МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ

ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ им. А. И. ГЕРЦЕНА»



Основная профессиональная образовательная программа

Направление подготовки 09.03.01 Информатика и вычислительная техника

Направленность (профиль) «Технологии разработки программного обеспечения»

форма обучения – очная

**ОТЧЁТ**

по реализации проекта для дисциплины «Базы данных»

по направлению “09.03.01 – Информатика и вычислительная техника”

(профиль: “Технологии разработки программного обеспечения ”)

Преподаватель: к.ф-м.н., доцент кафедры ИТиЭО

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Жуков Н. Н.)

Преподаватель: ассистент кафедры ИТиЭО

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Иванова Е. А.)

Студенты 2 курса:

Елкина Г.А.\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Логинова С.А.\_\_\_\_\_\_\_\_\_\_\_\_

Санкт-Петербург

2020

# Оглавление

[**Оглавление**](#_cw61j4fzzkcw) **1**

[Ответственные](#_gjdgxs) **2**

[Предметная область](#_30j0zll) **2**

[Ход выполнения нормализации](#_1fob9te) **3**

[ER – диаграмма](#_2et92p0) **4**

[Объяснение выбранной СУБД](#_3znysh7) **4**

[Исходный текст запросов](#_tyjcwt) **5**

[По созданию таблиц](#_3dy6vkm) 5

[Для манипуляции над БД](#_3rdcrjn) 7

# Ответственные

Данный проект выполняли две студентки 2 (второго) курса направления подготовки «Информатика и вычислительная техника»: Елкина Галина Александровна и Логинова Софья Андреевна.

В совместном ведение участников группы были разработаны отношения базы данных, выстроена логика взаимодействия внутри системы. Принятие всех решений, а также общие вопросы реализации обсуждались вместе.

В обязанности Елкиной Г.А. входил процесс разработки программной части на языке Python версии 3.7, а также написание запросов для работы с базой данных с использованием импортируемого модуля SQLite3. При выполнении данной части задания были применены следующие знания по языку программирования Python: создание классов, функций; ветвление; импорт файлов; отлавливание ошибок при помощи конструкций «try...except»; формирование запросов к базе данных.

Логинова С.А. отвечала за нормализацию сущностей, составление ER-диаграммы, а также подготовки презентации и текста отчета. Для реализации первого блока поручений потребовалось понимание материалов лекций по первой (1НФ), второй(2НФ) и третьей(3НФ) нормальным формам; для воплощения второго ー знание нотации UML и умение работать с онлайн-системами для составления ER-диаграмм. Подготовка презентации и текста отчета включала в себя навык работы с MS Word и MS PowerPoint.

# Предметная область

Организация-заказчик: городская студенческая биржа труда и обучения Санкт-Петербургского государственного университета технологии и дизайна (Региональный центр содействия трудоустройству и адаптации к рынку труда выпускников), адрес сайта и работающей версии системы: www.embit.ru

Городская студенческая биржа труда и обучения (ГСБТО) Санкт-Петербургского государственного университета технологии и дизайна, является также Региональным центром содействия трудоустройству и адаптации к рынку труда выпускников.

Задача: спроектировать и разработать базу данных с учетом современных тенденций и требований. Основным назначением Системы является содействие в трудоустройстве студентов, выпускников ВУЗов.

База данных должна позволять решать следующие задачи:

1. предоставлять возможность поиска вакансий и резюме соискателей в базе данных системы;
2. предоставлять возможность добавлять новые вакансии и резюме в базу данных системы, а также изменять уже существующие данные;
3. позволять открывать новые филиалы (пункты, где студентам будут оказывать содействие в трудоустройстве) в других городах (с помощью сотрудников этих филиалов и самостоятельно) с возможностью добавления компаний и вакансий в эти филиалы;
4. позволять формировать реестр компаний, трудоустраивающих студентов и выпускников, предоставляющих места для прохождения практик.

Перед группой стояла следующие цели:

* спроектировать реляционную базу данных (список сущностей, список атрибутов сущностей и анализ взаимосвязи между сущностями);
* провести нормализацию полученных сущностей;
* представить ER-диаграмму;
* составить запросы для реализации базы данных;
* предоставить отчет по выполненной работе.

# Ход выполнения нормализации

Нормализация была проведена, исходя из того, что мы имеем две “главные” сущности – ВАКАНСИЯ и РЕЗЮМЕ. Для устранения избыточности информации из сущности ВАКАНСИЯ была выделена сущность ФИЛИАЛ, которая в свою очередь связана с сущностью КОМПАНИЯ, а из сущности РЕЗЮМЕ была выделена повторяющаяся информация о владельце резюме, которая стала основой для создания сущности СОИСКАТЕЛЬ. После некоторой доработки проекта была выделена сущность ИЗБРАННОЕ для хранения информации о вакансиях, отобранных пользователем. Связывание сущностей произошло с помощью внешних ключей, чтобы избежать аномалий добавления, удаления и изменения сущностей.

