Оглавление

[Введение 4](#_Toc74686158)

[Глава 1. Анализ предметной области 5](#_Toc74686159)

[**1.1 Теория баз данных 5**](#_Toc74686160)

[**1.2 СУБД PostgreSQL 6**](#_Toc74686161)

[**1.3 Среда разработки Qt Creator 6**](#_Toc74686162)

[**1.4 Предметная область 6**](#_Toc74686163)

[Глава 2. Алгоритм решения задач 8](#_Toc74686164)

[**2.1 Выбор средств реализации 8**](#_Toc74686165)

[**2.2 Серверная часть 8**](#_Toc74686166)

[**2.3 Клиентская часть 19**](#_Toc74686167)

[Глава 3. Контрольный пример 32](#_Toc74686168)

[Заключение 48](#_Toc74686169)

[Список использованной литературы 49](#_Toc74686170)

# Введение

Самым современным средством для накопления и поиска информации в компьютерах являются базы данных, позволяющие организовать хранение большого объёма информации с обеспечением высокой скорости поиска и обновления.

За десятилетие, а особенно за последние несколько лет, человечество сгенерировало очень много данных. Их количество растет в геометрической прогрессии, поэтому в настоящее время актуальность баз данных возрастает как никогда. Сбор, хранение, быстрый и надёжный поиск нужной информации в разных областях нашей жизни, необходим практически на любом этапе работы с информацией.

Целью курсовой работы является создание клиент-серверного приложения, серверная часть которого реализована на PostgreSQL и представляющая собой организацию учебной работы преподавателей вуза, а клиентская в Qt Creator . В рамках заданной предметной области необходимо реализовать заданную схему отношений, выделив сущности и их атрибуты, так чтобы связи между сущностями соответствовали схеме. А также в рамках курсовой работы необходимо на стороне сервера реализовать и использовать при демонстрации приложения следующие компоненты: постоянные таблицы, запросы, индексы, триггеры, функции, хранимые процедуры и пользовательские роли. Для того, чтобы достичь данной цели, необходимо решить следующие задачи:

1. Собрать информацию о предметной области;
2. Реализовать серверную часть согласно требованиям;
3. Реализовать клиентскую часть;
4. Разработать контрольный пример, демонстрирующий работоспособность информационной базы и приложения.

# Глава 1. Анализ предметной области

## 1.1 Теория баз данных

База данных – поименованная и организованная совокупность взаимосвязанных данных, которые отражают состояние объектов конкретной предметной области и находятся под центральным программным управлением.

Предметная область – часть реальной среды, которая описывается и отражается в БД. Это информационная сторона функционирования автоматизированной системы, отражающая множество объектов и связей между ними.

Отличительной чертой баз данных следует считать то, что данные хранятся совместно с их описанием, а в прикладных программах описание данных не содержится. Независимые от программ пользователя данные обычно называются метаданными. В ряде современных систем метаданные, содержащие также информацию о пользователях, форматы отображения, статистику обращения к данным и др. сведения, хранятся в словаре базы данных.

Исторически первые базы данных создавались на основе файловых систем, и вся ответственность за работу с ними возлагалась на прикладное программное обеспечение, использовавшее эти базы. Файловые базы данных сегодня практически не применяются. В современной технологии баз данных предполагается, что создание базы данных, ее поддержка и обеспечение доступа пользователей к ней осуществляются централизованно с помощью специального программного инструментария — системы управления базами данных.

Система управления базами данных (СУБД) – это совокупность языковых и программных средств, предназначенных для создания, ведения и совместного использования БД многими пользователями.

Кроме базы данных и программного обеспечения, обеспечивающего основную функциональность СУБД, в состав современных серверов баз данных входят всевозможные средства разработки и механизмы взаимодействия с пользователем на высоком уровне (генераторы отчетов, конструкторы таблиц, построители запросов и форм). Эти средства разработки, сами являясь приложениями пользователя, позволяют создавать приложения, функционирующие как часть СУБД (например, формы и отчеты MS Access или веб-публикации в Oracle и MS SQL Server).

## 1.2 СУБД PostgreSQL

PostgreSQL — это система управления базами данных с открытым исходным кодом корпоративного класса. Он поддерживает как SQL для реляционных, так и JSON для нереляционных запросов. Он поддерживается опытным сообществом разработчиков, которые внесли огромный вклад в создание высоконадежной системы СУБД.

PostgreSQL (первоначально назывался Postgres) был создан профессором информатики Майклом Стоунбрейкером и его командой. Сегодня это стало одной из популярных баз данных с открытым исходным кодом.

## 1.3 Среда разработки Qt Creator

Qt Creator (ранее известная под кодовым названием Greenhouse) — кроссплатформенная свободная IDE для разработки на С, С++ и QML. Разработана Trolltech (Digia) для работы с фреймворком Qt. Включает в себя графический интерфейс отладчика и визуальные средства разработки интерфейса как с использованием QtWidgets, так и QML.

QML (Qt Meta Language или Qt Modeling Language) — декларативный язык программирования, в основании которого лежит среда JavaScript. QML используется для разработки приложений, делающих основной упор на пользовательский интерфейс и, в целом, на дизайн графической части. Является частью Qt Quick, среды разработки пользовательского интерфейса, распространяемой вместе с Qt.

## 1.4 Предметная область

Предметная область представляет собой организацию работы преподавателей вуза. В ней выделено две основных роли: преподаватель и кафедра.

Преподаватели могут отслеживать свою учебную нагрузку, а также видеть информацию по дисциплинам, которые они ведут. Кроме этого, преподаватель может передать часть своей нагрузки другому преподавателю с кафедры, если последний имеет мало нагрузки.

Кафедра же имеет полный доступ к изменению, удалению и обновлению информации о преподавателях с кафедры, её дисциплинах и нагрузке преподавателей кафедры. Добавление, удаление и обновление информации о кафедрах вуза происходит тоже через эту роль. А также кафедра имеет доступ к результатам запросов.

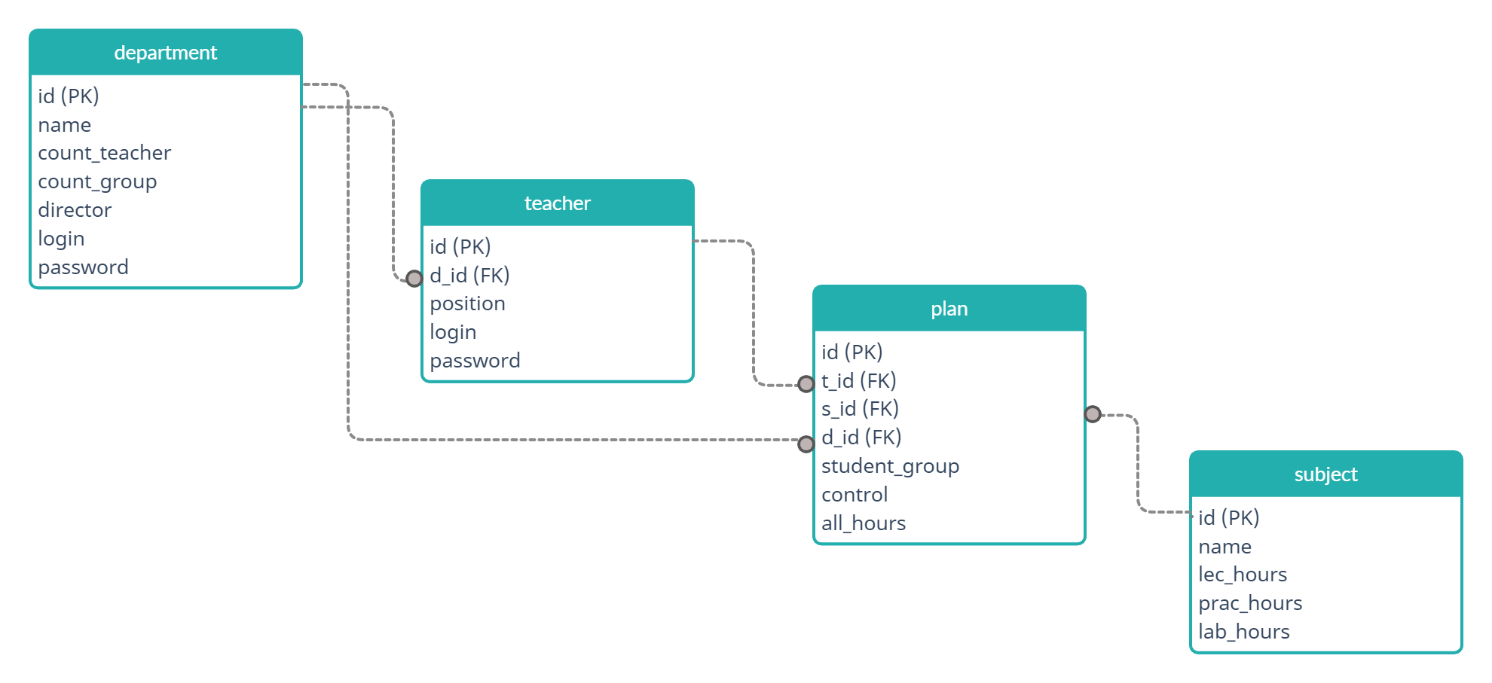
Сущностей в предметной области 4. Это кафедра, преподаватель, план преподавателя и дисциплина.

Кафедра имеет следующие атрибуты: код кафедры, наименование, число преподавателей на кафедре, число групп на кафедре, кто заведующий кафедрой, логин и пароль.

Атрибуты преподавателя: ФИО, должность, логин, пароль, кафедра.

Атрибуты плана преподавателя: номер нагрузки преподавателя, ФИО, дисциплина, группа студентов, кафедра группы, форма контроля, сколько часов дисциплины.

Схема отношений:



# Глава 2. Алгоритм решения задач

## 2.1 Выбор средств реализации

Для реализации серверной части приложения была выбрана СУБД PostgreSQL, а для реализации клиентской части - среда разработки Qt Creator.

## 2.2 Серверная часть

В первую очередь я создала таблицы согласно варианту (листинг 1).

CREATE TABLE mydb.department(

id varchar(20) NOT NULL PRIMARY KEY,

name varchar(50) NOT NULL,

count\_teacher int NOT NULL DEFAULT 0,

count\_group int NOT NULL DEFAULT 0,

director varchar(50) NULL,

login varchar(20) NOT NULL,

password varchar(20) NOT NULL);

CREATE TABLE mydb.teacher(

id varchar(50) NOT NULL PRIMARY KEY,

position varchar(30) NOT NULL,

login varchar(20) NOT NULL,

password varchar(20) NOT NULL,

d\_id varchar(20) NULL,

FOREIGN KEY (d\_id) REFERENCES mydb.department(id));

CREATE TABLE mydb.plan(

id serial NOT NULL PRIMARY KEY,

t\_id varchar(50) NULL,

s\_id varchar(20) NULL,

d\_id varchar(20) NULL,

student\_group varchar(20) NOT NULL,

control varchar(20) NOT NULL,

all\_hours int NULL,

FOREIGN KEY (d\_id) REFERENCES mydb.department(id),

FOREIGN KEY (t\_id) REFERENCES mydb.teacher(id),

FOREIGN KEY (s\_id) REFERENCES mydb.subject(id));

CREATE TABLE mydb.subject(

id varchar(20) NOT NULL PRIMARY KEY,

name varchar(50) NOT NULL,

lec\_hours int NOT NULL,

prac\_hours int NOT NULL,

lab\_hours int NOT NULL);

Листинг 1. Создание таблиц.

Далее я написала требуемые в задании запросы:

Составной многотабличный запрос с параметром, включающий соединение таблиц и CASE-выражение, который выводит дисциплины и сколько часов по ним, если часов 0, то выводит надпись, что нет занятий такого типа (листинг 2);

SELECT s.name, p.all\_hours,

CASE

WHEN s.lec\_hours=0

THEN 'Нет занятий'

ELSE CAST(s.lec\_hours AS varchar(3))

END lec\_hours,

CASE

WHEN s.prac\_hours=0

THEN 'Нет занятий'

ELSE CAST(s.prac\_hours AS varchar(3))

END prac\_hours,

CASE

WHEN s.lab\_hours=0

THEN 'Нет занятий'

ELSE CAST(s.lab\_hours AS varchar(3))

END lab\_hours

FROM mydb.subject s

JOIN mydb.plan p ON s.id = p.s\_id

GROUP BY s.name, p.all\_hours,s.lec\_hours,s.prac\_hours,s.lab\_hours;

Листинг 2. Запрос 2a CASE.

Запрос на основе обновляющего представления (многотабличного VIEW), в котором критерий упорядоченности задает пользователь при выполнении (листинг 3);

CREATE TRIGGER update\_view

INSTEAD OF UPDATE ON mydb.work\_view

FOR EACH ROW

EXECUTE PROCEDURE mydb.v\_update();

CREATE OR REPLACE FUNCTION mydb.v\_update()

RETURNS TRIGGER AS $$

BEGIN

IF(NEW.fio <> OLD.fio)

THEN UPDATE mydb.teacher SET id=NEW.fio WHERE id=OLD.fio;

END IF;

IF(NEW.position <> OLD.position)

THEN UPDATE mydb.teacher SET position=NEW.posithion WHERE id=OLD.fio;

END IF;

IF(NEW.subject <> OLD.subject)

THEN UPDATE mydb.subject SET name=NEW.subject WHERE id=OLD.s\_id;

END IF;

IF(NEW.student\_group <> OLD.student\_group)

THEN UPDATE mydb.plan SET student\_group=NEW.student\_group WHERE id=OLD.p\_id;

END IF;

IF(NEW.control <> OLD.control)

THEN UPDATE mydb.plan SET control=NEW.control WHERE id=OLD.p\_id;

END IF;

IF(NEW.all\_hours <> OLD.all\_hours)

THEN UPDATE mydb.plan SET all\_hours=NEW.all\_hours WHERE id=OLD.p\_id;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE VIEW mydb.work\_view AS

SELECT t.id AS fio, t.position,s.id AS s\_id, s.name AS subject, p.id AS p\_id, p.student\_group, p.control,p.all\_hours

FROM mydb.plan p

JOIN mydb.teacher t ON t.id = p.t\_id

JOIN mydb.subject s ON s.id = p.s\_id;

Листинг 3. Запрос 2b VIEW и триггер для обновления представления.

Запрос, содержащий некоррелированный подзапрос в разделе SELECT и выводящий среднее количество часов (листинг 4);

Select ROUND(AVG(p.all\_hours),1),

(Select ROUND(AVG(s.lec\_hours),1) FROM mydb.subject s),

(Select ROUND(AVG(s.prac\_hours),1) FROM mydb.subject s),

(Select ROUND(AVG(s.lab\_hours),1) FROM mydb.subject s)

FROM mydb.plan p;

Листинг 4. Запрос 2c некор. SELECT.

Запрос, содержащий некоррелированный подзапрос в разделе FROM и выводящий нагрузку заведующего кафедрой (листинг 5);

SELECT d.name,t.id,t.position,p.s\_id,p.student\_group

FROM (SELECT id, position, d\_id FROM mydb.teacher WHERE position='Заведующий кафедрой') AS t

JOIN mydb.plan p ON t.id=p.t\_id

JOIN mydb.department d ON d.id=t.d\_id

WHERE d.id='КБ-1';

Листинг 5. Запрос 2c некор. FROM.

Запрос, содержащий некоррелированный подзапрос WHERE и выводящий дисциплины, у которых общее кол-во часов меньше среднего (листинг 6);

Select s.name, p.all\_hours

FROM mydb.subject s

JOIN mydb.plan p ON s.id=p.s\_id

WHERE p.all\_hours<(select AVG(p.all\_hours) FROM mydb.plan p)

GROUP BY s.name,p.all\_hours;

Листинг 6. Запрос 2c некор. WHERE.

Запрос, содержащий коррелированный подзапрос в разделе FROM и выводящий сколько нагрузки у всех преподавателей с кафедры вместе(листинг 7);

Select t.d\_id, SUM(pp.c)

FROM mydb.teacher t,

LATERAL(select COUNT(\*) AS c FROM mydb.plan p WHERE p.t\_id=t.id) pp

GROUP BY t.d\_id;

Листинг 7. Запрос 2c кор. FROM.

Запрос, содержащий коррелированный подзапрос в разделе SELECT и выводящий сколько нагрузки у каждого преподавателя (листинг 8);

SELECT t.id, t.d\_id, (SELECT COUNT(\*) FROM mydb.plan p WHERE t.id=p.t\_id) AS sum\_plan

FROM mydb.teacher t

ORDER BY sum\_plan DESC;

Листинг 8. Запрос 2c кор. SELECT.

Запрос, содержащий коррелированный подзапрос в разделе WHERE и выводящий тех, кто на кафедре не имеет нагрузки (листинг 9);

Select t.d\_id, t.id

FROM mydb.teacher t

WHERE (select COUNT(\*) FROM mydb.plan p WHERE p.t\_id=t.id)=0

ORDER BY t.d\_id;

Листинг 9. Запрос 2c кор. WHERE.

Многотабличный запрос, содержащий группировку записей, агрегативные функции и параметр, используемый в разделе HAVING, выводящий преподавателей, у которых количество нагрузки меньше 10 (листинг 10);

SELECT t.id AS name, COUNT(\*)

FROM mydb.plan p

JOIN mydb.teacher t ON p.t\_id=t.id

GROUP BY t.id

HAVING COUNT(\*)<10;

Листинг 10. Запрос 2d HAVING.

Запрос, содержащий предикат ANY, выводящий преподавателей, которые ведут дисциплину РПОЗОС студентам с кафедры КБ-2 (листинг 11).

SELECT t.id AS name

FROM mydb.teacher t

WHERE 'РПОЗОС'=any

(SELECT p.s\_id

FROM mydb.plan p

WHERE t.id=p.t\_id AND p.d\_id='КБ-2');

Листинг 11. Запрос 2e ANY.

