**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра ИС**

Курсовая РАБОТА

**по дисциплине «Управление данными»**

Тема: Разработка приложения для работы с базой данных для работников регистратуры поликлиники

**Вариант 14**

|  |  |  |
| --- | --- | --- |
| Студент гр. 1281 |  | Лазарчик М.П. |
| Преподаватель |  | Татарникова Т.М. |

Санкт-Петербург

2023

**Задание**

**на курсовую работу**

Студент Лазарчик М.П.

Группа 1281

Тема работы: Разработка приложения для работы с базой данных для работников регистратуры поликлиники.

Исходные данные:

Спроектировать базу данных, построить программу, обеспечивающую взаимодействие с ней в режиме диалога, для работников регистратуры поликлиники. В БД хранятся сведения о больных: ФИО, адрес, диагноз (может быть уточнен), дате заболевания; сведения о врачах: ФИО, номер кабинета, номер участка, дни и часы приёма; описание болезней: название (диагноз), симптомы, лекарство. Возможно появление новых больных. Врач может уволиться из поликлиники.

Содержание пояснительной записки:

Введение, анализ предметной области, обоснование выбора модели данных, обоснование выбора СУБД, описание функций групп пользователей, описание функций управления данными, организация защиты БД, заключение, список использованных источников, руководство пользователя БД, листинг программного кода.

Предполагаемый объём пояснительной записки:

Не менее 20 страниц.

Дата сдачи реферата: 29.05.2023

|  |  |  |
| --- | --- | --- |
| Студент гр. 1281 |  | Лазарчик М.П. |
| Преподаватель |  | Татарникова Т.М. |

**АННОТАЦИЯ**

Данная работа заключается в формировании практических навыков проектирования баз данных. В ходе выполнения данной курсовой работы была спроектирована и разработана база данных, связанная с работой регистратуры в поликлиники, а также реализовано приложение с интерфейсом на языке программирования python, которое позволяет работать с этой базой данных.

**SUMMARY**

This course work consists in the formation of practical skills in database design. In the course of this course work, databases related to working registration in clinics were designed and obtained, and a interface application for the python programming language was implemented that allows you to work with this database.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 5](#_Toc27349222)

[1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 6](#_Toc27349223)

[2. ОБОСНОВАНИЕ МОДЕЛИ ДАННЫХ 6](#_Toc27349224)

[3. ОБОСНОВАНИЕ ВЫБОРА СУБД 7](#_Toc27349225)

[4. ОПИСАНИЕ ФУНКЦИЙ ГРУПП ПОЛЬЗОВАТЕЛЕЙ 7](#_Toc27349226)

[5. ОПИСАНИЕ ФУНКЦИЙ УПРАВЛЕНИЯ ДАННЫМИ 8](#_Toc27349227)

[6. ОРГАНИЗАЦИЯ ЗАЩИТЫ БД 8](#_Toc27349228)

[ЗАКЛЮЧЕНИЕ 9](#_Toc27349229)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 10](#_Toc27349230)

[ПРИЛОЖЕНИЕ А. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 11](#_Toc27349231)

[ПРИЛОЖЕНИЕ Б. ЛИСТИНГ ПРОГРАММНОГО КОДА 14](#_Toc27349232)

### ВВЕДЕНИЕ

Процесс работы регистратуры поликлиники предполагает наличие больших объёмов информации о больных (ФИО, адрес, диагноз), о врачах(ФИО, номер кабинета, номер участка и т.д.) , о болезнях(их название, симптомы, лекарство), кроме того, необходимо управлять данными(добавлять новых больных и удалять вылечившихся, изменение болезни в результате появления новых симптомов и т.д.) , также необходимо я вести отчётность о работе поликлиники, включающего в себя: количество больных, заболеваний, врачей и многое другое. Из этого следует необходимость в удобном для работников регистратуры, эффективном и структурированном хранении информации в электронном виде. Следовательно, в такой ситуации логичным будет создание реляционной базы данных.

### 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

Проектирование базы данных осуществляется для работников поликлиники. В базе данных определены следующие сущности: пациенты, диагнозы, врачи. Атрибуты сущностей: идентификационный номер, ФИО, адрес, дата заболевания, диагноз, лечащий врач; идентификационный номер, название, симптомы и лечение; идентификационный номер, ФИО, номер кабинета и участка, дни и часы приёма.

Работникам регистратуры могут потребоваться следующие сведения: адрес, дата заболевания, диагноз данного больного; ФИО лечащего врача данного больного; номер кабинета, дни и часы приёма данного врача; больные, находящиеся на лечении у данного врача; симптомы данного заболевания и рекомендуемое лекарство. Увольнять врача, изменять диагноз и вносить информацию о новом пациенте может только администратор.

Предусмотрена выдача отчёта о работе поликлиники, который предусматривает информирование о количестве больных, ФИО каждого врача и число лечащихся у него больных, количестве заболеваний по каждому виду болезни, расписании работы врачей поликлиники.

### 2. ОБОСНОВАНИЕ МОДЕЛИ ДАННЫХ

Для реализации базы данных была выбрана реляционная модель, так как данные в таком случае представлены объектов и связей между ними в виде таблиц, что в наивысшей мере подходит для данного варианта задания. Данная модель характерна простотой и доступностью для понимания конечным пользователем, а также наличием математического аппарата для работы с отношениями. Реляционная модель базы данных обладает следующими свойствами:

1. Каждый элемент таблицы соответствует одному элементу данных.