После выделения конкретных сущностей, был сформирован список атрибутов для каждой из них:

СОИСКАТЕЛЬ:

каждый соискатель является авторизованным пользователем системы и имеет следующий набор данных: логин, пароль, имя, фамилию, номер телефона; указание города и статуса, в котором пребывает соискатель (студент/выпускник) является необязательным для заполнения. В качестве ключевого атрибута был использован первичный ключ – login, который заполняется при регистрации пользователем в системе и обязан быть уникальным для каждого человека.

РЕЗЮМЕ:

данная сущность хранит информацию о: занятости, желаемой специальности, опыте работы, необходимости проходить или не проходить практику. При этом тип занятости, наличие практики и опыт не являются обязательными к заполнению соискателем. В качестве ключевого атрибута был использован суррогатный ключ – id, в качестве внешнего ключа, который связывает резюме и сущность СОИСКАТЕЛЬ, взят атрибут login.

КОМПАНИЯ:

каждая компания имеет обязательные к заполнению данные о: названии и пароле (для обеспечения входа в систему уполномоченными сотрудниками), сайте, почте, а также для облегчения подбора работников, в данную сущность внесены сведения о практике, т.к. от конкретной компании зависит, предоставляет она возможность оформлении практики для студентов или нет. Сведения о филиалах компании, расположенных по определенным адресам, вынесены в отдельную сущность ФИЛИАЛ. В качестве ключевого атрибута выбран первичный ключ – name, являющийся также уникальным логином для входа в систему сотрудниками.

ФИЛИАЛ:

в сущности ФИЛИАЛ хранится информация о городе, в котором открыт филиал той или иной компании. В качестве ключевого атрибута выбран суррогатный ключ – id и внешний ключ, связывающий филиал с сущностью КОМПАНИЯ, – nameOfCompany.

ВАКАНСИЯ:

в данной сущности выделена обязательная информация о: филиале, наименовании профессии, по которой осуществляется набор сотрудников; и не обязательная о: типе занятости, зарплате, опыте работы. В качестве ключевого атрибута выбран суррогатный ключ – id; в качестве внешнего ключа использован атрибут idOfFil, связывающий вакансию с сущностью ФИЛИАЛ соответственно.

ИЗБРАННОЕ

сущность предназначена для сохранения пользователем (представителем компании или соискателем) подборки вакансия – резюме (много) или резюме - вакансии (много). Параметр status представляет собой текстовый комментарий, обозначающий состояние данной подборки: может содержать, например, комментарий в виде "активная подборка" или "архив". Параметр holder содержит информацию о том, кто создал данную подборку - соискатель или представитель компании; информация заносится в данную графу автоматически.

# ER – диаграмма

# Объяснение выбранной СУБД

Для реализации поставленных задач, выбор команды разработчиков пал на модуль SQLite3, основанный на базе двух языков – Python версии 3.7 и SQL. Он был выбран в качестве СУБД так как, во-первых, является встроенной библиотекой для языка программирования Python и не требует дополнительной установки модулей; во-вторых, является удобным интерпретатором языка SQL и позволяет писать запросы к БД, не прибегая к знанию иных фреймворков. Выбор основного языка для реализации обусловлен удобством и простотой его синтаксиса, а также наличием элементов и инструментов, подходящих для реализации многих вещей, которые не связанных напрямую с самими запросами к базе данных, но необходимы для взаимодействия с системой и пользователем (наличие функций, классов и т.д).

Задачи и пути их реализации:

1. Создать базу данных, с последующей возможностью управления.

Для создания базы данных с помощью SQLite достаточно только обратиться к модулю и прописать SQL запрос на создание таблиц.

Для управления БД были созданы функции на Python с использованием SQLite для обращения к самой БД и манипуляцией над данными внутри БД.

1. Связать таблицы внутри БД.

SQLite предоставляет возможность работы со связанными с помощью внешних ключей таблиц. Это позволяет достигнуть целостности БД, а также избежать лишних действий при изменении и удалении каких-либо записей в связанных таблицах.

1. Прописать отлов возможных ошибок SQL запросов.

Для отлова ошибок SQL запросов были использованы конструкция Python “try...except...else” и атрибут SQLite–объекта “Error”. С помощью них можно узнать, что именно стало причиной ошибки, и продолжить работу с программой без краша.