Создала индексы для увеличения скорости выполнения запросов (листинг 12).

CREATE INDEX idx\_c\_t ON mydb.department(count\_teacher);

CREATE INDEX idx\_c\_g ON mydb.department(count\_group);

CREATE INDEX idx\_director ON mydb.department(director);

CREATE INDEX idx\_s\_g ON mydb.plan(student\_group);

CREATE INDEX idx\_all\_h ON mydb.plan(all\_hours);

CREATE INDEX idx\_lec\_h ON mydb.subject(lec\_hours);

CREATE INDEX idx\_pr\_h ON mydb.subject(prac\_hours);

CREATE INDEX idx\_lab\_h ON mydb.subject(lab\_hours);

Листинг 12. Индексы.

Разработала четыре триггера, которые заполняют предусмотренное для каждого поле автоматически при добавлении, обновлении и удалении данных.

Первый триггер изменяет поле количества преподавателей на кафедре (листинг 13).

CREATE TRIGGER teacher\_counter

AFTER INSERT OR UPDATE OR DELETE ON mydb.teacher FOR EACH ROW EXECUTE

PROCEDURE mydb.show\_count\_teacher();

CREATE OR REPLACE FUNCTION mydb.show\_count\_teacher() RETURNS TRIGGER AS $$

BEGIN

IF TG\_OP = 'INSERT' THEN

UPDATE mydb.department SET count\_teacher = count\_teacher+1

WHERE mydb.department.id = NEW.d\_id;

RETURN NEW;

ELSIF TG\_OP = 'UPDATE' THEN

UPDATE mydb.department SET count\_teacher = count\_teacher+1

WHERE mydb.department.id = NEW.d\_id;

UPDATE mydb.department SET count\_teacher = count\_teacher-1

WHERE mydb.department.id = OLD.d\_id;

RETURN NEW;

ELSIF TG\_OP = 'DELETE' THEN

UPDATE mydb.department SET count\_teacher = count\_teacher-1

WHERE mydb.department.id = OLD.d\_id;

RETURN OLD;

END IF;

END;

$$ LANGUAGE plpgsql;

Листинг 13. Триггер для поля count\_teacher.

Второй триггер изменяет поле количества групп на кафедре (листинг 14).

CREATE TRIGGER group\_counter

AFTER INSERT OR UPDATE OR DELETE ON mydb.plan FOR EACH ROW EXECUTE

PROCEDURE mydb.show\_count\_group();

CREATE OR REPLACE FUNCTION mydb.show\_count\_group() RETURNS TRIGGER AS $$

DECLARE

new\_sg varchar;

old\_sg varchar;

BEGIN

new\_sg=NEW.student\_group;

old\_sg=OLD.student\_group;

IF TG\_OP = 'INSERT' THEN

UPDATE mydb.department SET count\_group = count\_group+1

WHERE mydb.department.id = NEW.d\_id AND new\_sg<>all

(SELECT p.student\_group FROM mydb.plan p

WHERE p.id<>NEW.id);

RETURN NEW;

ELSIF TG\_OP = 'UPDATE' THEN

UPDATE mydb.department SET count\_group = count\_group+1

WHERE mydb.department.id = NEW.d\_id AND new\_sg<>all

(SELECT p.student\_group FROM mydb.plan p

WHERE p.id<>NEW.id);

UPDATE mydb.department SET count\_group = count\_group-1

WHERE mydb.department.id = OLD.d\_id AND new\_sg<>all

(SELECT p.student\_group FROM mydb.plan p

WHERE p.id<>NEW.id);

RETURN NEW;

ELSIF TG\_OP = 'DELETE' THEN

UPDATE mydb.department SET count\_group = count\_group-1

WHERE mydb.department.id = OLD.d\_id AND old\_sg<>all

(SELECT p.student\_group FROM mydb.plan p);

RETURN OLD;

END IF;

END;

$$ LANGUAGE plpgsql;

Листинг 14. Триггер для поля count\_group.

Третий триггер изменяет поле заведующего кафедрой (листинг 15).

CREATE TRIGGER set\_director

AFTER INSERT OR UPDATE OR DELETE ON mydb.teacher FOR EACH ROW EXECUTE

PROCEDURE mydb.show\_director();

CREATE OR REPLACE FUNCTION mydb.show\_director() RETURNS TRIGGER AS $$

BEGIN

IF TG\_OP = 'INSERT' THEN

UPDATE mydb.department SET director = NEW.id

WHERE NEW.position='Заведующий кафедрой' AND NEW.d\_id=id;

RETURN NEW;

ELSIF TG\_OP = 'UPDATE' THEN

UPDATE mydb.department SET director = NEW.id

WHERE NEW.position='Заведующий кафедрой' AND NEW.d\_id=id;

UPDATE mydb.department SET director = NULL

WHERE OLD.position = 'Заведующий кафедрой' AND NEW.position<>'Руководитель кафедры' AND NEW.d\_id=id;

RETURN NEW;

ELSIF TG\_OP = 'DELETE' THEN

UPDATE mydb.department SET director = NULL

WHERE OLD.position = 'Заведующий кафедрой' AND NEW.d\_id=id;

RETURN OLD;

END IF;

END;

$$ LANGUAGE plpgsql;

Листинг 15. Триггер для поля director.

Четвёртый триггер изменяет поле на сколько часов рассчитана дисциплина (листинг 16).

CREATE TRIGGER set\_all\_hours

AFTER INSERT OR UPDATE OR DELETE ON mydb.subject FOR EACH ROW EXECUTE

PROCEDURE mydb.set\_hours();

CREATE OR REPLACE FUNCTION mydb.set\_hours() RETURNS TRIGGER AS $$

BEGIN

IF TG\_OP = 'INSERT' THEN

UPDATE mydb.plan SET all\_hours = mydb.hours\_calc(NEW.lec\_hours, NEW.prac\_hours, NEW.lab\_hours)

WHERE NEW.id=s\_id;

RETURN NEW;

ELSIF TG\_OP = 'UPDATE' THEN

UPDATE mydb.plan SET all\_hours = mydb.hours\_calc(NEW.lec\_hours, NEW.prac\_hours, NEW.lab\_hours)

WHERE NEW.id=s\_id;

UPDATE mydb.plan SET all\_hours = NULL

WHERE OLD.id=s\_id AND NEW.id<>s\_id;

RETURN NEW;

ELSIF TG\_OP = 'DELETE' THEN

UPDATE mydb.plan SET all\_hours = NULL

WHERE OLD.id=s\_id;

RETURN OLD;

END IF;

END;

$$ LANGUAGE plpgsql;

Листинг 16. Триггер для поля all\_hours.

Реализовала операции добавления, удаления и обновления в виде хранимых процедур с параметрами для всех таблиц (листинг 17).

CREATE OR REPLACE PROCEDURE mydb.update\_department(id varchar(20), name varchar(50), login varchar(20), password varchar(20))

LANGUAGE plpgsql AS $$

BEGIN

UPDATE mydb.department d

SET id = COALESCE(update\_department.id, d.id),

name = COALESCE(update\_department.name, d.name),

login = COALESCE(update\_department.login, d.login),

password = COALESCE(update\_department.password, d.password)

WHERE d.id=update\_department.id;

END

$$;

CREATE OR REPLACE PROCEDURE mydb.update\_teacher(id varchar(50), tposition varchar(30), login varchar(20), password varchar(20), d\_id varchar(20))

LANGUAGE plpgsql AS $$

BEGIN

UPDATE mydb.teacher t

SET id = COALESCE(update\_teacher.id, t.id),

position = COALESCE(update\_teacher.tposition, t.position),

login = COALESCE(update\_teacher.login, t.login),

password = COALESCE(update\_teacher.password, d.password),

d\_id = COALESCE(update\_teacher.d\_id, t.d\_id)

WHERE t.id=update\_teacher.id;

END

$$;

CREATE OR REPLACE PROCEDURE mydb.update\_plan(id int, t\_id varchar(50), s\_id varchar(20), d\_id varchar(20), student\_group varchar(20), control varchar(20))

LANGUAGE plpgsql AS $$

BEGIN

UPDATE mydb.plan p

SET t\_id = COALESCE(update\_plan.t\_id, p.t\_id),

s\_id = COALESCE(update\_plan.s\_id, p.s\_id),

d\_id = COALESCE(update\_plan.d\_id, p.d\_id),

student\_group = COALESCE(update\_plan.student\_group, p.student\_group),

control = COALESCE(update\_plan.control, p.control)

WHERE p.id=update\_plan.id;

END

$$;

CREATE OR REPLACE PROCEDURE mydb.update\_subject(id varchar(20), name varchar(50), lec\_hours int, prac\_hours int, lab\_hours int)

LANGUAGE plpgsql AS $$

BEGIN

UPDATE mydb.subject s

SET id = COALESCE(update\_subject.id, s.id),

name = COALESCE(update\_subject.name, s.name),

lec\_hours = COALESCE(update\_subject.lec\_hours, s.lec\_hours),

prac\_hours = COALESCE(update\_subject.prac\_hours, s.prac\_hours),

lab\_hours =COALESCE(update\_subject.lab\_hours, s.lab\_hours)

WHERE s.id=update\_subject.id;

END

$$;

CREATE OR REPLACE PROCEDURE mydb.insert\_department(id varchar(20), name varchar(50), login varchar(20), password varchar(20))

LANGUAGE plpgsql AS $$

BEGIN

INSERT INTO mydb.department(id, name, login, password)

VALUES (insert\_department.id, insert\_department.name, insert\_department.login, insert\_department.password);

END

$$;

CREATE OR REPLACE PROCEDURE mydb.insert\_teacher(id varchar(50), tposition varchar(30), login varchar(20), password varchar(20), d\_id varchar(20))

LANGUAGE plpgsql AS $$

BEGIN

INSERT INTO mydb.teacher(id, position, login, password, d\_id)

VALUES (insert\_teacher.id, insert\_teacher.tposition, insert\_teacher.login, insert\_teacher.password, insert\_teacher.d\_id);

END

$$;

CREATE OR REPLACE PROCEDURE mydb.insert\_plan(t\_id varchar(50), s\_id varchar(20), d\_id varchar(20), student\_group varchar(20), control varchar(20),lec int,prac int,lab int)

LANGUAGE plpgsql AS $$

BEGIN

INSERT INTO mydb.plan(t\_id, s\_id, d\_id, student\_group, control,all\_hours)

VALUES (insert\_plan.t\_id, insert\_plan.s\_id, insert\_plan.d\_id, insert\_plan.student\_group, insert\_plan.control,mydb.hours\_calc(lec, prac, lab));

END

$$;

CREATE OR REPLACE PROCEDURE mydb.insert\_subject(id varchar(20), name varchar(50), lec\_hours int, prac\_hours int, lab\_hours int)

LANGUAGE plpgsql AS $$

BEGIN

INSERT INTO mydb.subject(id, name, lec\_hours,prac\_hours,lab\_hours)

VALUES (insert\_subject.id, insert\_subject.name, insert\_subject.lec\_hours, insert\_subject.prac\_hours, insert\_subject.lab\_hours);

END

$$;

CREATE OR REPLACE PROCEDURE mydb.delete\_department(del\_id varchar(20))

LANGUAGE plpgsql AS $$

BEGIN

DELETE FROM mydb.department WHERE id=del\_id;

END

$$;

CREATE OR REPLACE PROCEDURE mydb.delete\_teacher(del\_id varchar(20))

LANGUAGE plpgsql AS $$

BEGIN

DELETE FROM mydb.teacher WHERE id=del\_id;

END

$$;

CREATE OR REPLACE PROCEDURE mydb.delete\_plan(del\_id int)

LANGUAGE plpgsql AS $$

BEGIN

DELETE FROM mydb.plan WHERE id=del\_id;

END

$$;

CREATE OR REPLACE PROCEDURE mydb.delete\_subject(del\_id varchar(20))

LANGUAGE plpgsql AS $$

BEGIN

DELETE FROM mydb.subject WHERE id=del\_id;

END

$$;

Листинг 17. Хранимые процедуры для добавления, удаления и обновления данных.

Потом я реализовала транзакцию, которая по нажатию соответствующей кнопки удаляет у одного преподавателя с кафедры строку нагрузки и добавляет её другому преподавателю кафедры, у которого нагрузка меньше 10. Если такого нет, транзакция будет откатана (листинг 18). Кроме того, в этой транзакции я реализовала курсор.

CREATE OR REPLACE PROCEDURE mydb.move(dep varchar(20), id int, t\_id varchar(50),s\_id varchar(20),d\_id varchar(20),student\_group varchar(20),control varchar(20), all\_hours int)

LANGUAGE plpgsql AS $$

DECLARE

flag integer=0;

teacher\_id record;

teacher\_array CURSOR FOR(SELECT t.id FROM mydb.teacher t

WHERE t.id<>move.t\_id AND t.d\_id=move.dep);

BEGIN

DELETE FROM mydb.plan pl WHERE pl.id=move.id;

FOR teacher\_id IN teacher\_array

LOOP

IF(SELECT COUNT(\*) FROM mydb.plan p

GROUP BY p.t\_id

HAVING p.t\_id=teacher\_id.id)<10

THEN flag=1;

INSERT INTO mydb.plan( t\_id, s\_id, d\_id, student\_group, control, all\_hours)

VALUES (teacher\_id.id,move.s\_id,move.d\_id,move.student\_group,move.control,move.all\_hours);

END IF;

EXIT WHEN flag=1;

END LOOP;

IF flag=0

THEN RAISE NOTICE 'Нет возможности отдать';

ROLLBACK;

END IF;

END;

$$;

Листинг 18. Транзакция.

Далее я создала собственную скалярную и векторную функции (листинг 19). Первая высчитывает общее кол-во часов дисциплины, а вторая возвращает группы, которым ведут пары преподаватели с кафедры.

CREATE OR REPLACE FUNCTION mydb.hours\_calc(lec int, prac int, lab int)

RETURNS int AS $$

BEGIN

RETURN (lec + prac + lab)\*1.5;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION mydb.show\_group(dep text)

RETURNS setof varchar(20) AS $$

BEGIN

RETURN QUERY

SELECT p.student\_group

FROM mydb.plan p

JOIN mydb.teacher t on t.id=p.t\_id

WHERE t.d\_id=dep

GROUP BY p.student\_group;

END

$$ LANGUAGE plpgsql;

Листинг 19. Скалярная и векторная функция.

И в завершение серверной части приложения я создала пользовательские роли с различными привилегиями (листинг 20). Первая роль – гостевая. Через неё подключаемся к БД до того, как пользователь вошёл в аккаунт. Эта роль имеет минимальное количество привилегий. Вторая роль – это роль административная. Под этой ролью входит кафедра, и эта роль имеет больше всего прав. Третья роль – пользовательская. Под ней входят преподаватели. Роль позволяет выводить информацию из таблиц и производить транзакцию.

CREATE ROLE visitor WITH LOGIN;

ALTER ROLE visitor WITH PASSWORD 'visitor';

GRANT USAGE ON SCHEMA mydb TO visitor;

GRANT SELECT ON mydb.department TO visitor;

GRANT SELECT ON mydb.teacher TO visitor;

CREATE ROLE kafedra WITH LOGIN;

ALTER ROLE kafedra WITH PASSWORD 'kafedra';

GRANT USAGE ON SCHEMA mydb TO kafedra;

GRANT ALL ON ALL TABLES IN SCHEMA mydb TO kafedra;

GRANT USAGE, SELECT ON ALL SEQUENCES IN SCHEMA mydb TO kafedra;

CREATE ROLE teacher WITH LOGIN;

ALTER ROLE teacher WITH PASSWORD 'teacher';

GRANT USAGE ON SCHEMA mydb TO teacher;

GRANT SELECT ON ALL TABLES IN SCHEMA mydb TO teacher;

GRANT DELETE ON mydb.plan TO teacher;

GRANT INSERT ON mydb.plan TO teacher;

GRANT UPDATE ON mydb.department TO teacher;

GRANT USAGE, SELECT ON ALL SEQUENCES IN SCHEMA mydb TO teacher;

Листинг 20. Пользовательские роли.

## 2.3 Клиентская часть

В начале я реализовала вход в БД через роли, в зависимости от логина и пароля открываются соответствующие формы(листинг 21).

QSqlDatabase db = QSqlDatabase::addDatabase("QPSQL");

db.setDatabaseName("postgres");

db.setHostName("localhost");

db.setUserName("visitor");

db.setPassword("visitor");

if (!db.open())

{

QMessageBox::warning(this, "Error", db.lastError().text());

}

QString login=ui->lineLogin->text();

QString password=ui->linePassword->text();

QSqlQuery query;

query.exec("SELECT login, password FROM mydb.department;");

bool flag = false;

QString userRole;

while (query.next())

{

if((login == query.value(0).toString()) && (password == query.value(1).toString()))

{

userRole="kafedra";

flag=true;

db.close();

db.setDatabaseName("postgres");

db.setHostName("localhost");

db.setUserName(userRole);

db.setPassword(userRole);

db.open();

admin \*a=new admin(login);

a->show();

this->close();

}

}

if(!flag)

{

query.exec("SELECT login, password FROM mydb.teacher;");

while (query.next())

{

if((login == query.value(0).toString()) && (password == query.value(1).toString()))

{

userRole="teacher";

flag=true;

db.close();

db.setDatabaseName("postgres");

db.setHostName("localhost");

db.setUserName(userRole);

db.setPassword(userRole);

db.open();

client \*c=new client(login);

c->show();

this->close();

}

}

}

if(!flag) QMessageBox::information(this, "ERROR", "Неверные данные");

Листинг 21. Вход в БД под соответствующими ролями.

