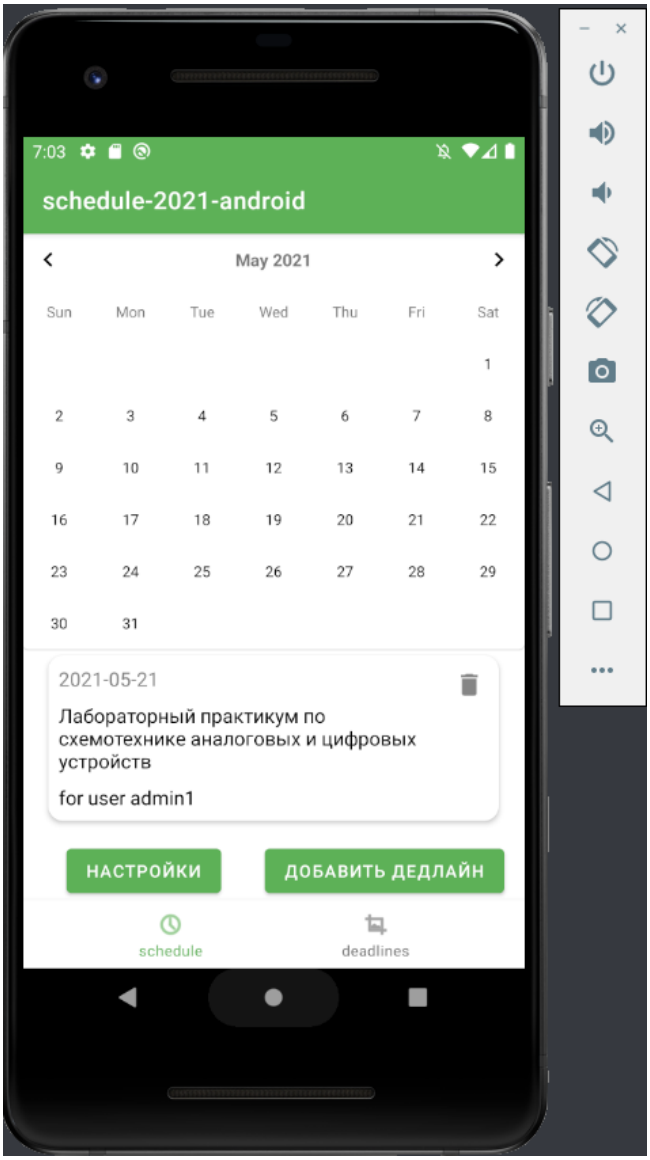
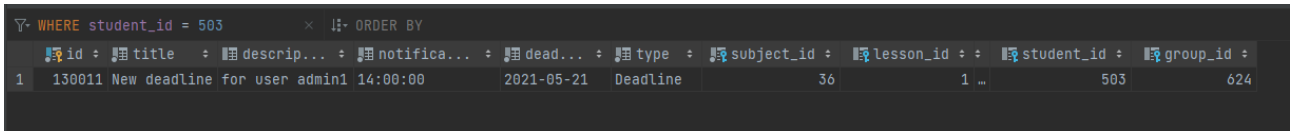


Рис. 5.17. Экран дедлайнов после добавления нового дедлайна



WHERE student_id = 503 ORDER BY

	id	title	descrip...	notifica...	dead...	type	subject_id	lesson_id	…	student_id	group_id
1	130011	New deadline	for user admin1	14:00:00	2021-05-21	Deadline	36	1	…	503	624

Рис. 5.18. Таблица дедлайнов после добавления дедлайна

Таким образом, дедлайн был успешно добавлен. Теперь попробуем его удалить:

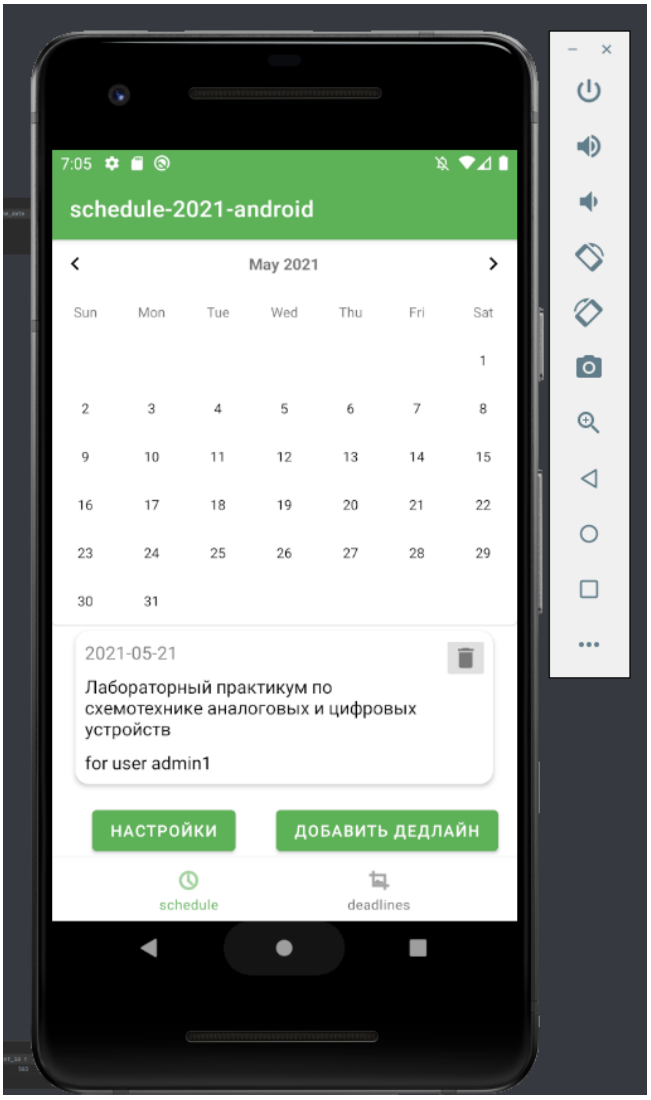


Рис. 5.19. Экран дедлайнов во время нажатия кнопки удаления

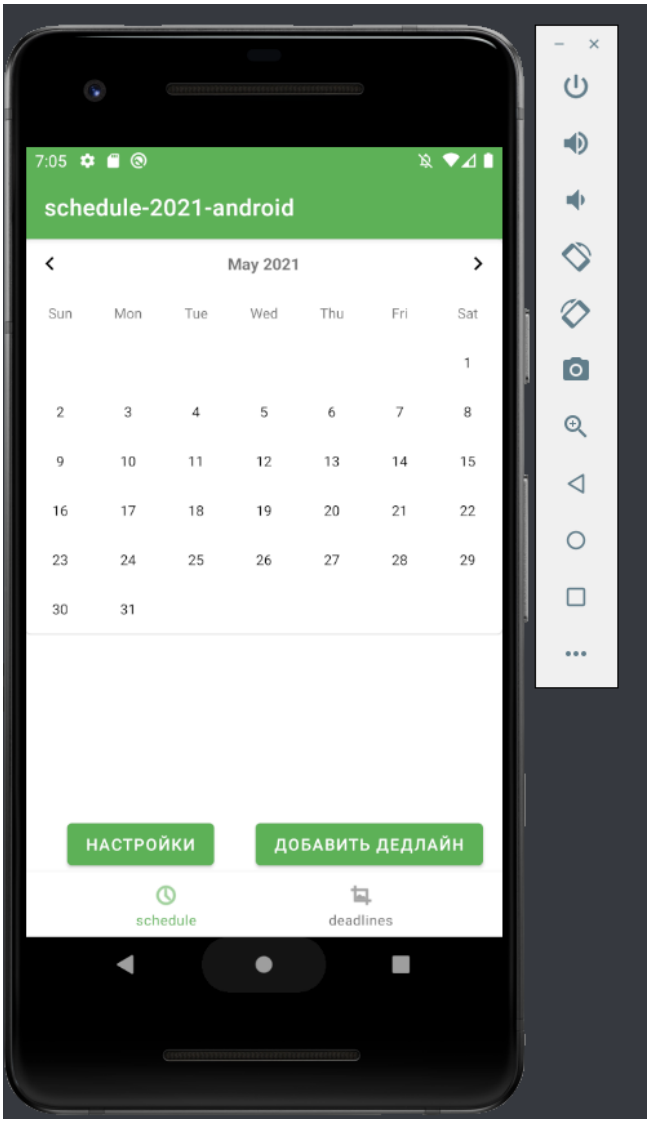


Рис. 5.20. Экран дедлайнов после нажатия кнопки удаления

В приложении дедлайн был успешно удален. Теперь посмотрим на состояние базы данных:

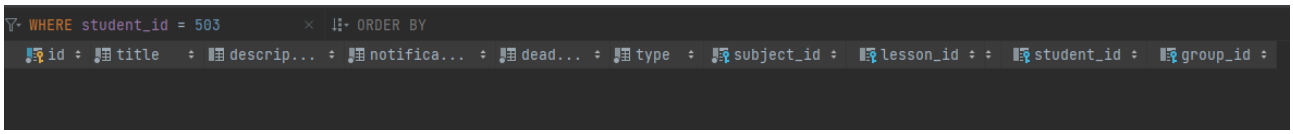


Рис. 5.21. Таблица дедлайнов после удаления дедлайна

Теперь у пользователя admin1 нет дедлайнов и в базе данных, и в приложении.

Теперь посмотрим экран настроек:

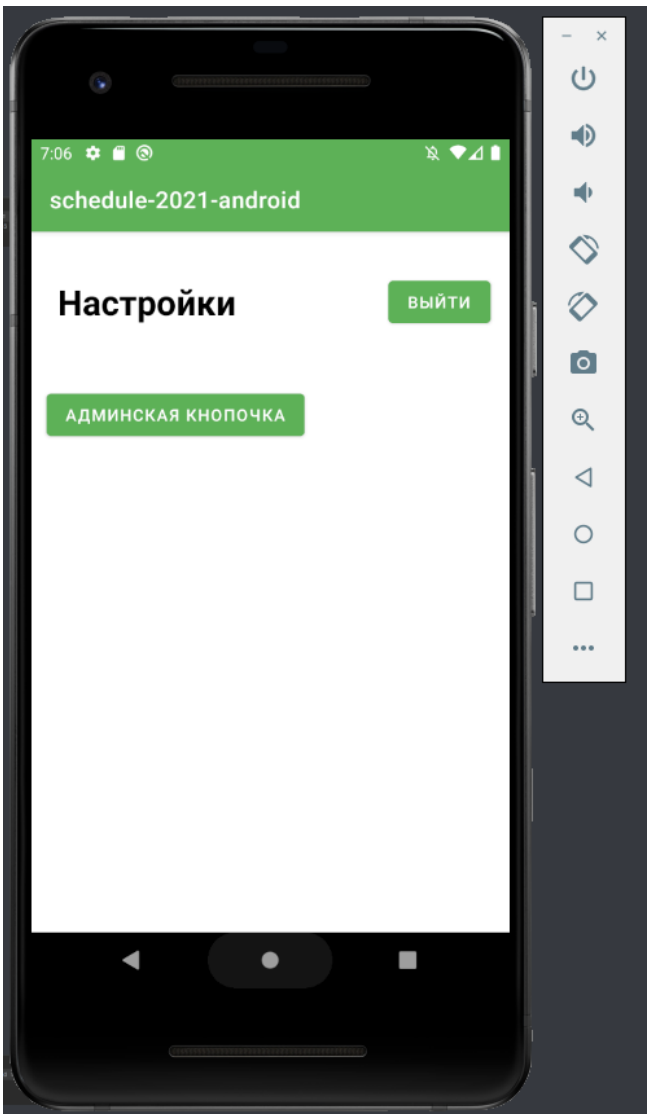


Рис. 5.22. Экран настроек для пользователя admin1

Можно заметить, что у пользователя admin1 есть админская кнопка, поскольку у пользователя есть права админа. Нажмем на нее/ Открывается экран регистрации:

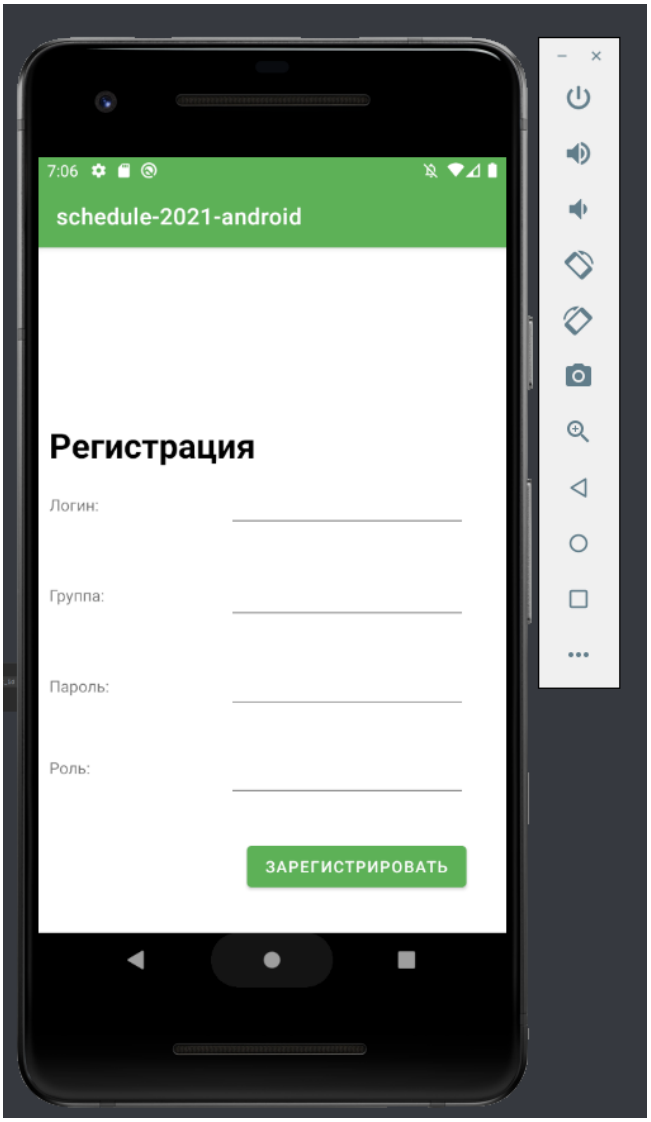


Рис. 5.23. Экран регистрации

Попробуем зарегистрировать нового пользователя с правами обычного студента:

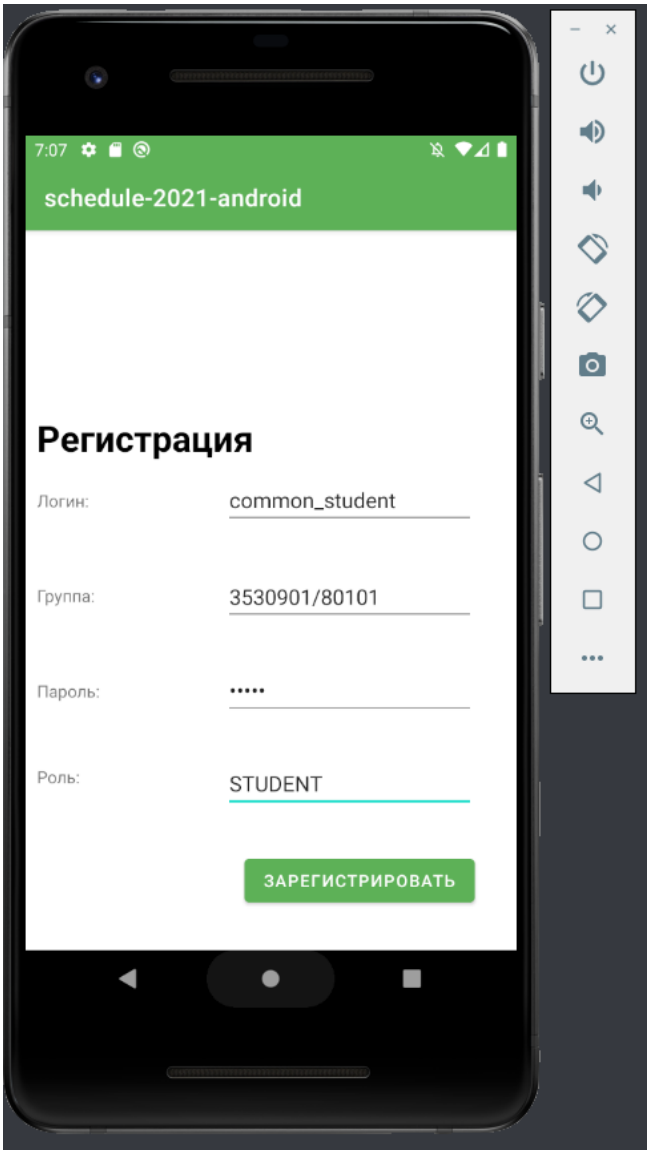


Рис. 5.24. Экран регистрации после заполнения полей

Теперь посмотрим состояние таблицы студентов:

WHERE ORDER BY id DESC

id	name	group_id	password	role
505	common_student	1408	$2a$10$MgeyjWU5rvVa1Y3EzDAtHOy2b3lJXM6eo4q5agqplO4DVuRYldDR2	STUDENT
504	telepova.mp	624	$2a$10$NA01z/qyTacP8RIkOjtMiuons3XayCEvbyYY3jigELmjzgqOBXdga	STUDENT
503	admin1	624	$2a$10$WK2c/zUIvxgazTPWSs1qv.uDz93OFCqBWflN/vgz73ufeg3x7q2Dy	ADMIN

Рис. 5.25. Таблица студентов после регистрации нового пользователя

Можно заметить, что новый студент был зарегистрирован успешно. Попробуем залогиниться за него и проверить, как выглядят настройки у обычного пользователя:

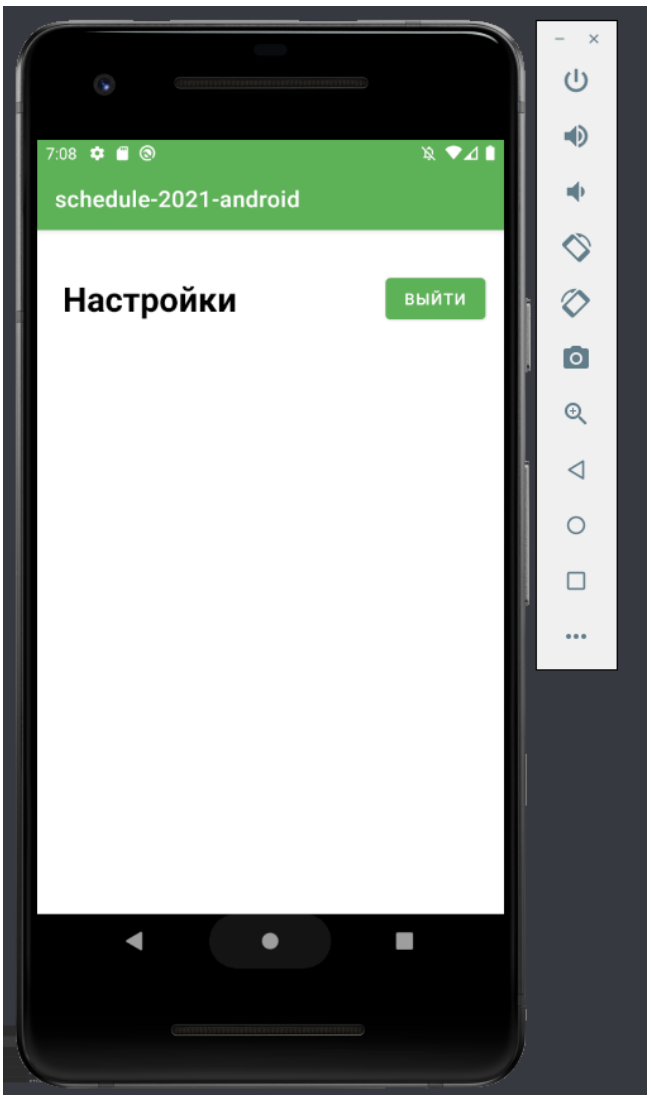


Рис. 5.26. Экран настроек пользователя common_student

Значит, у обычного нет доступа к админской кнопке, а значит, нет возможности регистрировать новых пользователей.

Таким образом, во время тестирования была доказана работоспособность основного функционала приложения.

5.3.5. Демонстрация результатов преподавателю

Результаты реализации технического задания были продемонстрированы на занятии 17 мая Мяснову Александру Владимировичу, за данную презентацию была получена оценка "зачет".

5.4. Выводы

В данной курсовой работе мы выбрали и согласовали способ реализации курсовой работы, написали и согласовали техническое задание по этой работе, реализовали всю требуемую функциональность, протестировали корректность работы программы и продемонстрировали готовую работу преподавателю. Были

систематизированы и углублены полученные знания и самостоятельно изучен и применен на практике ряд избранных вопросов. За данную работу была получена оценка "зачет".