# Исходный текст запросов

## По созданию таблиц

PRAGMA foreign\_keys = on; //для подключения возможности связывания таблиц через внешние ключи

CREATE TABLE IF NOT EXISTS companies //создание таблицы компаний

(login VARCHAR PRIMARY KEY,

password CHAR(8) NOT NULL,

site VARCHAR NOT NULL,

email VARCHAR NOT NULL,

practice VARCHAR NOT NULL);

CREATE TABLE IF NOT EXISTS filiations //создание таблицы филиалов

(id INTEGER PRIMARY KEY,

nameOfComp VARCHAR NOT NULL,

city VARCHAR NOT NULL,

FOREIGN KEY (nameOfComp) REFERENCES companies(login) ON UPDATE CASCADE ON DELETE CASCADE);

CREATE TABLE IF NOT EXISTS vacancies //создание таблицы вакансий

(id INTEGER PRIMARY KEY,

name VARCHAR NOT NULL,

idOfFil INTEGER NOT NULL,

employment VARCHAR,

pay INTEGER,

experience VARCHAR,

FOREIGN KEY (idOfFil) REFERENCES filiations(id) ON UPDATE CASCADE ON DELETE CASCADE);

CREATE TABLE IF NOT EXISTS users //создание таблицы соискателей

(login VARCHAR PRIMARY KEY,

password CHAR(8) NOT NULL,

fname VARCHAR NOT NULL,

lname VARCHAR NOT NULL,

phone CHAR(11) NOT NULL,

city VARCHAR,

status VARCHAR);

CREATE TABLE IF NOT EXISTS resumes //создание таблицы резюме

(id INTEGER PRIMARY KEY,

login VARCHAR NOT NULL,

profession VARCHAR NOT NULL,

experience VARCHAR,

employment VARCHAR,

practice VARCHAR,

FOREIGN KEY (login) REFERENCES users(login) ON UPDATE CASCADE ON DELETE CASCADE)

При изменении и удалении в родительской таблице данные дочерних таблиц будут изменяться или удаляться каскадно, т.е. при изменении родительской таблицы в дочерней таблице также появятся эти изменения, при удалении данных родительской таблицы в дочерней таблице данные также удалятся.

## Для манипуляции над БД

1. Добавление новой записи в таблицу

INSERT INTO {table} VALUES {params}

table – название таблицы  
params – значения столбцов таблицы

1. UPDATE {table} SET ({columnsStr}) = ({params}) WHERE {prCol} = "{str(prKey)}"

table – название таблицы  
columnsStr – строка с названиями столбцов таблицы, которые должны быть изменены  
params – значения столбцов таблицы, на которые изменяются старые значения  
prCol – название столбца первичного ключа таблицы  
prKey – значение первичного ключа, по которому выбирается запись для изменения

1. DELETE FROM {table} WHERE {column} = "{str(value)}"

table – название таблицы  
column – название столбца, по которому происходит выбор данных для удаления (реализовано так, что в этот параметр передается название столбца первичного ключа)  
value – значение столбца, по которому производится выбор данных для удаления

1. SELECT \* FROM {table.name} WHERE

table.name – название таблицы, по которой происходит поиск  
Данная команда была реализована для просмотра всей таблицы в отдельных случаях