При входе в преподавательский аккаунт открывается форма с информацией о преподавателе и таблицами с нагрузкой и дисциплинами. Форма заполняется следующим образом (листинг 22):

ui->setupUi(this);

query.exec(QString("SELECT t.id,t.position,d.name,d.id FROM mydb.teacher t JOIN mydb.department d ON d.id = t.d\_id WHERE t.login='%1';").arg(login));

query.first();

ui->label\_fio->setText(query.value(0).toString());

ui->label\_position->setText("Должность: "+query.value(1).toString());

ui->label\_dep->setText("Кафедра: "+query.value(2).toString());

department = query.value(3).toString();

fio = query.value(0).toString();

model1->setQuery(QString("SELECT s.id, s.name,p.student\_group,p.d\_id,p.control,p.all\_hours,p.id FROM mydb.plan p JOIN mydb.subject s ON s.id = p.s\_id JOIN mydb.teacher t ON t.id = p.t\_id WHERE t.login='%1';").arg(login));

model1->*setHeaderData*(0, Qt::Horizontal, tr("Сокращение"));

model1->*setHeaderData*(1, Qt::Horizontal, tr("Предмет"));

model1->*setHeaderData*(2, Qt::Horizontal, tr("Группа"));

model1->*setHeaderData*(3, Qt::Horizontal, tr("Кафедра группы"));

model1->*setHeaderData*(4, Qt::Horizontal, tr("Форма контроля"));

model1->*setHeaderData*(5, Qt::Horizontal, tr("Часы"));

model1->*setHeaderData*(6, Qt::Horizontal, tr("План"));

ui->tableView1->*setModel*(model1);

ui->tableView1->setColumnHidden(0, true);

ui->tableView1->setColumnHidden(6, true);

ui->tableView1->show();

model2->setQuery(QString("SELECT s.name,s.lec\_hours,s.prac\_hours,s.lab\_hours FROM mydb.plan p JOIN mydb.subject s ON s.id = p.s\_id JOIN mydb.teacher t ON t.id = p.t\_id WHERE t.login='%1' GROUP BY s.id;").arg(login));

model2->*setHeaderData*(0, Qt::Horizontal, tr("Предмет"));

model2->*setHeaderData*(1, Qt::Horizontal, tr("Лекции"));

model2->*setHeaderData*(2, Qt::Horizontal, tr("Практика"));

model2->*setHeaderData*(3, Qt::Horizontal, tr("Лабораторные"));

ui->tableView2->*setModel*(model2);

ui->tableView2->show();

Листинг 22. Заполнение формы данными.

Затем я реализовала вызова транзакции по нажатию кнопки и при условии выбора строки нагрузки (листинг 23):

query.exec(QString("CALL mydb.move('%1', %2, '%3', '%4','%5','%6','%7',%8);").arg(department)

.arg(model1->*data*(model1->*index*(idx, 6)).toInt())

.arg(fio)

.arg(model1->*data*(model1->*index*(idx, 0)).toString())

.arg(model1->*data*(model1->*index*(idx, 3)).toString())

.arg(model1->*data*(model1->*index*(idx, 2)).toString())

.arg(model1->*data*(model1->*index*(idx, 4)).toString())

.arg(model1->*data*(model1->*index*(idx, 5)).toString()));

Листинг 23. Реализация вызова транзакции.

Реализация обновления таблиц (листинг 24):

model1->*clear*();

model1->setQuery(QString("SELECT s.id, s.name,p.student\_group,p.d\_id,p.control,p.all\_hours,p.id FROM mydb.plan p JOIN mydb.subject s ON s.id = p.s\_id JOIN mydb.teacher t ON t.id = p.t\_id WHERE t.login='%1';").arg(log));

model1->*setHeaderData*(0, Qt::Horizontal, tr("Сокращение"));

model1->*setHeaderData*(1, Qt::Horizontal, tr("Предмет"));

model1->*setHeaderData*(2, Qt::Horizontal, tr("Группа"));

model1->*setHeaderData*(3, Qt::Horizontal, tr("Кафедра группы"));

model1->*setHeaderData*(4, Qt::Horizontal, tr("Форма контроля"));

model1->*setHeaderData*(5, Qt::Horizontal, tr("Часы"));

model1->*setHeaderData*(6, Qt::Horizontal, tr("План"));

ui->tableView1->*setModel*(model1);

ui->tableView1->setColumnHidden(0, true);

ui->tableView1->setColumnHidden(6, true);

ui->tableView1->show();

model2->*clear*();

model2->setQuery(QString("SELECT s.name,s.lec\_hours,s.prac\_hours,s.lab\_hours FROM mydb.plan p JOIN mydb.subject s ON s.id = p.s\_id JOIN mydb.teacher t ON t.id = p.t\_id WHERE t.login='%1' GROUP BY s.id;").arg(log));

model2->*setHeaderData*(0, Qt::Horizontal, tr("Предмет"));

model2->*setHeaderData*(1, Qt::Horizontal, tr("Лекции"));

model2->*setHeaderData*(2, Qt::Horizontal, tr("Практика"));

model2->*setHeaderData*(3, Qt::Horizontal, tr("Лабораторные"));

ui->tableView2->*setModel*(model2);

ui->tableView2->show();

ui->pushButton->setDisabled(true);

Листинг 24. Реализация обновления таблиц.

Разработка пользовательской формы на этом окончена. Далее идёт административная, то есть форма кафедры.

При запуске формы в ней отрисовываются все таблицы, при этом таблицы teacher, plan и subject отрисовываются не полностью, а в соответствии с кафедрой, под которой зашли в приложение. А также по полученным из запросов данным заполняется часть комбобоксов на форме (листинг 25).

ui->setupUi(this);

log=login;

model1->setQuery(QString("SELECT \* FROM mydb.department d;"));

model1->*setHeaderData*(0, Qt::Horizontal, tr("Сокращение"));

model1->*setHeaderData*(1, Qt::Horizontal, tr("Название"));

model1->*setHeaderData*(2, Qt::Horizontal, tr("Сколько преподавателей на кафедре"));

model1->*setHeaderData*(3, Qt::Horizontal, tr("Сколько групп на кафедре"));

model1->*setHeaderData*(4, Qt::Horizontal, tr("Заведующий кафедрой"));

model1->*setHeaderData*(5, Qt::Horizontal, tr("Логин"));

model1->*setHeaderData*(6, Qt::Horizontal, tr("Пароль"));

ui->tableView\_2->*setModel*(model1);

ui->tableView\_2->show();

for(int i=0;i<model1->*rowCount*();i++){

ui->comboBox\_15->addItem(model1->*data*(model1->*index*(i, 0)).toString());

}

for(int i=0;i<model1->*rowCount*();i++){

ui->comboBox\_11->addItem(model1->*data*(model1->*index*(i, 0)).toString());

}

model2->setQuery(QString("SELECT t.id,t.position,t.login,t.password,t.d\_id FROM mydb.teacher t JOIN mydb.department d ON t.d\_id = d.id WHERE d.login='%1';").arg(login));

model2->*setHeaderData*(0, Qt::Horizontal, tr("ФИО"));

model2->*setHeaderData*(1, Qt::Horizontal, tr("Должность"));

model2->*setHeaderData*(2, Qt::Horizontal, tr("Логин"));

model2->*setHeaderData*(3, Qt::Horizontal, tr("Пароль"));

model2->*setHeaderData*(4, Qt::Horizontal, tr("Кафедра"));

ui->tableView\_3->*setModel*(model2);

ui->tableView\_3->show();

dep = model2->*data*(model2->*index*(0, 4)).toString();

ui->comboBox\_t2->addItem(model2->*data*(model2->*index*(0, 4)).toString());

for(int i=0;i<model2->*rowCount*();i++){

ui->comboBox\_9->addItem(model2->*data*(model2->*index*(i, 0)).toString());

}

for(int i=0;i<model2->*rowCount*();i++){

ui->comboBox\_13->addItem(model2->*data*(model2->*index*(i, 0)).toString());

}

model3->setQuery(QString("SELECT p.id,p.t\_id,p.s\_id,p.d\_id,p.student\_group,p.control,p.all\_hours FROM mydb.plan p JOIN mydb.teacher t ON t.id = p.t\_id JOIN mydb.department d ON t.d\_id = d.id WHERE d.login='%1';").arg(login));

model3->*setHeaderData*(0, Qt::Horizontal, tr("План"));

model3->*setHeaderData*(1, Qt::Horizontal, tr("Преподаватель"));

model3->*setHeaderData*(2, Qt::Horizontal, tr("Дисциплина"));

model3->*setHeaderData*(3, Qt::Horizontal, tr("Кафедра группы"));

model3->*setHeaderData*(4, Qt::Horizontal, tr("Группа"));

model3->*setHeaderData*(5, Qt::Horizontal, tr("Форма контроля"));

model3->*setHeaderData*(6, Qt::Horizontal, tr("Часы"));

ui->tableView\_4->*setModel*(model3);

ui->tableView\_4->setColumnHidden(0, true);

ui->tableView\_4->show();

model4->setQuery(QString("SELECT s.id, s.name,s.lec\_hours,s.prac\_hours,s.lab\_hours FROM mydb.subject s JOIN mydb.plan p ON s.id = p.s\_id JOIN mydb.teacher t ON t.id = p.t\_id JOIN mydb.department d ON t.d\_id = d.id WHERE d.login='%1' GROUP BY s.id;").arg(login));

model4->*setHeaderData*(0, Qt::Horizontal, tr("Сокращение"));

model4->*setHeaderData*(1, Qt::Horizontal, tr("Дисциплина"));

model4->*setHeaderData*(2, Qt::Horizontal, tr("Лекции"));

model4->*setHeaderData*(3, Qt::Horizontal, tr("Практики"));

model4->*setHeaderData*(4, Qt::Horizontal, tr("Лабораторные"));

ui->tableView\_5->*setModel*(model4);

ui->tableView\_5->show();

model5->setQuery(QString("SELECT s.id FROM mydb.subject s;"));

for(int i=0;i<model5->*rowCount*();i++){

ui->comboBox\_14->addItem(model5->*data*(model5->*index*(i, 0)).toString());

}

for(int i=0;i<model5->*rowCount*();i++){

ui->comboBox\_10->addItem(model5->*data*(model5->*index*(i, 0)).toString());

}

Листинг 25. Реализация заполнения формы.

Если выбрать в какой-либо таблице строку, то данные из неё попадут в ячейки в разделе обновления (листинг 26).

void admin::**on\_tableView\_2\_clicked**(const QModelIndex &index)

{

ui->pushButton\_dep\_up->setDisabled(false);

ui->pushButton\_dep\_del->setDisabled(false);

idx1 = ui->tableView\_2->selectionModel()->currentIndex().row();

ui->comboBox\_dep1->removeItem(0);

ui->comboBox\_dep1->addItem(model1->*data*(model1->*index*(idx1, 0)).toString());

ui->lineEdit\_5->setText(model1->*data*(model1->*index*(idx1, 1)).toString());

ui->lineEdit\_6->setText(model1->*data*(model1->*index*(idx1, 5)).toString());

ui->lineEdit\_7->setText(model1->*data*(model1->*index*(idx1, 6)).toString());

}

void admin::**on\_tableView\_3\_clicked**(const QModelIndex &index)

{

ui->pushButton\_t\_up->setDisabled(false);

ui->pushButton\_t\_del->setDisabled(false);

idx2 = ui->tableView\_3->selectionModel()->currentIndex().row();

ui->comboBox\_t3->removeItem(0);

ui->comboBox\_t3->addItem(model2->*data*(model2->*index*(idx2, 0)).toString());

ui->comboBox\_t4->setCurrentIndex(ui->comboBox\_t4->findText(model2->*data*(model2->*index*(idx2, 1)).toString()));

ui->comboBox\_t5->setCurrentIndex(ui->comboBox\_t5->findText(model2->*data*(model2->*index*(idx2, 4)).toString()));

ui->lineEdit\_11->setText(model2->*data*(model2->*index*(idx2, 2)).toString());

ui->lineEdit\_12->setText(model2->*data*(model2->*index*(idx2, 3)).toString());

}

void admin::**on\_tableView\_4\_clicked**(const QModelIndex &index)

{

ui->pushButton\_p\_up->setDisabled(false);

ui->pushButton\_p\_del->setDisabled(false);

idx3 = ui->tableView\_4->selectionModel()->currentIndex().row();

ui->comboBox\_13->setCurrentIndex(ui->comboBox\_13->findText(model3->*data*(model3->*index*(idx3, 1)).toString()));

ui->comboBox\_14->setCurrentIndex(ui->comboBox\_14->findText(model3->*data*(model3->*index*(idx3, 2)).toString()));

ui->comboBox\_15->setCurrentIndex(ui->comboBox\_15->findText(model3->*data*(model3->*index*(idx3, 3)).toString()));

ui->lineEdit\_14->setText(model3->*data*(model3->*index*(idx3, 4)).toString());

}

void admin::**on\_tableView\_5\_clicked**(const QModelIndex &index)

{

ui->pushButton\_s\_up->setDisabled(false);

ui->pushButton\_s\_del->setDisabled(false);

idx4 = ui->tableView\_5->selectionModel()->currentIndex().row();

ui->comboBox\_17->removeItem(0);

ui->comboBox\_17->addItem(model4->*data*(model4->*index*(idx4, 0)).toString());

ui->lineEdit\_20->setText(model4->*data*(model4->*index*(idx4, 1)).toString());

ui->lineEdit\_21->setText(model4->*data*(model4->*index*(idx4, 2)).toString());

ui->lineEdit\_22->setText(model4->*data*(model4->*index*(idx4, 3)).toString());

ui->lineEdit\_23->setText(model4->*data*(model4->*index*(idx4, 4)).toString());

}

Листинг 26. Реализация заполнения ячеек в разделе обновления.

Далее я реализовала методы добавления, удаления и обновления данных в таблицах по нажатию соответствующих кнопок, а также обновление самих таблиц (пример для таблицы department представлен в листинге 27).

void admin::**on\_pushButton\_dep\_add\_clicked**()

{

query.exec(QString("CALL mydb.insert\_department('%1','%2','%3','%4');").arg(ui->lineEdit->text())

.arg(ui->lineEdit\_2->text())

.arg(ui->lineEdit\_3->text())

.arg(ui->lineEdit\_4->text()));

}

void admin::**on\_pushButton\_dep\_up\_t\_clicked**()

{

model1->*clear*();

model1->setQuery(QString("SELECT \* FROM mydb.department d;"));

model1->*setHeaderData*(0, Qt::Horizontal, tr("Сокращение"));

model1->*setHeaderData*(1, Qt::Horizontal, tr("Название"));

model1->*setHeaderData*(2, Qt::Horizontal, tr("Сколько преподавателей на кафедре"));

model1->*setHeaderData*(3, Qt::Horizontal, tr("Сколько групп на кафедре"));

model1->*setHeaderData*(4, Qt::Horizontal, tr("Заведующий кафедрой"));

model1->*setHeaderData*(5, Qt::Horizontal, tr("Логин"));

model1->*setHeaderData*(6, Qt::Horizontal, tr("Пароль"));

ui->tableView\_2->*setModel*(model1);

ui->tableView\_2->show();

ui->pushButton\_dep\_up->setDisabled(true);

ui->pushButton\_dep\_del->setDisabled(true);

}

void admin::**on\_pushButton\_dep\_del\_clicked**()

{

query.exec(QString("CALL mydb.delete\_department('%1');").arg(model1->*data*(model1->*index*(idx1, 0)).toString()));

}

void admin::**on\_pushButton\_dep\_up\_clicked**()

{

query.exec(QString("CALL mydb.update\_department('%1','%2','%3','%4');").arg(ui->comboBox\_dep1->currentText())

.arg(ui->lineEdit\_5->text())

.arg(ui->lineEdit\_6->text())

.arg(ui->lineEdit\_7->text()));

}

Листинг 27. Пример добавления, удаления, обновления данных и обновления таблицы.

Реализация отображения результатов запросов (листинг 28):

void admin::**on\_comboBox\_currentIndexChanged**(const QString &arg1)

{

if (ui->comboBox->currentIndex()==0)

{

model5->*clear*();

ui->tableView->*setModel*(model5);

ui->textBrowser->setText("");

}

if (ui->comboBox->currentIndex()==1)

{

model5->*clear*();

model5->setQuery(QString("SELECT s.name, p.all\_hours,CASE WHEN s.lec\_hours=0 THEN 'Нет занятий' ELSE CAST(s.lec\_hours AS varchar(3)) END lec\_hours, CASE WHEN s.prac\_hours=0 THEN 'Нет занятий' ELSE CAST(s.prac\_hours AS varchar(3)) END prac\_hours, CASE WHEN s.lab\_hours=0 THEN 'Нет занятий' ELSE CAST(s.lab\_hours AS varchar(3)) END lab\_hours FROM mydb.subject s JOIN mydb.plan p ON s.id = p.s\_id GROUP BY s.name, p.all\_hours,s.lec\_hours,s.prac\_hours,s.lab\_hours;"));

model5->*setHeaderData*(0, Qt::Horizontal, tr("Дисциплина"));

model5->*setHeaderData*(1, Qt::Horizontal, tr("Часы"));

model5->*setHeaderData*(2, Qt::Horizontal, tr("Лекции"));

model5->*setHeaderData*(3, Qt::Horizontal, tr("Практики"));

model5->*setHeaderData*(4, Qt::Horizontal, tr("Лабораторные"));

ui->tableView->*setModel*(model5);

ui->tableView->show();

ui->textBrowser->setText("Выводит дисциплины и сколько часов по ним, если часов 0, то выводит надпись, что нет занятий такого типа");

}

if (ui->comboBox->currentIndex()==2)

{

model5->*clear*();

model5->setQuery(QString("SELECT \* FROM mydb.work\_view;"));

model5->*setHeaderData*(0, Qt::Horizontal, tr("ФИО"));

model5->*setHeaderData*(1, Qt::Horizontal, tr("Должность"));

model5->*setHeaderData*(2, Qt::Horizontal, tr(""));

model5->*setHeaderData*(3, Qt::Horizontal, tr("Дисциплина"));

model5->*setHeaderData*(4, Qt::Horizontal, tr(""));

model5->*setHeaderData*(5, Qt::Horizontal, tr("Группа"));

model5->*setHeaderData*(6, Qt::Horizontal, tr("Форма контроля"));

model5->*setHeaderData*(7, Qt::Horizontal, tr("Часы"));

ui->tableView->*setModel*(model5);

ui->tableView->setColumnHidden(2, true);

ui->tableView->setColumnHidden(4, true);

ui->tableView->show();

ui->textBrowser->setText("Многотабличное представление с критерием упорядочетельности");

}

if (ui->comboBox->currentIndex()==3)

{

model5->*clear*();

model5->setQuery(QString("Select ROUND(AVG(p.all\_hours),1), (Select ROUND(AVG(s.lec\_hours),1) FROM mydb.subject s), (Select ROUND(AVG(s.prac\_hours),1) FROM mydb.subject s), (Select ROUND(AVG(s.lab\_hours),1) FROM mydb.subject s) FROM mydb.plan p;"));

model5->*setHeaderData*(0, Qt::Horizontal, tr("Среднее общее"));

model5->*setHeaderData*(1, Qt::Horizontal, tr("Среднее лекций"));

model5->*setHeaderData*(2, Qt::Horizontal, tr("Среднее практик"));

model5->*setHeaderData*(3, Qt::Horizontal, tr("Среднее лабораторных"));

ui->tableView->*setModel*(model5);

ui->tableView->show();

ui->textBrowser->setText("Среднее количество часов");

}

if (ui->comboBox->currentIndex()==4)

{

model5->*clear*();

model5->setQuery(QString("SELECT d.name,t.id,t.position,p.s\_id,p.student\_group FROM (SELECT id, position, d\_id FROM mydb.teacher WHERE position='Заведующий кафедрой') AS t JOIN mydb.plan p ON t.id=p.t\_id JOIN mydb.department d ON d.id=t.d\_id WHERE d.id='КБ-1';"));

model5->*setHeaderData*(0, Qt::Horizontal, tr("Кафедра"));

model5->*setHeaderData*(1, Qt::Horizontal, tr("ФИО"));

model5->*setHeaderData*(2, Qt::Horizontal, tr("Должность"));

model5->*setHeaderData*(3, Qt::Horizontal, tr("Дисциплина"));

model5->*setHeaderData*(4, Qt::Horizontal, tr("Группа"));

ui->tableView->*setModel*(model5);

ui->tableView->show();

ui->textBrowser->setText("Нагрузка заведующего кафедрой КБ-1");

}

if (ui->comboBox->currentIndex()==5)

{

model5->*clear*();

model5->setQuery(QString("Select s.name, p.all\_hours FROM mydb.subject s JOIN mydb.plan p ON s.id=p.s\_id WHERE p.all\_hours<(select AVG(p.all\_hours) FROM mydb.plan p) GROUP BY s.name,p.all\_hours;"));

model5->*setHeaderData*(0, Qt::Horizontal, tr("Название"));

model5->*setHeaderData*(1, Qt::Horizontal, tr("Часы"));

ui->tableView->*setModel*(model5);

ui->tableView->show();

ui->textBrowser->setText("Дисциплины, у которых общее кол-во часов меньше среднего");

}

if (ui->comboBox->currentIndex()==6)

{

model5->*clear*();

model5->setQuery(QString("SELECT t.id, t.d\_id, (SELECT COUNT(\*) FROM mydb.plan p WHERE t.id=p.t\_id) AS sum\_plan FROM mydb.teacher t ORDER BY sum\_plan DESC;"));

model5->*setHeaderData*(0, Qt::Horizontal, tr("ФИО"));

model5->*setHeaderData*(1, Qt::Horizontal, tr("Кафедра"));

model5->*setHeaderData*(0, Qt::Horizontal, tr("Кол-во нагрузки"));

ui->tableView->*setModel*(model5);

ui->tableView->show();

ui->textBrowser->setText("Сколько нагрузки у каждого преподавателя");

}

if (ui->comboBox->currentIndex()==7)

{

model5->*clear*();

model5->setQuery(QString("Select t.d\_id, SUM(pp.c) FROM mydb.teacher t, LATERAL(select COUNT(\*) AS c FROM mydb.plan p WHERE p.t\_id=t.id) pp GROUP BY t.d\_id;"));

model5->*setHeaderData*(0, Qt::Horizontal, tr("Кафедра"));

model5->*setHeaderData*(1, Qt::Horizontal, tr("Кол-во нагрузки"));

ui->tableView->*setModel*(model5);

ui->tableView->show();

ui->textBrowser->setText("Сколько нагрузки у всех преподавателей с кафедры вместе");

}

if (ui->comboBox->currentIndex()==8)

{

model5->*clear*();

model5->setQuery(QString("Select t.d\_id, t.id FROM mydb.teacher t WHERE (select COUNT(\*) FROM mydb.plan p WHERE p.t\_id=t.id)=0 ORDER BY t.d\_id;"));

model5->*setHeaderData*(0, Qt::Horizontal, tr("Кафедра"));

model5->*setHeaderData*(1, Qt::Horizontal, tr("ФИО"));

ui->tableView->*setModel*(model5);

ui->tableView->show();

ui->textBrowser->setText("Кто на кафедре не имеет нагрузки");

}

if (ui->comboBox->currentIndex()==9)

{

model5->*clear*();

model5->setQuery(QString("SELECT t.id AS name, COUNT(\*) FROM mydb.plan p JOIN mydb.teacher t ON p.t\_id=t.id GROUP BY t.id HAVING COUNT(\*)<10;"));

model5->*setHeaderData*(0, Qt::Horizontal, tr("ФИО"));

model5->*setHeaderData*(1, Qt::Horizontal, tr("Кол-во нагрузки"));

ui->tableView->*setModel*(model5);

ui->tableView->show();

ui->textBrowser->setText("Преподаватели, у которых количество нагрузки меньше 10");

}

if (ui->comboBox->currentIndex()==10)

{

model5->*clear*();

model5->setQuery(QString("SELECT t.id AS name FROM mydb.teacher t WHERE 'РПОЗОС'=any (SELECT p.s\_id FROM mydb.plan p WHERE t.id=p.t\_id AND p.d\_id='КБ-2');"));

model5->*setHeaderData*(0, Qt::Horizontal, tr("ФИО"));

ui->tableView->*setModel*(model5);

ui->tableView->show();

ui->textBrowser->setText("Преподаватели, которые ведут дисциплину РПОЗОС студентам с кафедры КБ-2");

}

if (ui->comboBox->currentIndex()==11)

{

model5->*clear*();

model5->setQuery(QString("SELECT \* FROM mydb.show\_group('%1');").arg(dep));

model5->*setHeaderData*(0, Qt::Horizontal, tr("Группа"));

ui->tableView->*setModel*(model5);

ui->tableView->show();

ui->textBrowser->setText("Группы, которым ведут занятия преподаватели с кафедры "+dep);

}

}

Листинг 28. Реализация отображения результатов запросов.

Реализация отображения, отсортированного по критериям, VIEW (листинг 29):

void admin::**on\_comboBox\_2\_currentIndexChanged**(const QString &arg1)

{

if (ui->comboBox\_2->currentIndex()==0)

{

model5->*clear*();

model5->setQuery(QString("SELECT \* FROM mydb.work\_view;"));

model5->*setHeaderData*(0, Qt::Horizontal, tr("ФИО"));

model5->*setHeaderData*(1, Qt::Horizontal, tr("Должность"));

model5->*setHeaderData*(2, Qt::Horizontal, tr(""));

model5->*setHeaderData*(3, Qt::Horizontal, tr("Дисциплина"));

model5->*setHeaderData*(4, Qt::Horizontal, tr(""));

model5->*setHeaderData*(5, Qt::Horizontal, tr("Группа"));

model5->*setHeaderData*(6, Qt::Horizontal, tr("Форма контроля"));

model5->*setHeaderData*(7, Qt::Horizontal, tr("Часы"));

ui->tableView->*setModel*(model5);

ui->tableView->setColumnHidden(2, true);

ui->tableView->setColumnHidden(4, true);

ui->tableView->show();

}

if (ui->comboBox\_2->currentIndex()==1)

{

model5->*clear*();

model5->setQuery(QString("SELECT \* FROM mydb.work\_view ORDER BY fio;"));

model5->*setHeaderData*(0, Qt::Horizontal, tr("ФИО"));

model5->*setHeaderData*(1, Qt::Horizontal, tr("Должность"));

model5->*setHeaderData*(2, Qt::Horizontal, tr(""));

model5->*setHeaderData*(3, Qt::Horizontal, tr("Дисциплина"));

model5->*setHeaderData*(4, Qt::Horizontal, tr(""));

model5->*setHeaderData*(5, Qt::Horizontal, tr("Группа"));

model5->*setHeaderData*(6, Qt::Horizontal, tr("Форма контроля"));

model5->*setHeaderData*(7, Qt::Horizontal, tr("Часы"));

ui->tableView->*setModel*(model5);

ui->tableView->setColumnHidden(2, true);

ui->tableView->setColumnHidden(4, true);

ui->tableView->show();

}

if (ui->comboBox\_2->currentIndex()==2)

{

model5->*clear*();

model5->setQuery(QString("SELECT \* FROM mydb.work\_view ORDER BY position;"));

model5->*setHeaderData*(0, Qt::Horizontal, tr("ФИО"));

model5->*setHeaderData*(1, Qt::Horizontal, tr("Должность"));

model5->*setHeaderData*(2, Qt::Horizontal, tr(""));

model5->*setHeaderData*(3, Qt::Horizontal, tr("Дисциплина"));

model5->*setHeaderData*(4, Qt::Horizontal, tr(""));

model5->*setHeaderData*(5, Qt::Horizontal, tr("Группа"));

model5->*setHeaderData*(6, Qt::Horizontal, tr("Форма контроля"));

model5->*setHeaderData*(7, Qt::Horizontal, tr("Часы"));

ui->tableView->*setModel*(model5);

ui->tableView->setColumnHidden(2, true);

ui->tableView->setColumnHidden(4, true);

ui->tableView->show();

}

if (ui->comboBox\_2->currentIndex()==3)

{

model5->*clear*();

model5->setQuery(QString("SELECT \* FROM mydb.work\_view ORDER BY subject;"));

model5->*setHeaderData*(0, Qt::Horizontal, tr("ФИО"));

model5->*setHeaderData*(1, Qt::Horizontal, tr("Должность"));

model5->*setHeaderData*(2, Qt::Horizontal, tr(""));

model5->*setHeaderData*(3, Qt::Horizontal, tr("Дисциплина"));

model5->*setHeaderData*(4, Qt::Horizontal, tr(""));

model5->*setHeaderData*(5, Qt::Horizontal, tr("Группа"));

model5->*setHeaderData*(6, Qt::Horizontal, tr("Форма контроля"));

model5->*setHeaderData*(7, Qt::Horizontal, tr("Часы"));

ui->tableView->*setModel*(model5);

ui->tableView->setColumnHidden(2, true);

ui->tableView->setColumnHidden(4, true);

ui->tableView->show();

}

if (ui->comboBox\_2->currentIndex()==4)

{

model5->*clear*();

model5->setQuery(QString("SELECT \* FROM mydb.work\_view ORDER BY student\_group;"));

model5->*setHeaderData*(0, Qt::Horizontal, tr("ФИО"));

model5->*setHeaderData*(1, Qt::Horizontal, tr("Должность"));

model5->*setHeaderData*(2, Qt::Horizontal, tr(""));

model5->*setHeaderData*(3, Qt::Horizontal, tr("Дисциплина"));

model5->*setHeaderData*(4, Qt::Horizontal, tr(""));

model5->*setHeaderData*(5, Qt::Horizontal, tr("Группа"));

model5->*setHeaderData*(6, Qt::Horizontal, tr("Форма контроля"));

model5->*setHeaderData*(7, Qt::Horizontal, tr("Часы"));

ui->tableView->*setModel*(model5);

ui->tableView->setColumnHidden(2, true);

ui->tableView->setColumnHidden(4, true);

ui->tableView->show();

}

if (ui->comboBox\_2->currentIndex()==5)

{

model5->*clear*();

model5->setQuery(QString("SELECT \* FROM mydb.work\_view ORDER BY control;"));

model5->*setHeaderData*(0, Qt::Horizontal, tr("ФИО"));

model5->*setHeaderData*(1, Qt::Horizontal, tr("Должность"));

model5->*setHeaderData*(2, Qt::Horizontal, tr(""));

model5->*setHeaderData*(3, Qt::Horizontal, tr("Дисциплина"));

model5->*setHeaderData*(4, Qt::Horizontal, tr(""));

model5->*setHeaderData*(5, Qt::Horizontal, tr("Группа"));

model5->*setHeaderData*(6, Qt::Horizontal, tr("Форма контроля"));

model5->*setHeaderData*(7, Qt::Horizontal, tr("Часы"));

ui->tableView->*setModel*(model5);

ui->tableView->setColumnHidden(2, true);

ui->tableView->setColumnHidden(4, true);

ui->tableView->show();

}

if (ui->comboBox\_2->currentIndex()==6)

{

model5->*clear*();

model5->setQuery(QString("SELECT \* FROM mydb.work\_view ORDER BY all\_hours;"));

model5->*setHeaderData*(0, Qt::Horizontal, tr("ФИО"));

model5->*setHeaderData*(1, Qt::Horizontal, tr("Должность"));

model5->*setHeaderData*(2, Qt::Horizontal, tr(""));

model5->*setHeaderData*(3, Qt::Horizontal, tr("Дисциплина"));

model5->*setHeaderData*(4, Qt::Horizontal, tr(""));

model5->*setHeaderData*(5, Qt::Horizontal, tr("Группа"));

model5->*setHeaderData*(6, Qt::Horizontal, tr("Форма контроля"));

model5->*setHeaderData*(7, Qt::Horizontal, tr("Часы"));

ui->tableView->*setModel*(model5);

ui->tableView->setColumnHidden(2, true);

ui->tableView->setColumnHidden(4, true);

ui->tableView->show();

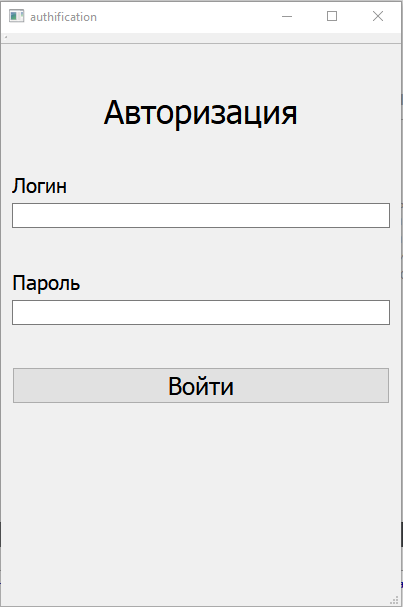
}

}

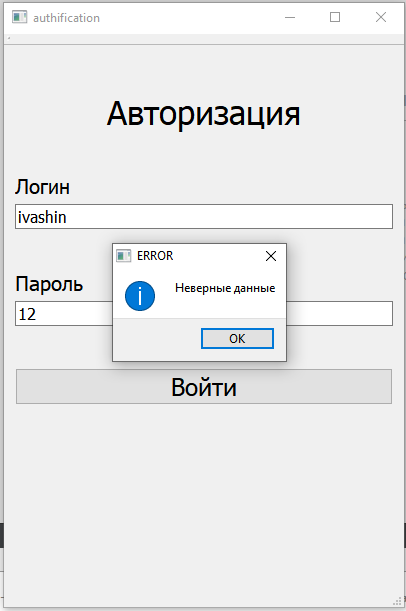
Листинг 29. Реализация отображения, отсортированного по критериям, VIEW.

# Глава 3. Контрольный пример

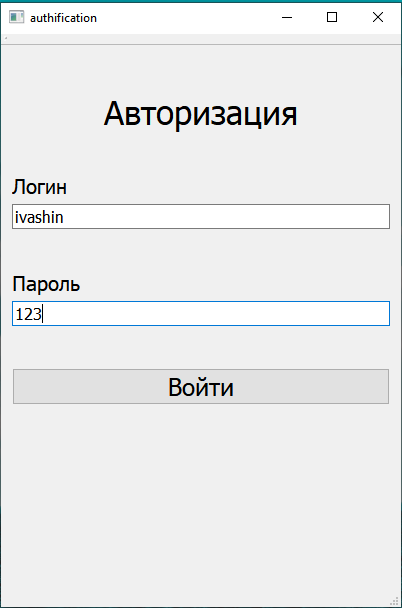
При запуске открывается окно авторизации.