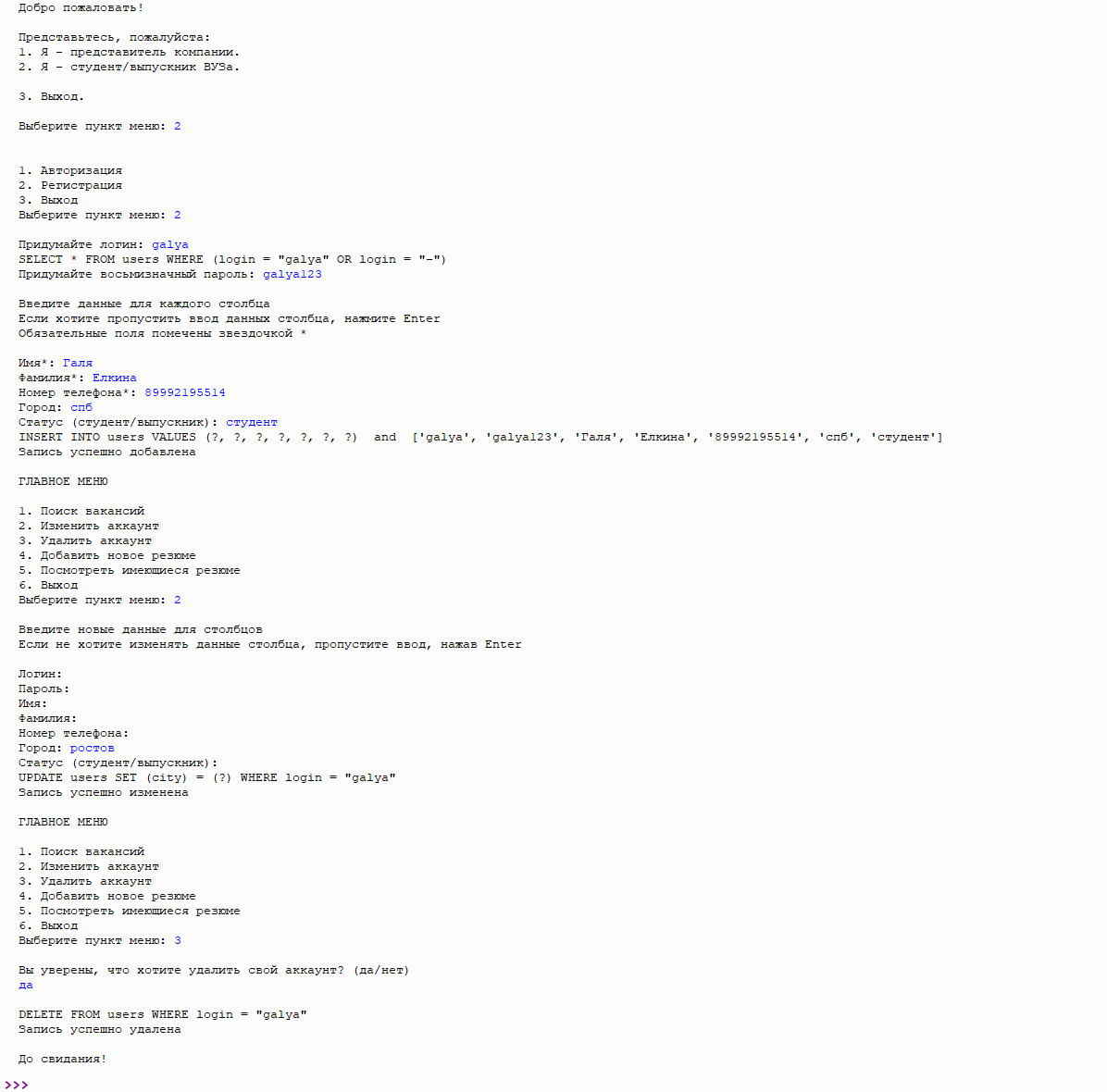
1. SELECT \* FROM {table.name} WHERE ({columns[i]} = "{str(values[i])}"[ OR {sameColomn} = "{nextValue}"])[ AND (nextConstruction)]

table.name – название таблицы, по которой происходит поиск  
columns[i] – название текущего столбца, по которому производится поиск  
values[i] – значение текущего столбца  