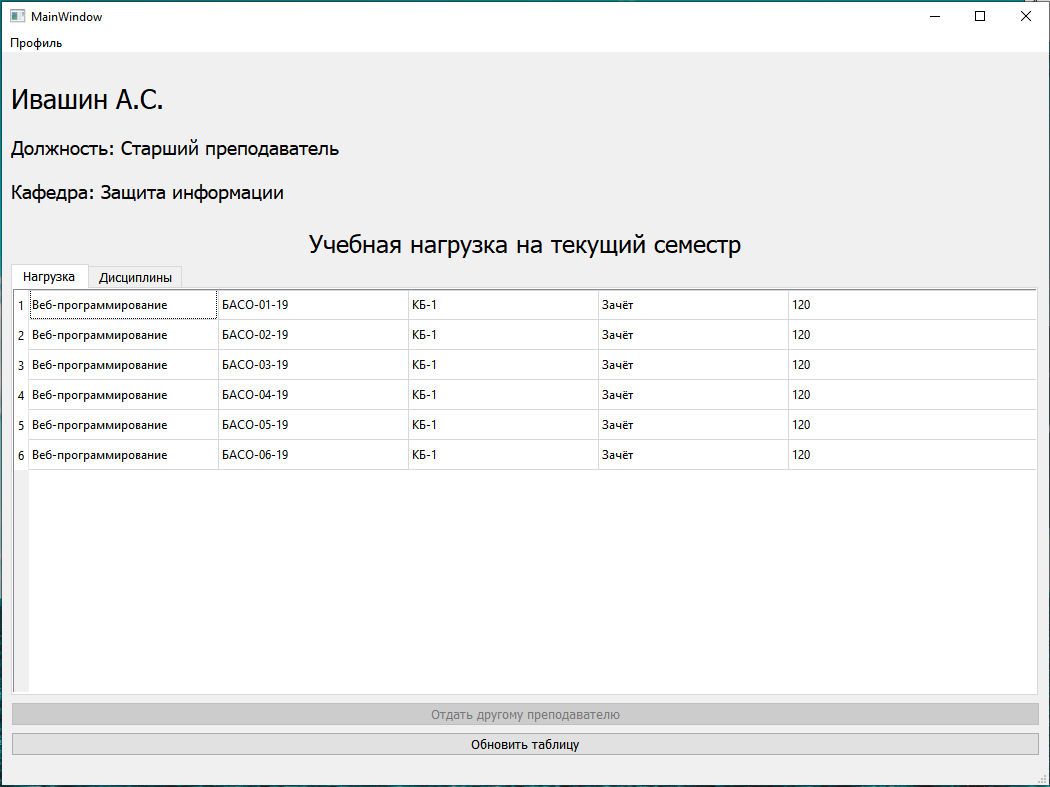
Если логин и пароль не нашлись в БД, то выводится ошибка.



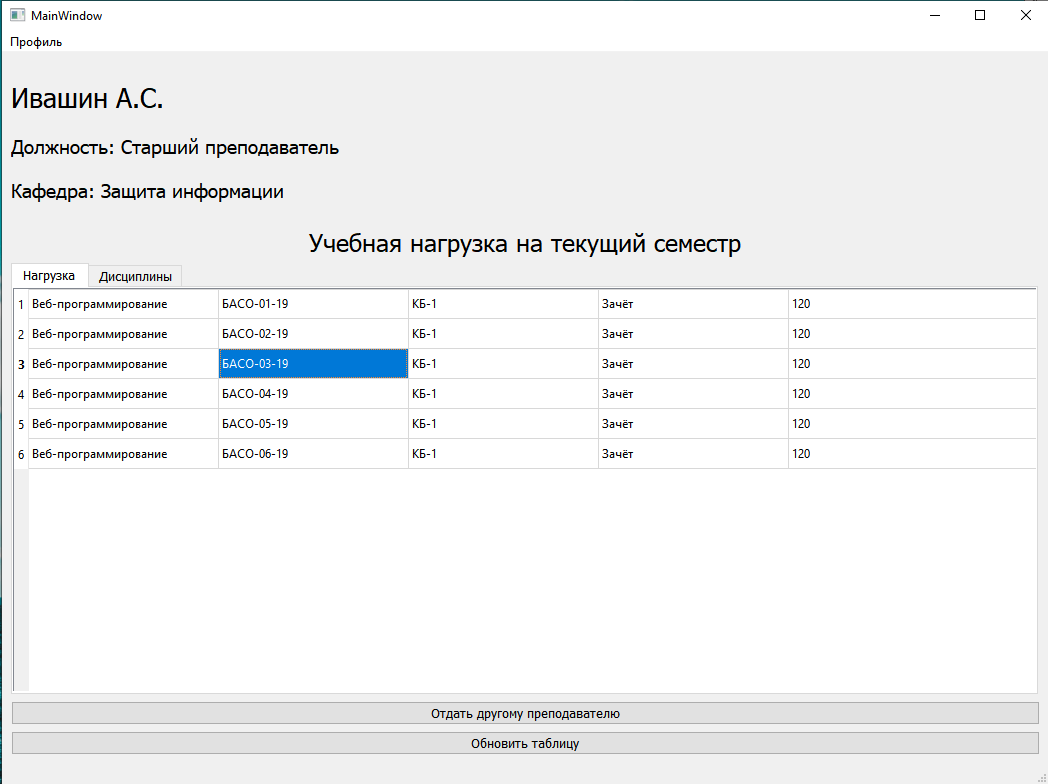
Введём логин и пароль преподавателя Ивашина А. С.



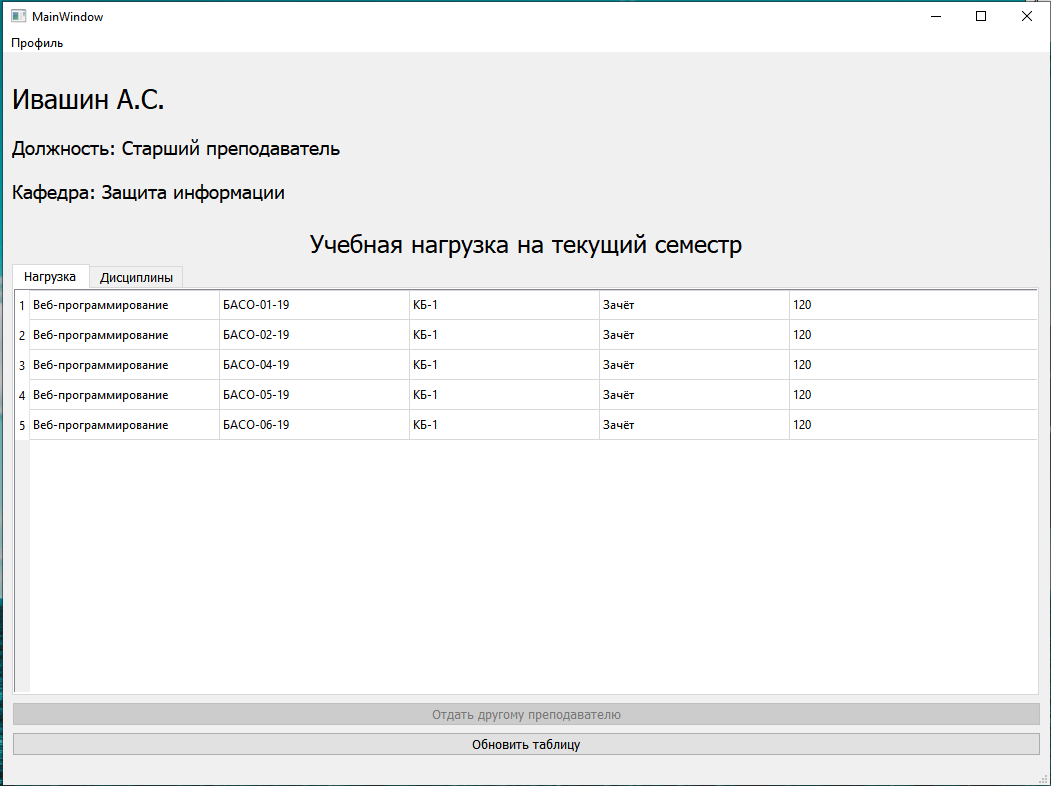
Пароль и логин верный, поэтому открывается пользовательская форма.



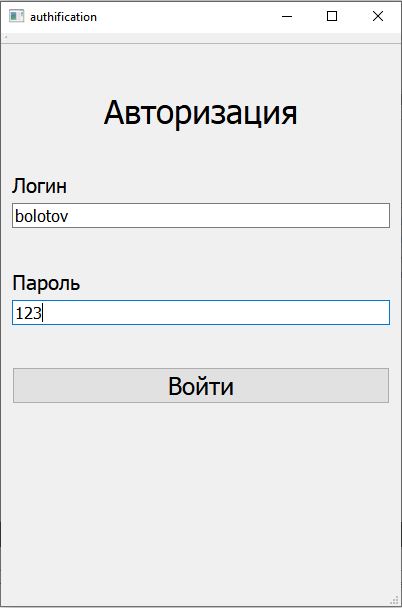
Выберем третью строку нагрузки для демонстрации работы транзакции.



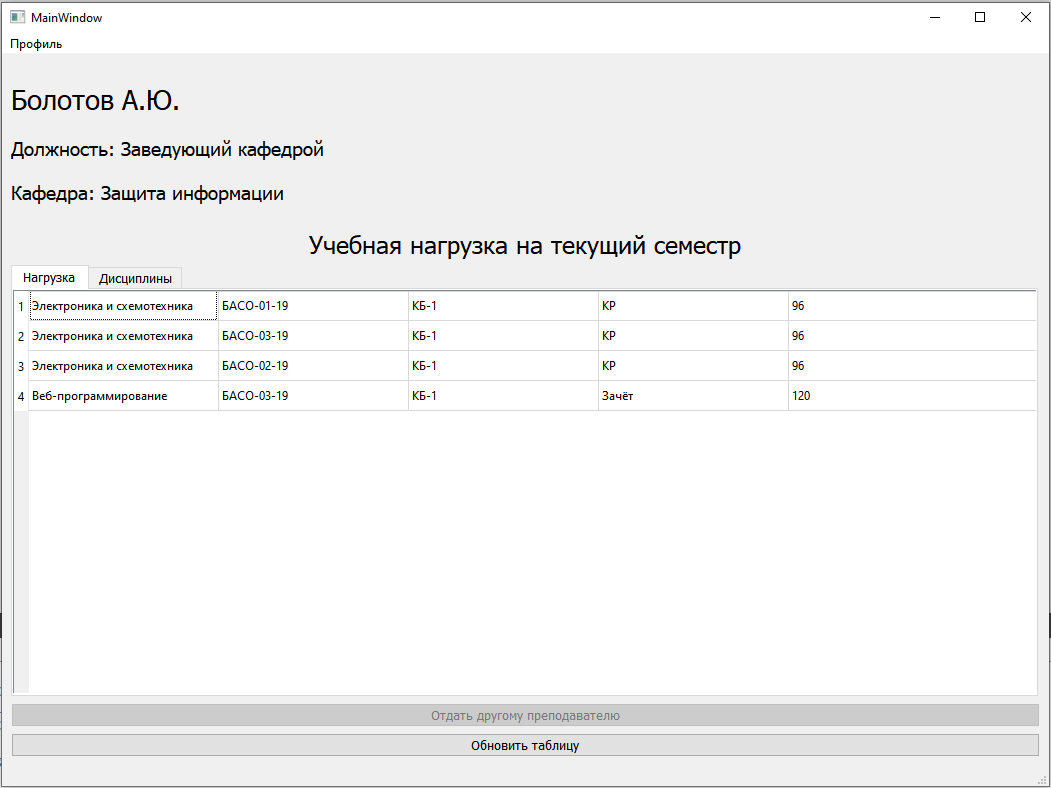
После нажатия на кнопку, вызывающую транзакцию, и нажатия на кнопку обновления таблицы, видим, что выбранная строка нагрузки исчезла.



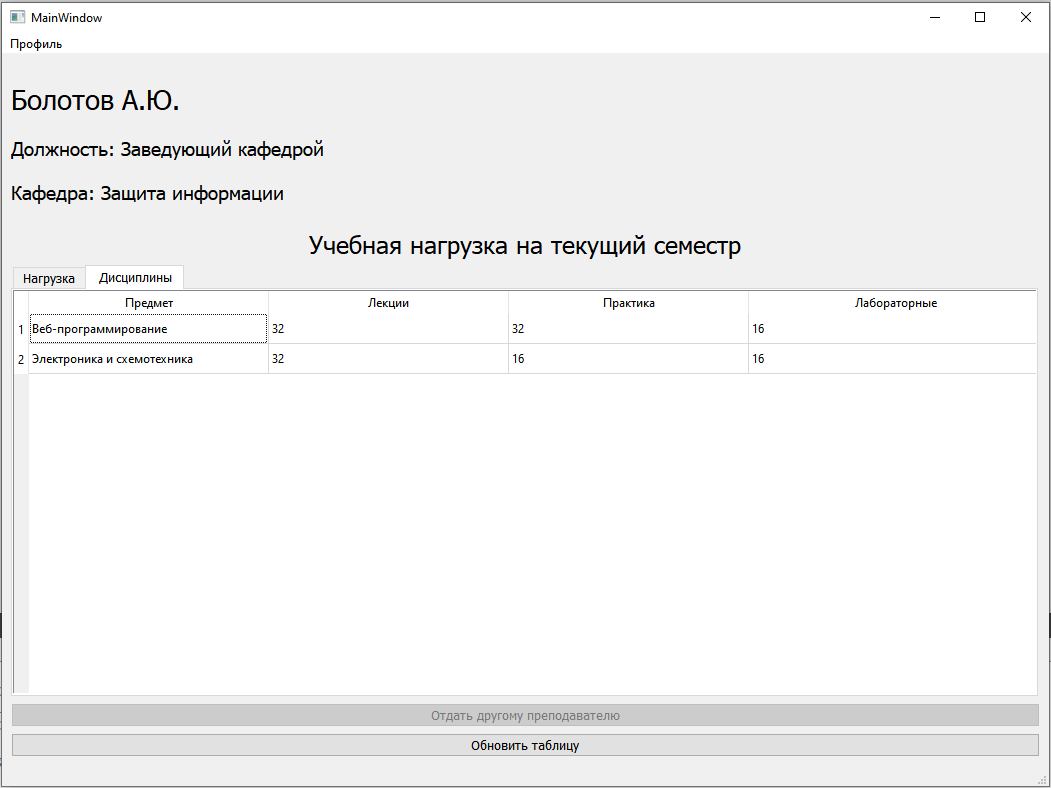
Ивашин А.С. преподаёт на кафедре КБ-1, поэтому строка нагрузки передалась первому преподавателю с кафедры, у которого нагрузки меньше 10. Этот преподаватель – Болотов А.Ю. Зайдём под его логином и паролем.



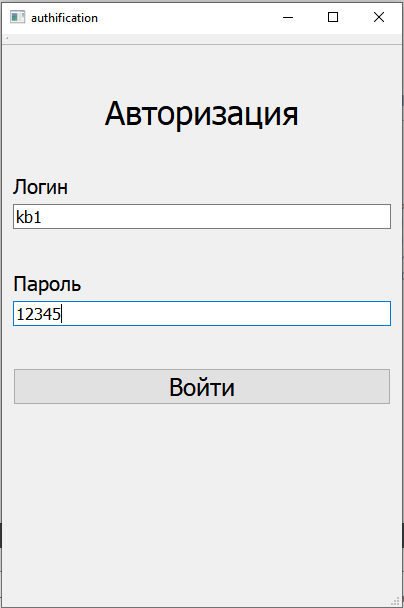
Мы видим, что строка нагрузки Ивашина А.С. передалась.



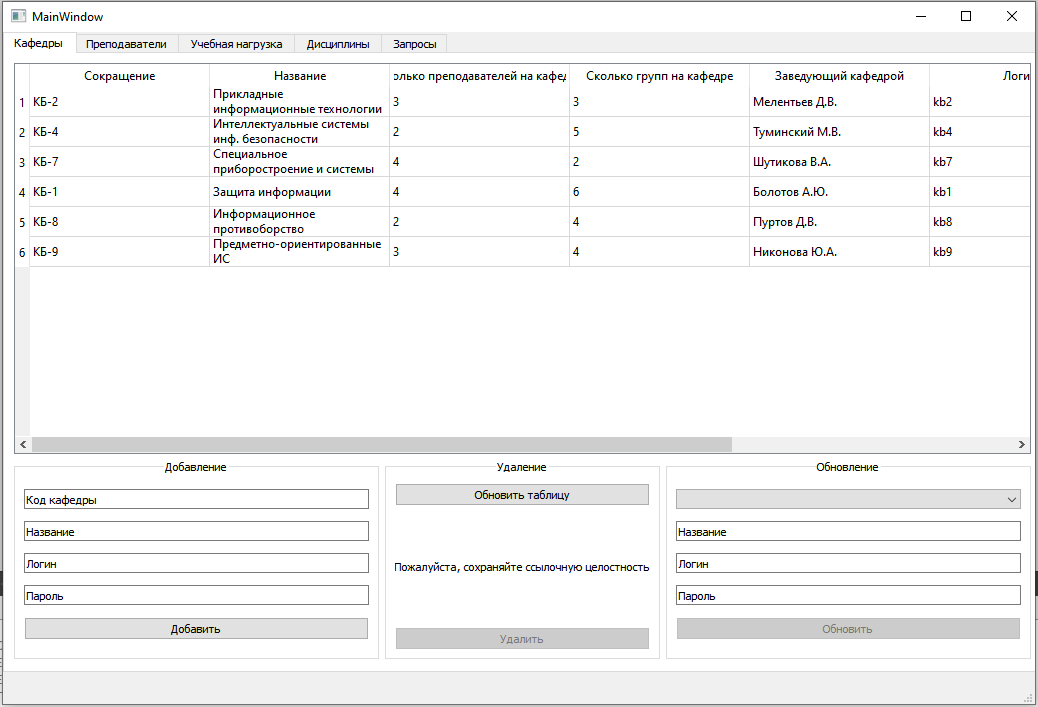
До транзакции Болотов А.Ю. ввёл только «Электронику и схемотехнику», но после в его дисциплинах появилось «Веб-программирование».



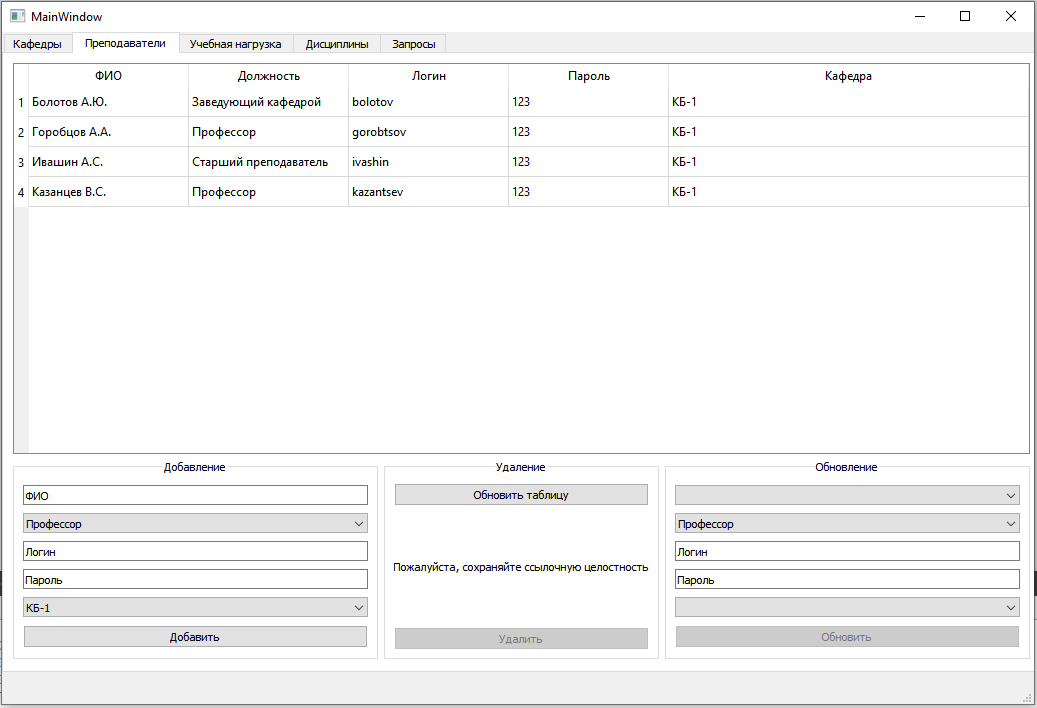
Теперь зайдём под административной ролью, например, под логином и паролем кафедры КБ-1.



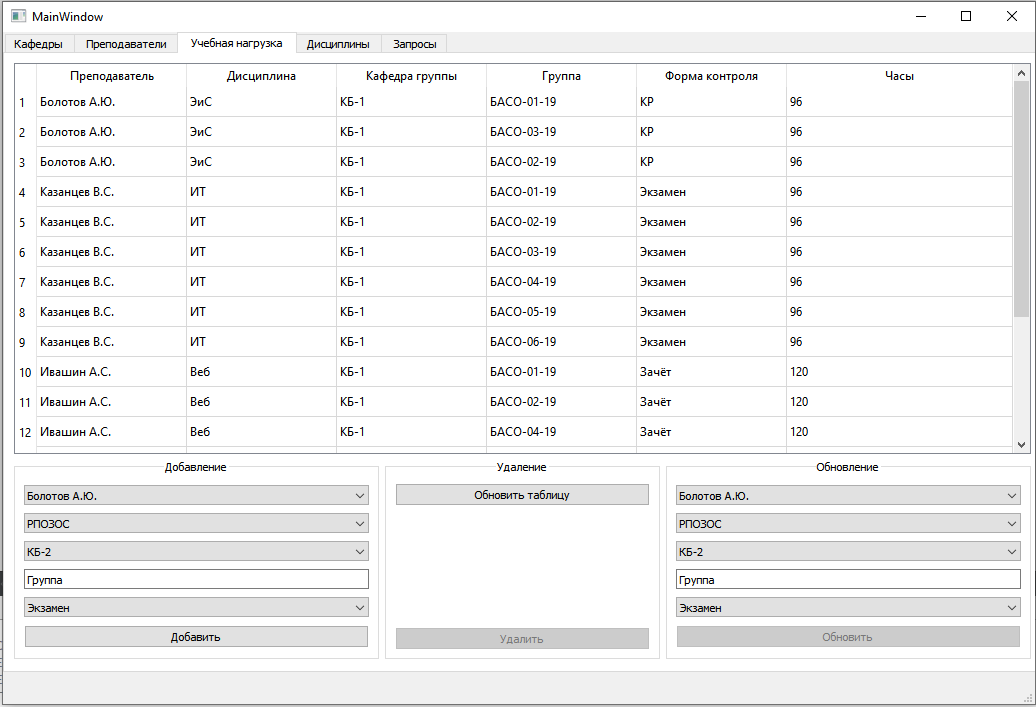
На вкладке «Кафедры» представлена информация о всех кафедрах.



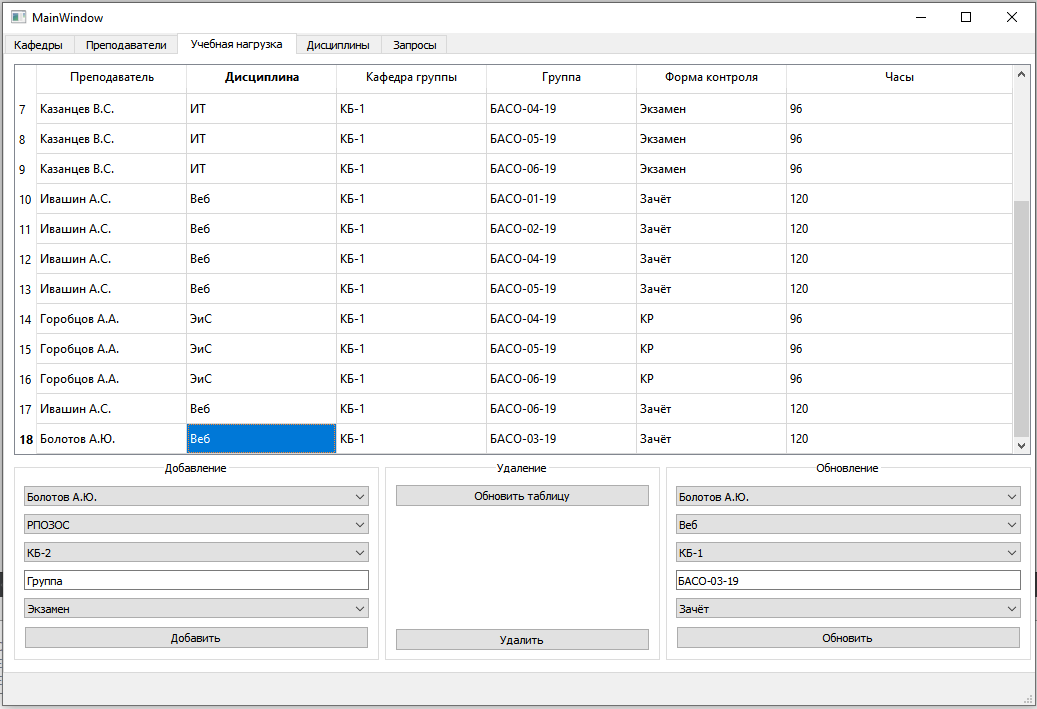
На вкладке «Преподаватели» представлена информация только о преподавателях с кафедры, под которой зашли.



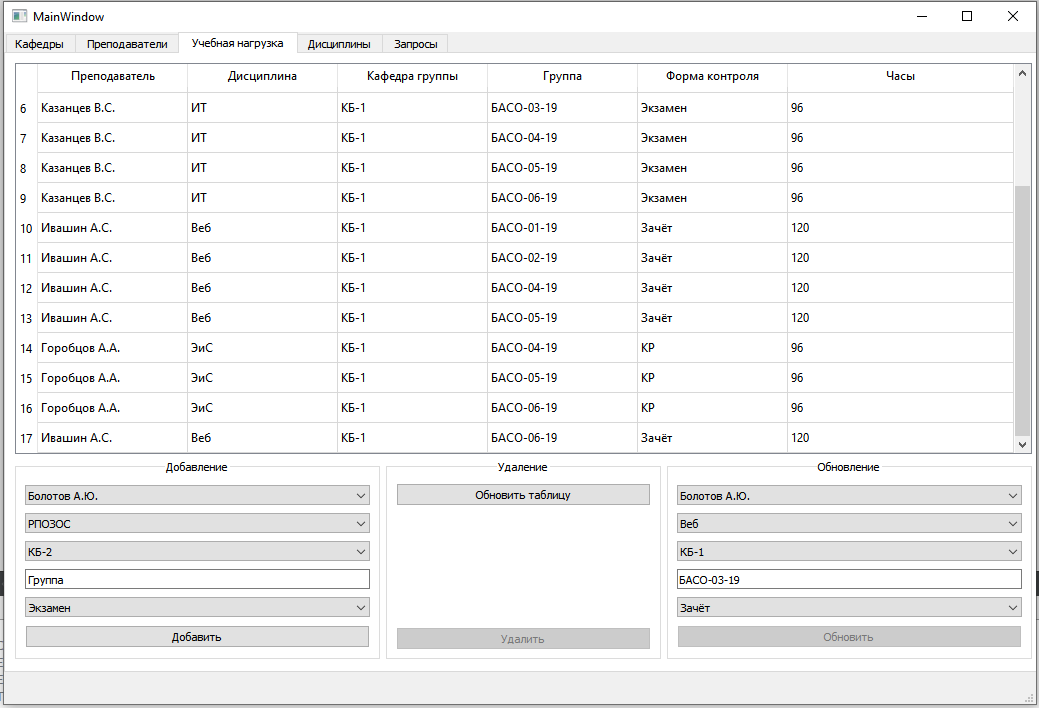
Аналогично на вкладке «Учебная нагрузка» представлена информация только о нагрузке преподавателей с кафедры, под которой зашли.



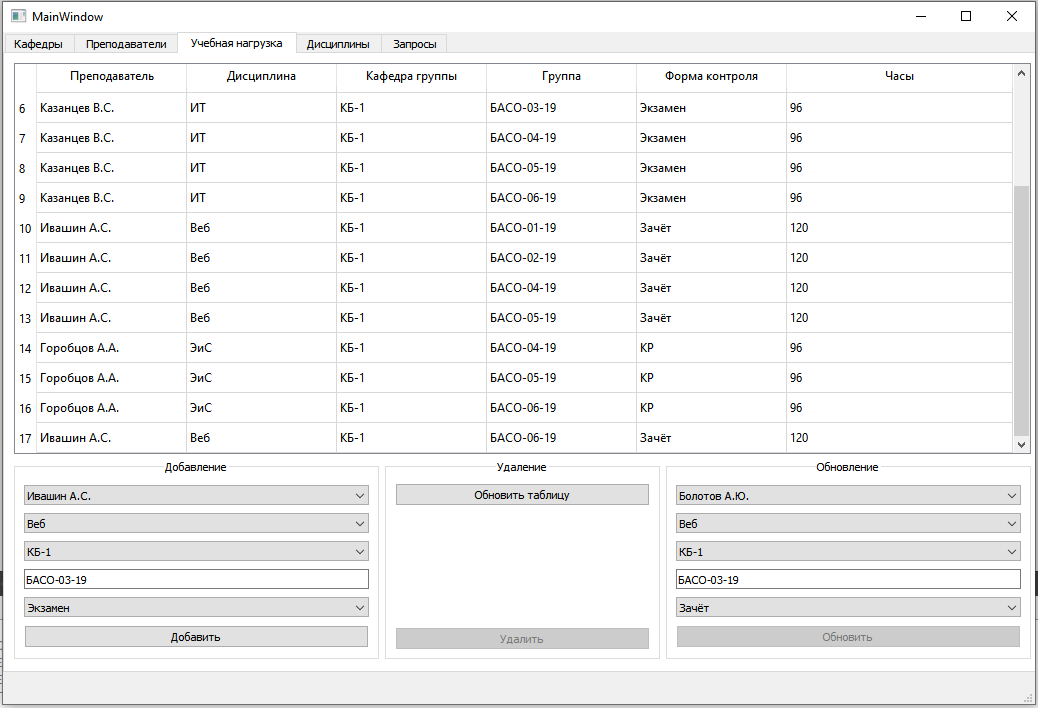
Для демонстрации функций добавления, удаления и обновления вернём Ивашину А.С. обратно строку нагрузки. Для этого выберем эту строку и удалим её.



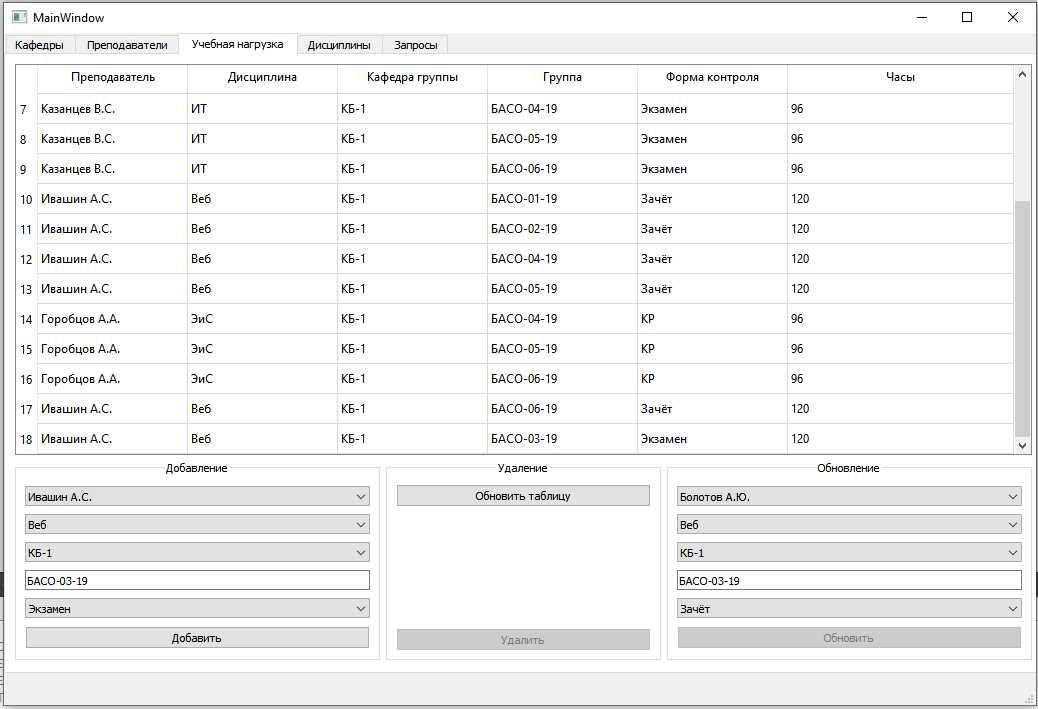
После обновления таблицы видим, что строка исчезла.



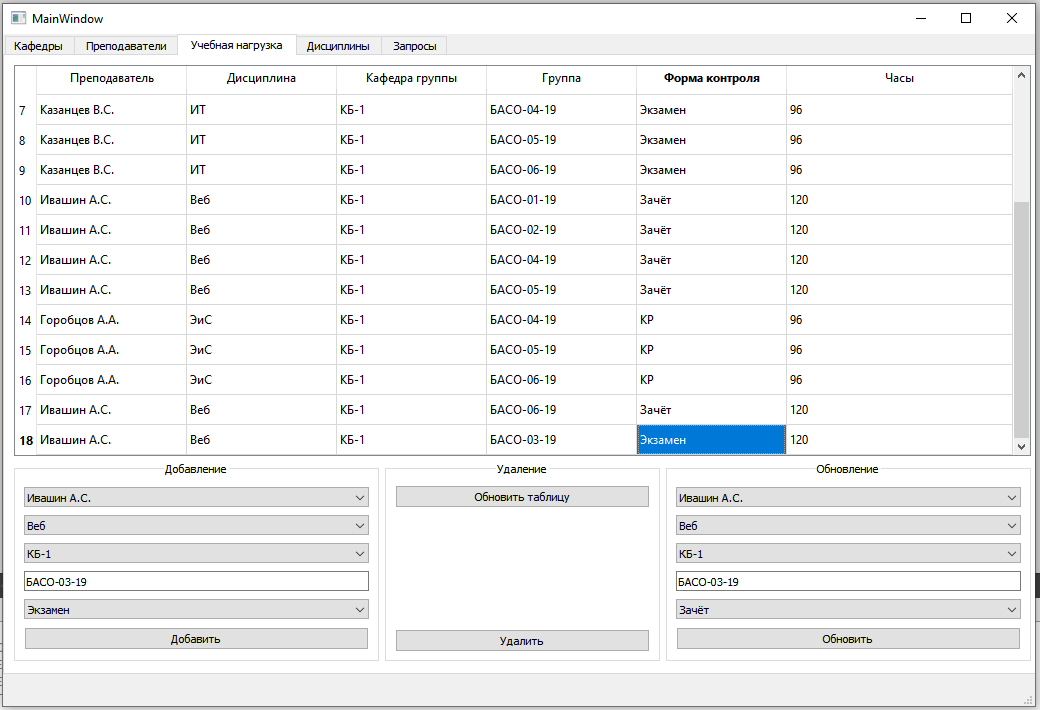
Забьём в форму добавления необходимые данные, но при этом изменим форму контроля на экзамен. Затем добавим строку.