sameColomn – название текущего столбца: для реализации поиска по нескольким значениям одного столбца  
nextValue – следующее значение для текущего столбца  
nextConstruction – конструкция, подобная предыдущей, для реализации поиска по нескольким столбцам

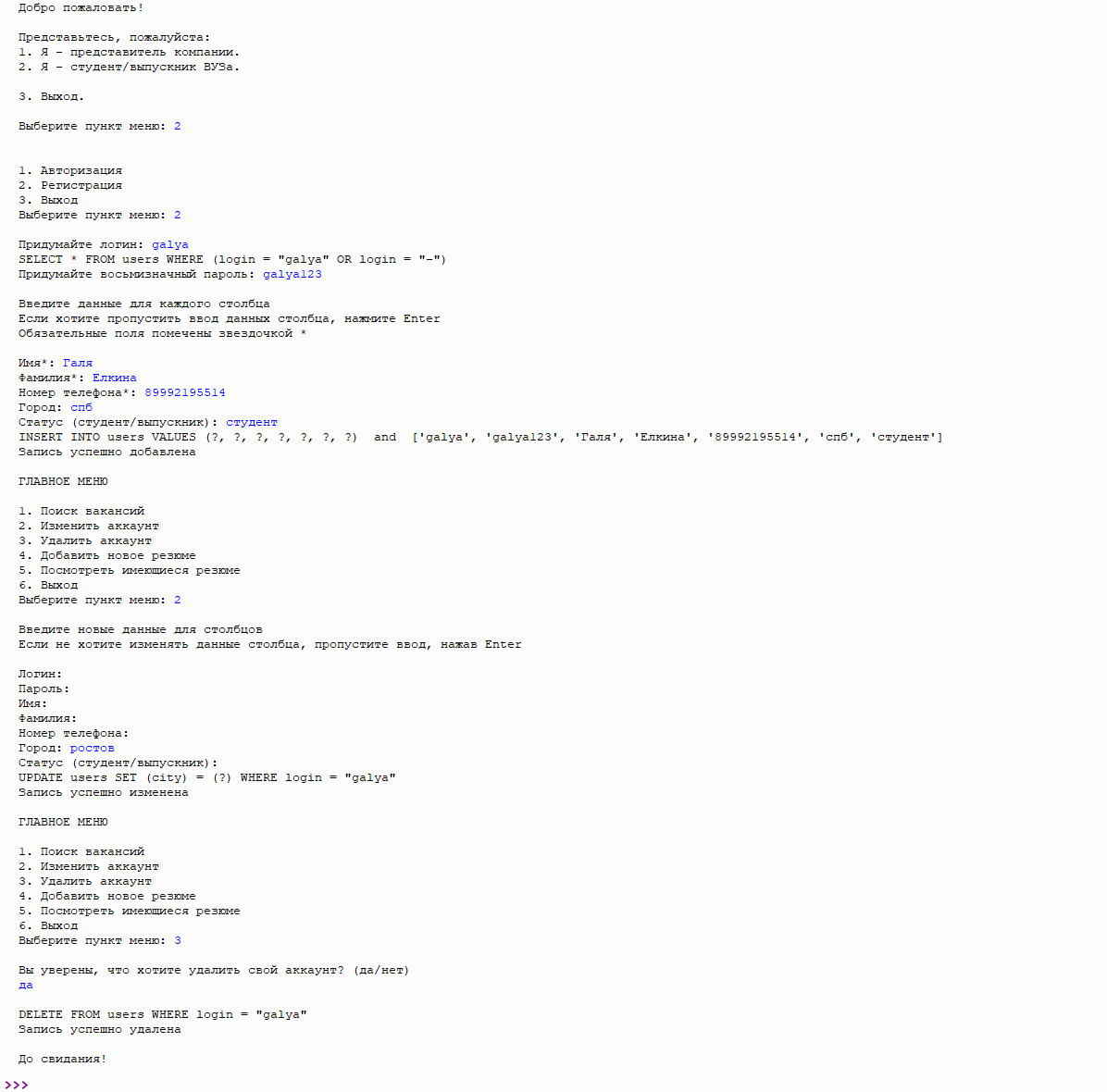
## **Примеры запросов и работа программы**