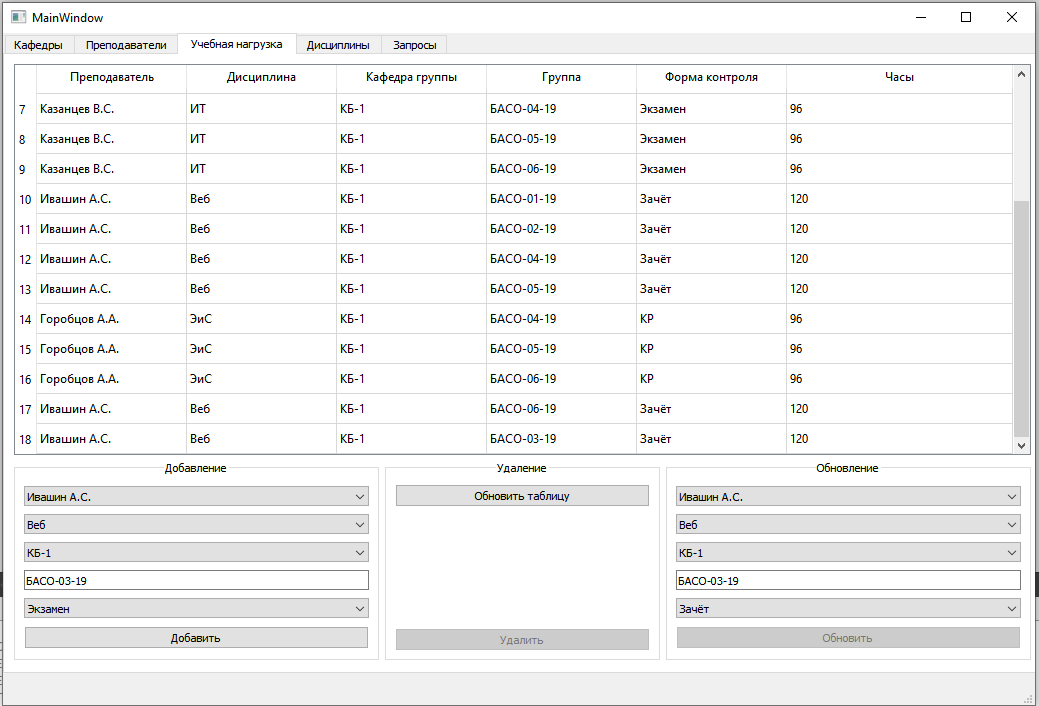
Строка успешно добавлена.



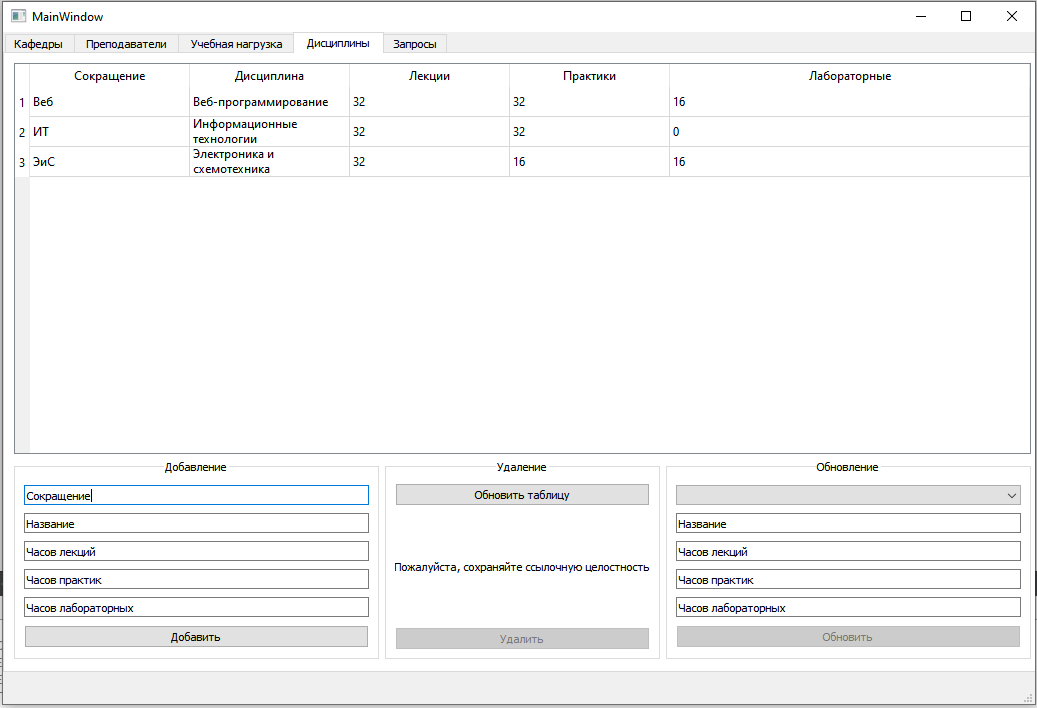
Осталось только обновить форму контроля в этой строке. Для этого выбираем строку и в форму обновления вносим необходимые данные. И обновляем.



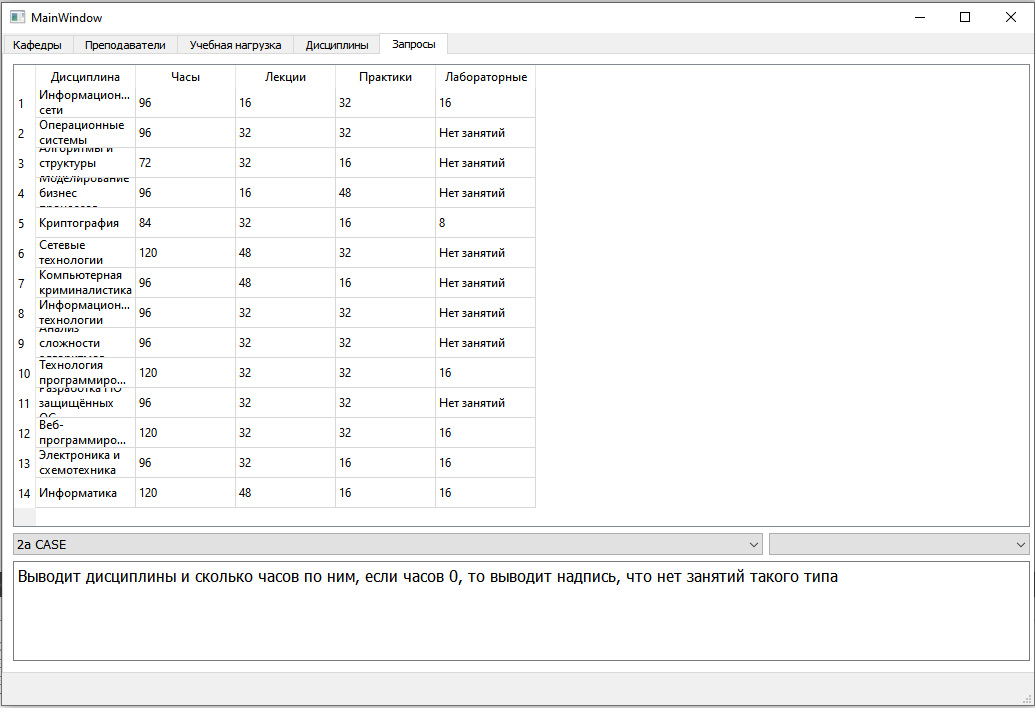
Форма контроля в строке изменилась на необходимую.



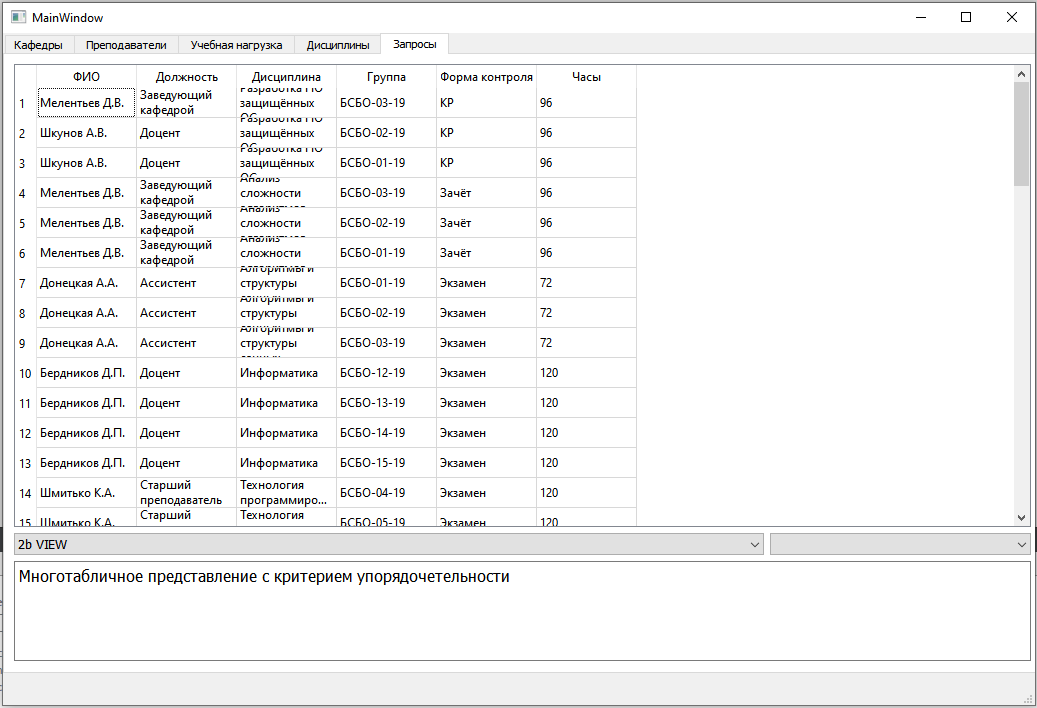
Дальше переходим на вкладку «Дисциплины», в которой представлена информация только о дисциплинах, которые ведут преподаватели с кафедры, под которой зашли.



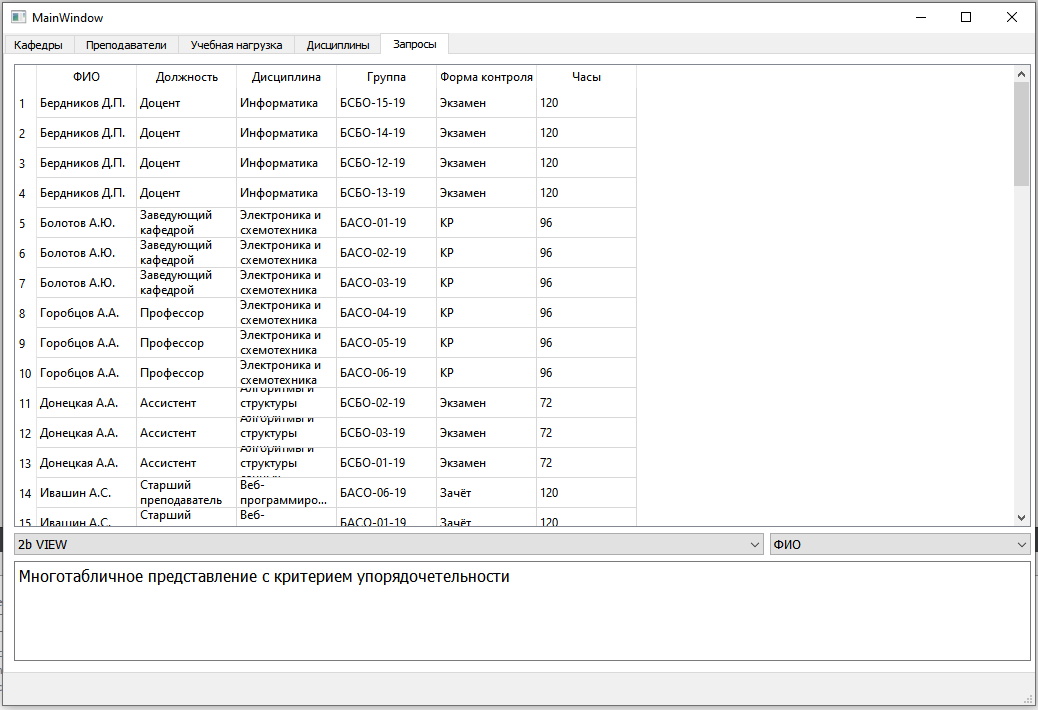
Для просмотра запросов перейдём на вкладку «Запросы». И в комбобоксе выберем запрос 2а CASE.



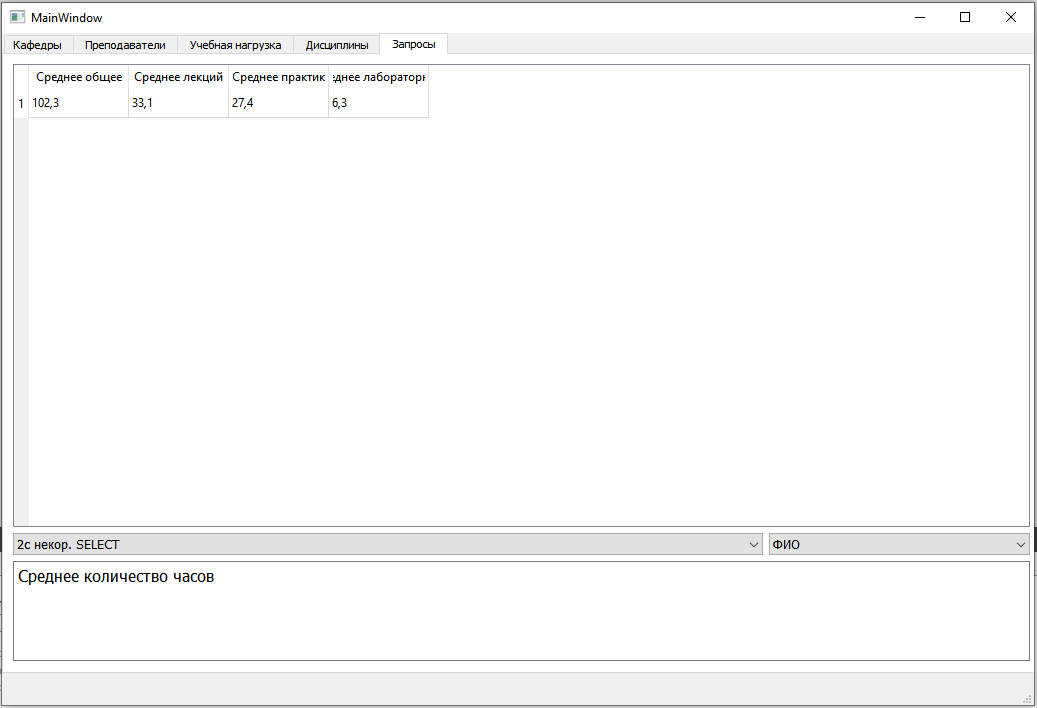
Теперь выберем запрос 2b VIEW.

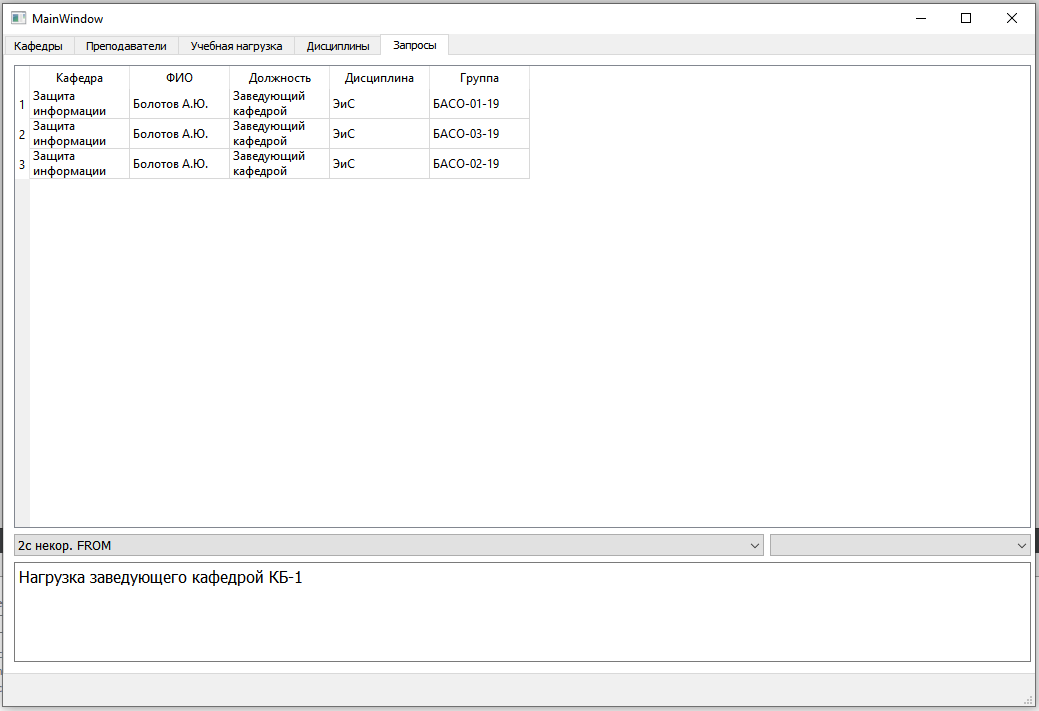


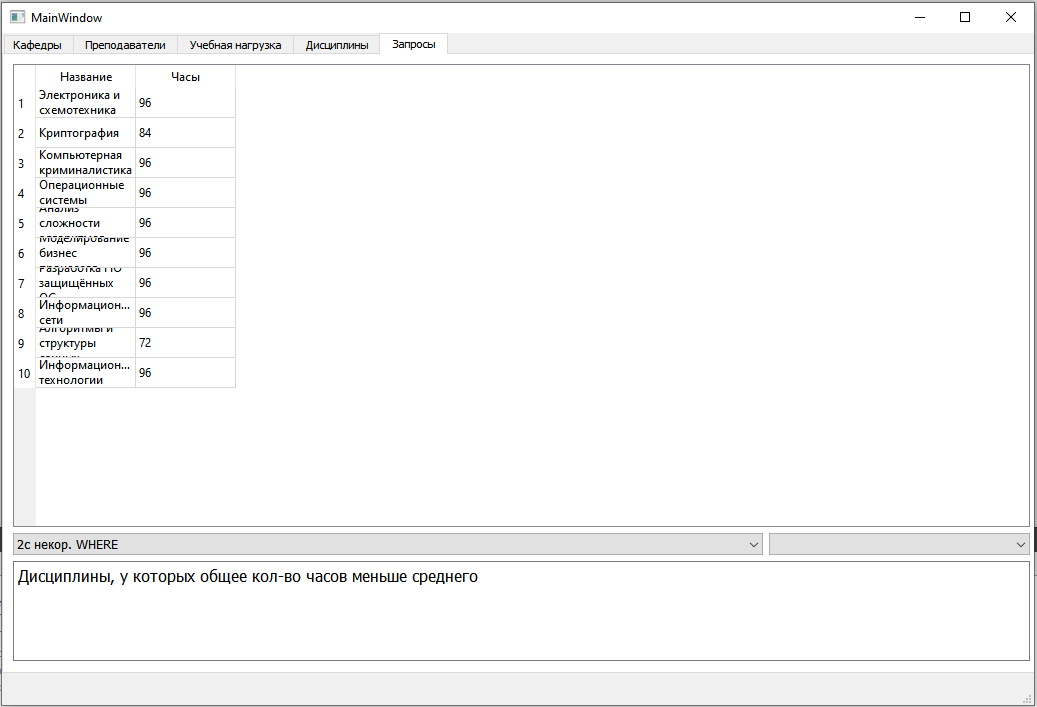
Результат запроса вывелся неупорядоченным. Для того чтобы упорядочить результат этого запроса, необходимо в соседнем комбобоксе выбрать критерий упорядоченности. Выберем «ФИО» и видим, что результат упорядочен по этому критерию. Аналогично с оставшимися критериями.

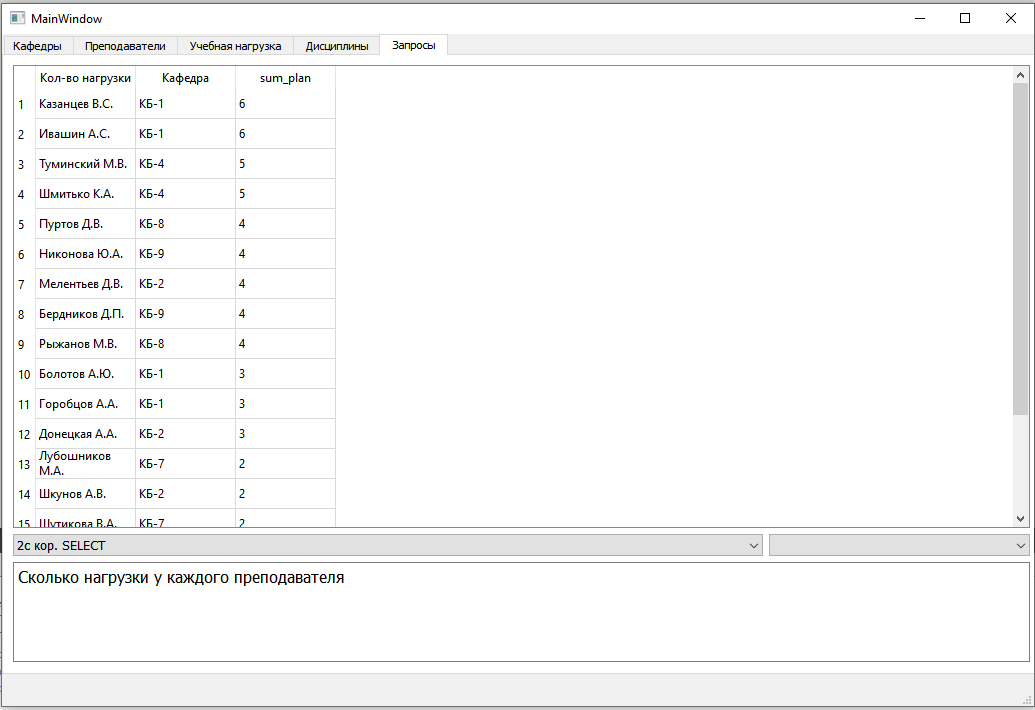


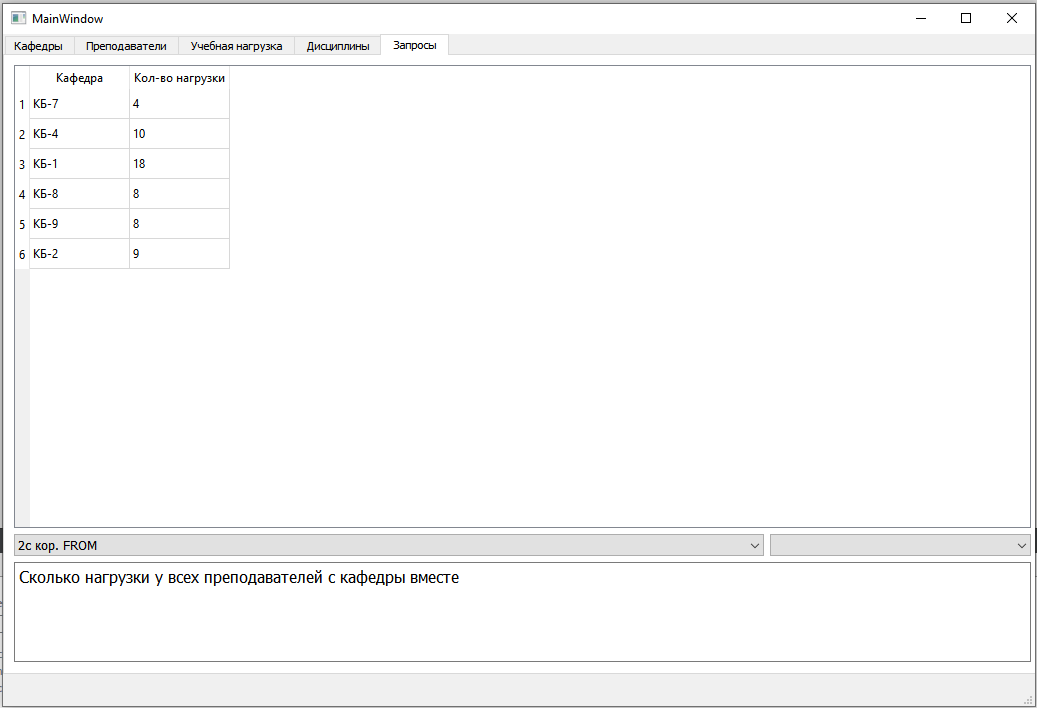
Посмотрим результаты оставшихся запросов.

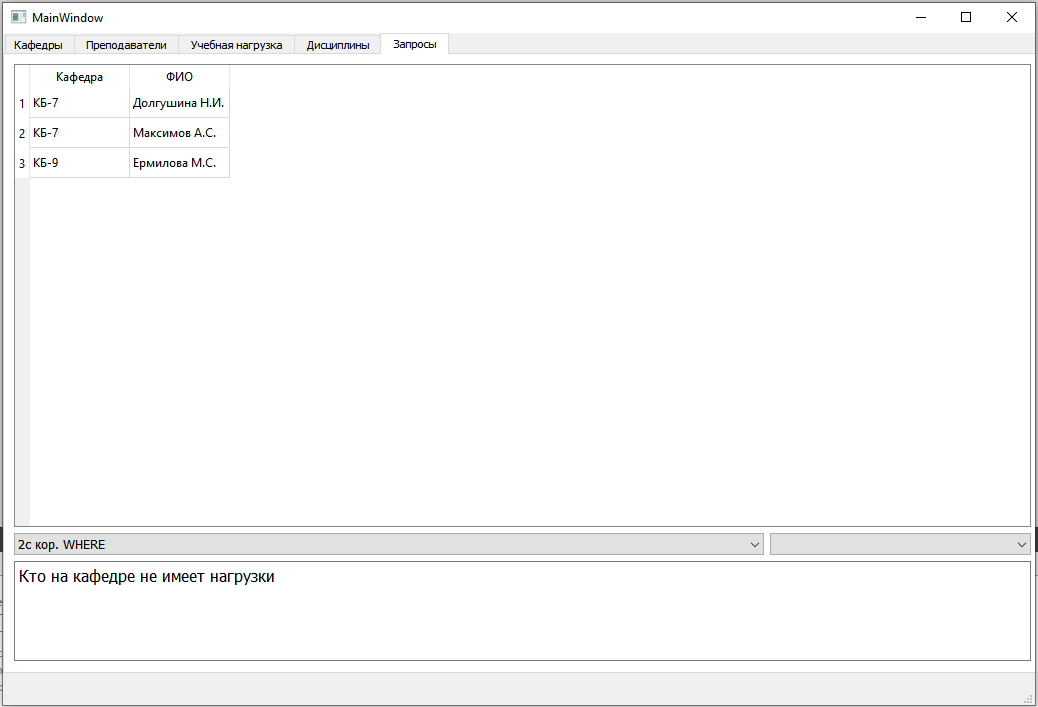


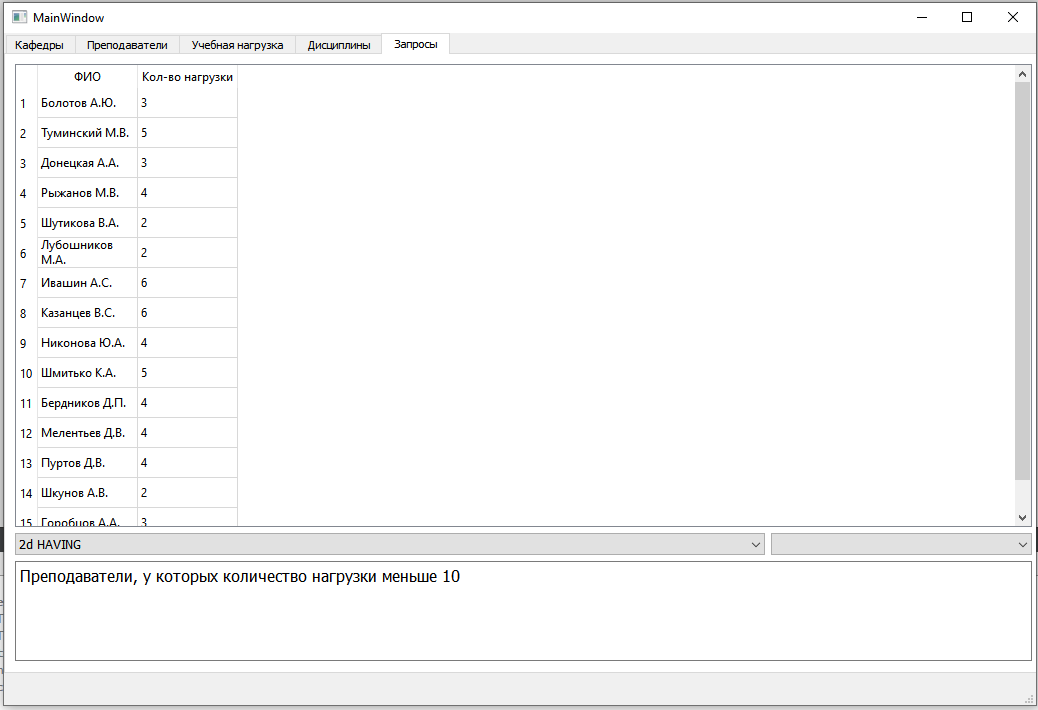


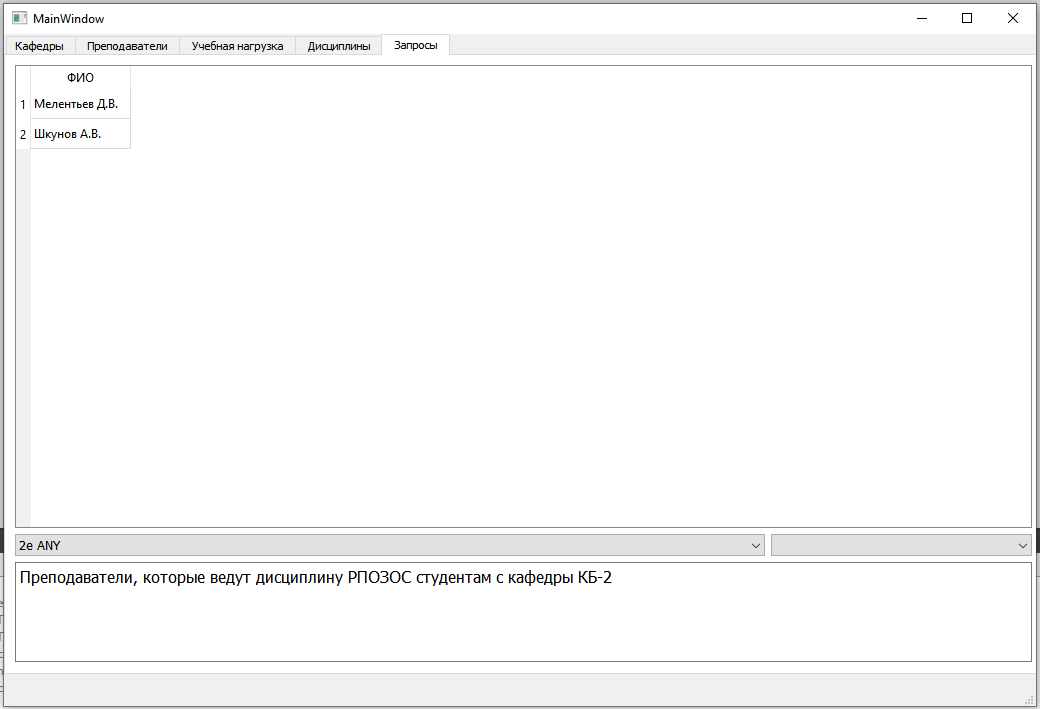


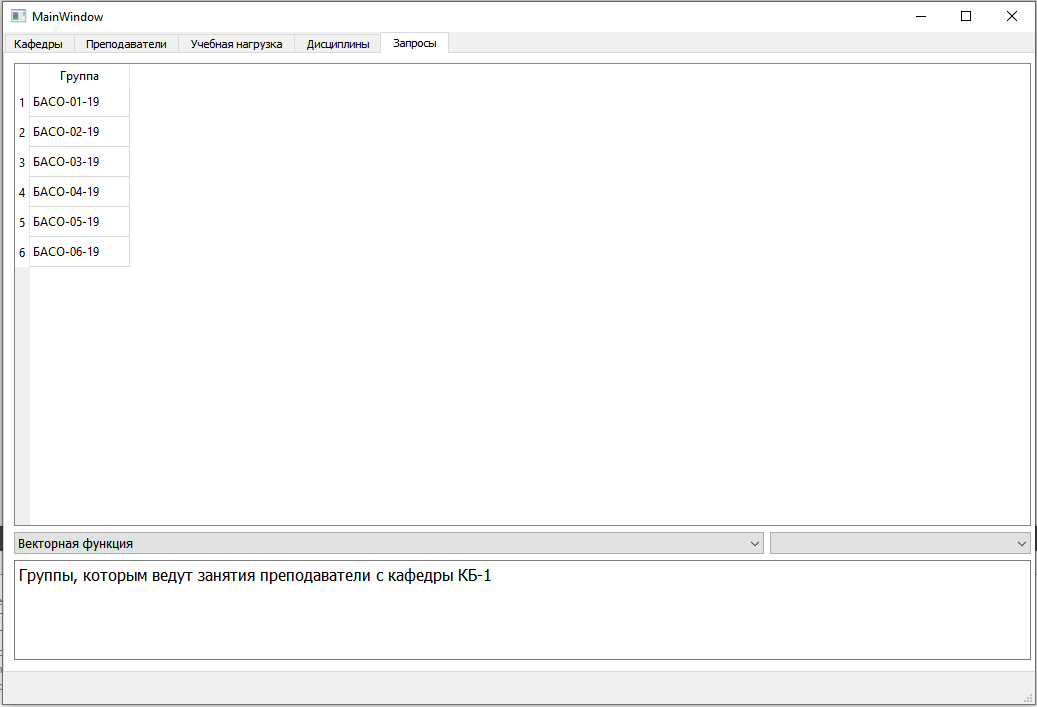












# Заключение

В ходе работы мной была изучена предметная область, а также я поработала с СУБД PostgreSQL и средой разработки Qt Creator. В соответствии с этим в работе были решены следующие задачи:

1. Изучена информация о предметной области;
2. Реализована серверная часть согласно требованиям;
3. Реализована клиентская часть;
4. Разработан контрольный пример, демонстрирующий работоспособность информационной базы и приложения.

Данный продукт можно использовать организации учебной деятельности преподавателей вуза.

# Список использованной литературы

1. Документация к PostgreSQL [Электронный ресурс] – Режим доступа: <https://postgrespro.ru/docs/postgresql/9.6/index> (Дата обращения 10.05.2021).
2. Хранение информации. Базы и хранилища данных [Электронный ресурс] – Режим доступа: <https://intuit.ru/studies/courses/13860/1257/lecture/24002?page=2> (Дата обращения 16.05.2021).
3. Что такое PostgreSQL? [Электронный ресурс] – Режим доступа: <https://coderlessons.com/tutorials/bazy-dannykh/uchebnik-postgresql/1-chto-takoe-postgresql> (Дата обращения 16.05.2021).
4. Qt Creator [Электронный ресурс] – Режим доступа: <https://ru.wikipedia.org/wiki/Qt_Creator> (Дата обращения 19.05.2021).
5. Qt Documentation [Электронный ресурс] – Режим доступа: <https://doc.qt.io/qt-5/> (Дата обращения 19.05.2021).