Рассмотрим сначала простые запросы и начало работы в программе.

Начиная работу в программе, пользователь регистрируется, и делается соответствующий запрос в базу данных для проверки уникальности логина нового пользователя, чтобы не возникало ошибок:

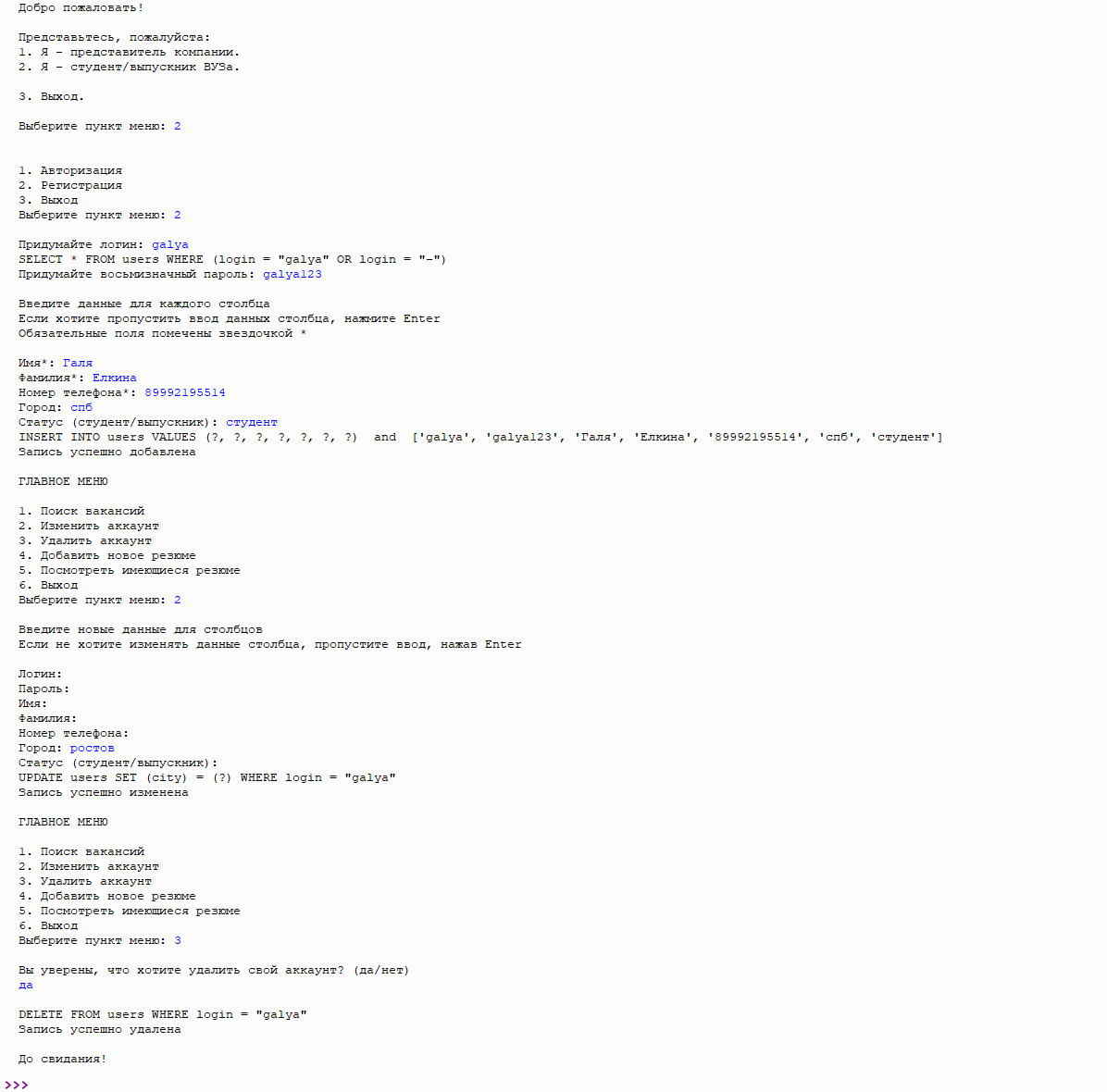


Далее происходит добавление пользователя:

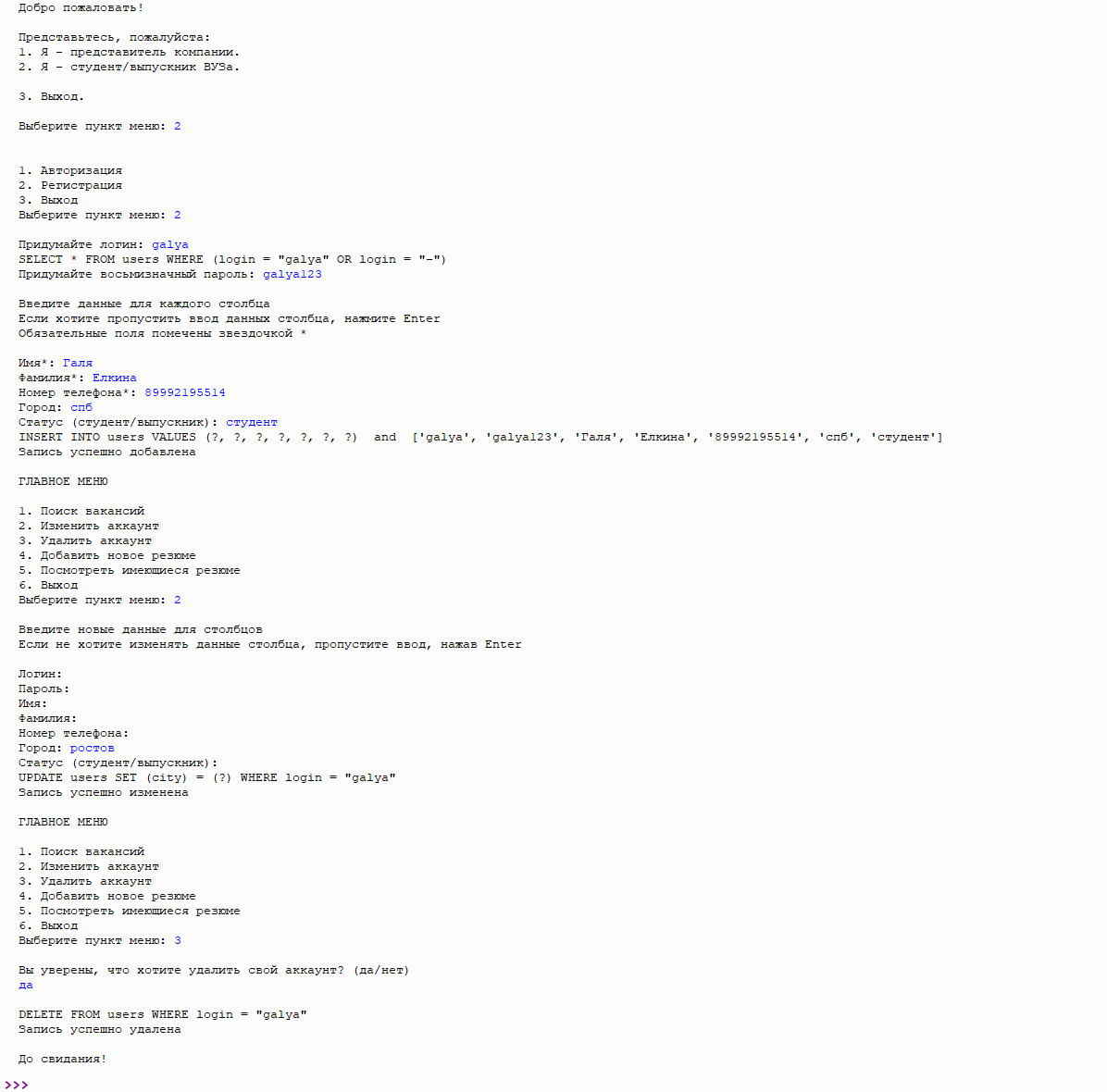


И пользователь попадает в главное меню, соответствующее его статусу (студент/выпускник или представитель компании).

При помощи данного меню пользователь может изменить данные своего аккаунта и не только:



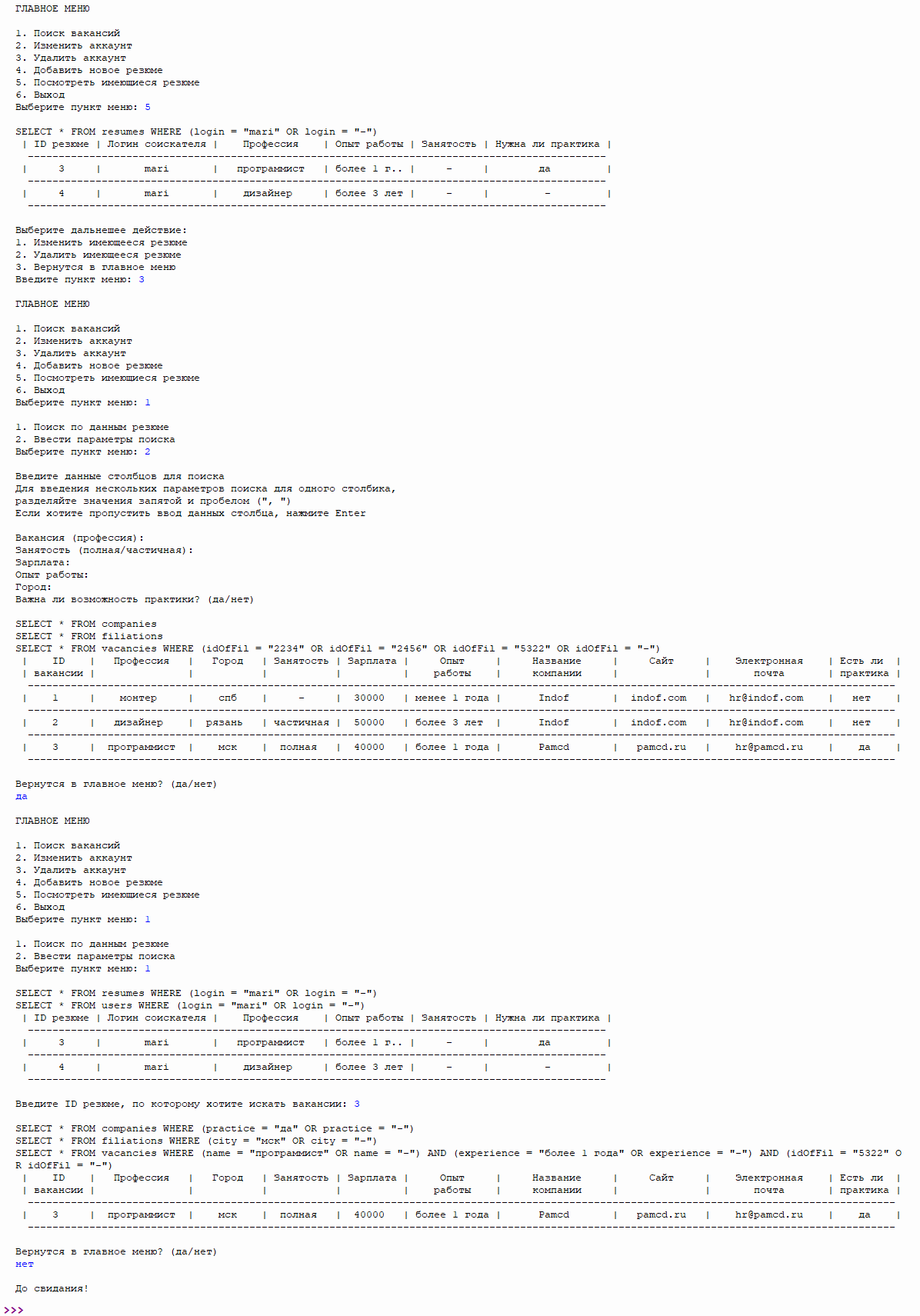
Также через главное меню можно удалить какую-либо доступную пользователю запись. Так как аккаунт, с которым мы работали нам не нужен, мы удалим его.



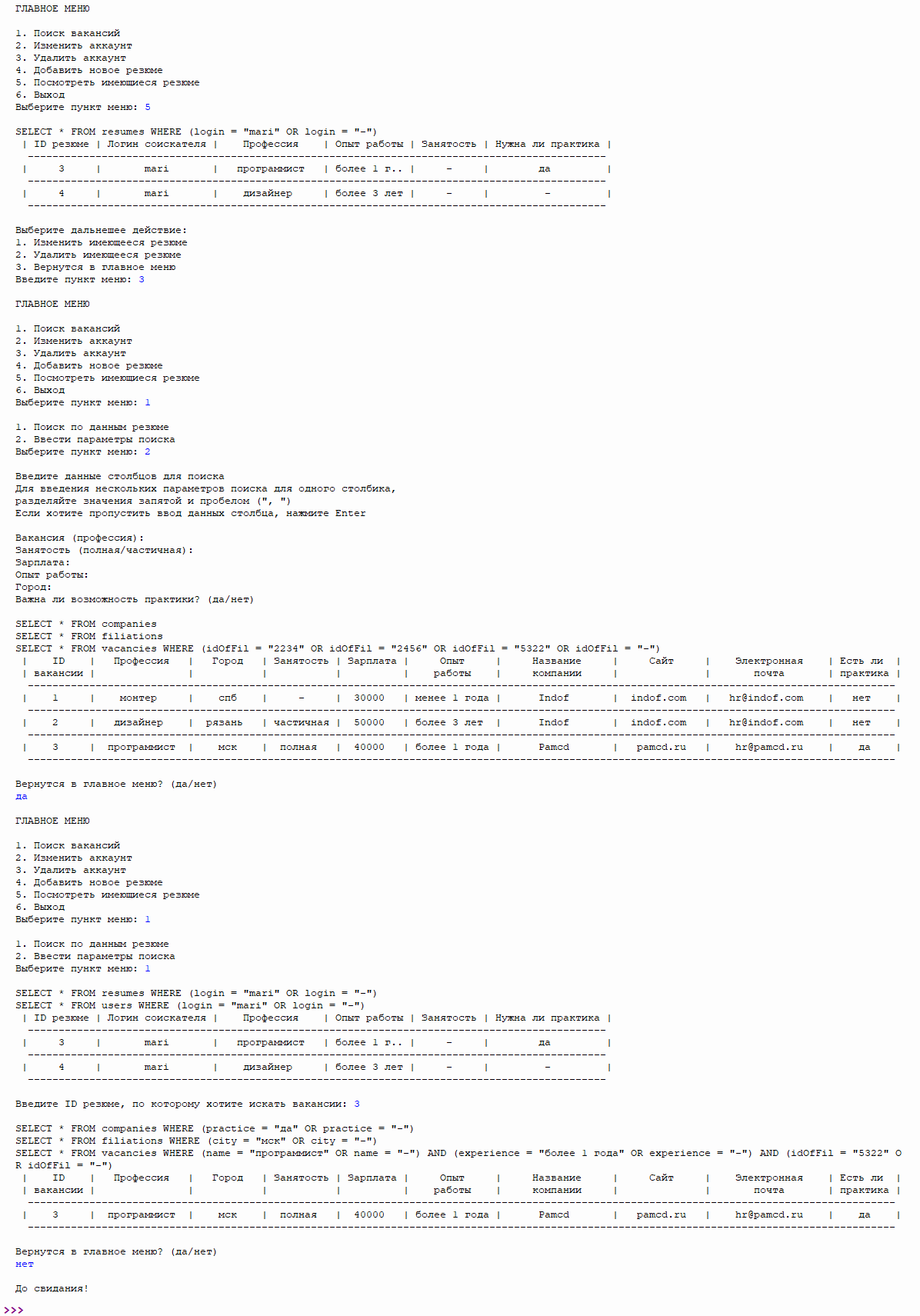
Итак, наша программа умеет составлять простые запросы для манипуляцией базой данных. Но также мы предусмотрели возможность довольно сложных запросов.

Теперь зайдем под логином уже зарегистрированного пользователя. И чтобы проверить работу сложных поисковых запросов перейдем к соответствующему пункту меню.

Ниже показаны действия, которые необходимы, чтобы пользователь мог увидеть все доступные вакансии в базе данных.



Далее вернемся в главное меню и выберем поиск по данным одного из резюме:



Как мы видим, запросы к базе данных получаются не такими уж большими. Но сама программа реализована так, что все запросы делаются в несколько этапов. Поэтому программа составляет несколько запросов и в определенном порядке их выстаивает и обрабатывает, чтобы получить нужный результат.