2. Все столбцы в таблице однородные.

3. Каждый столбец имеет уникальное имя.

4. Одинаковые строки в таблице отсутствуют.

5. Порядок следования строк и столбцов может быть произвольным.

### 3. ОБОСНОВАНИЕ ВЫБОРА СУБД

Для данной курсовой работы было решено использовать СУБД MySQL, которая может использоваться как SQL-сервер благодаря поддержке языка SQL. MySQL отличается высокой скоростью работы, высоким уровнем безопасности и надежностью. Еще одним плюсом использования MySQL является то, что эта СУБД бесплатна и имеет открытый исходный код, что позволяет пользователям добавлять свои собственные настройки.

В разработке приложения для работы с базой данных использовалась библиотека pymysql для языка программирования Python.

### 4. ОПИСАНИЕ ФУНКЦИЙ ГРУПП ПОЛЬЗОВАТЕЛЕЙ

При работе с базой данных в приложении пользователи делятся на две группы: сотрудники и администратор

Сотрудник регистратуры может запрашивать следующую информацию (SELECT):

1) адрес, дата заболевания, диагноз данного больного;

2) ФИО лечащего врача данного больного;

3) номер кабинета, дни и часы приёма данного врача;

4) больные, находящиеся на лечении у данного врача;

5) симптомы данного заболевания и рекомендуемое лекарство.

Администратор БД может вносить следующие изменения:

1) появление нового больного (INSERT);

2) увольнение врача (DELETE);

3) изменение диагноза (UPDATE).

4) получение отчета о работе поликлиники

### 5. ОПИСАНИЕ ФУНКЦИЙ УПРАВЛЕНИЯ ДАННЫМИ

Функции управления данными:

1. Хранение – создание информационных объектов «Пациенты», «Диагнозы», «Врачи».

2. Манипулирование данными – добавление нового пациента, обновление диагноза, увольнение врача, получение запрашиваемой информации.

3. Доступ к данным – ограничение прав доступа путём авторизации в приложении.

Управление данными осуществляется с помощью SQL-запросов, которые отправляются приложением на сервер, – процедура осуществляется вариативно и зависит от действий пользователя. Получение справки и отчёта происходит путём создания текстового файла с последующей записью необходимой информации в этот файл.

### 6. ОРГАНИЗАЦИЯ ЗАЩИТЫ БД

### В процессе разработки и проектирования исходной базы данных были соблюдены все ограничения целостности, которые включали в себя определение типов данных для каждого информационного объекта, установление первичных ключей для таблиц и обеспечение ссылочной целостности базы данных.

### В целях обеспечения безопасности базы данных и ограничения доступа был создан логин и пароль. При запуске приложения, пользователю необходимо пройти авторизацию, что гарантирует безопасность базы данных.

### ЗАКЛЮЧЕНИЕ

В рамках данной курсовой работы, была спроектирована и создана реляционная база данных, используя язык SQL и СУБД MySQL. База данных содержит информацию о пациентах, диагнозах и врачах. Для работы с базой данных было разработано приложение с интерфейсом на языке Python с использованием библиотеки mysql.connector.

Функционал приложения разделен на две группы пользователей. Пользователи имеют возможность получить отчет о болезни любого пациента и справку о работе поликлиники. База данных была проверена на наличие аномалий, и безопасность базы данных была гарантирована. База данных поддерживает весь требуемый функционал и готова к использованию.

### СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Онлайн курс по дисциплине «Управление данными» // open.etu.ru (Тататрникова Т.М.).

2. Конспект лекций онлайн-курса по дисциплине «Управление данными» // open.etu.ru (Тататрникова Т.М.).

### 

### ПРИЛОЖЕНИЕ А. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

1. Краткое описание программы:

Данная программа предназначена для работы с базой данных для работников поликлиники.

2. Минимальные системные требования:

Процессор Pentium с частотой 233 МГц или более быстрый (рекомендуется не менее 300 МГц); не менее 64 МБ оперативной памяти (рекомендуется не менее 128 МБ); не менее 1,5 ГБ свободного места на жестком диске; видеокарта и монитор, поддерживающие режим Super VGA с разрешением не менее чем 800x600 точек.

3. Установка программы:

Переместите папку config.py Starting\_Data.py и Interface.py

и в произвольную директорию, запустите скрипт sql удобным способом. На компьютере должен быть установлен MySQL Server.

4. Запуск программы:

Запустите файлы exe.

5. Работа с программой:

При запуске программы пользователю необходимо авторизоваться, как показано на рисунке 1.

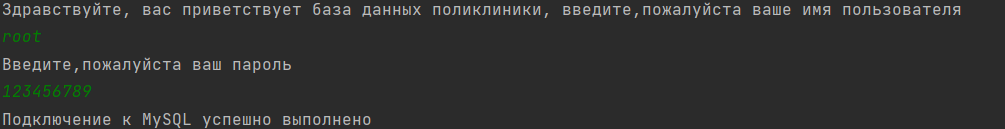


Рисунок 1 – Авторизация пользователя

Далее, работа с программой осуществляется на интуитивно понятном уровне, так как интерфейсом предусмотрено описание всех манипуляций. На рисунке 2 изображён интерфейс Администратора

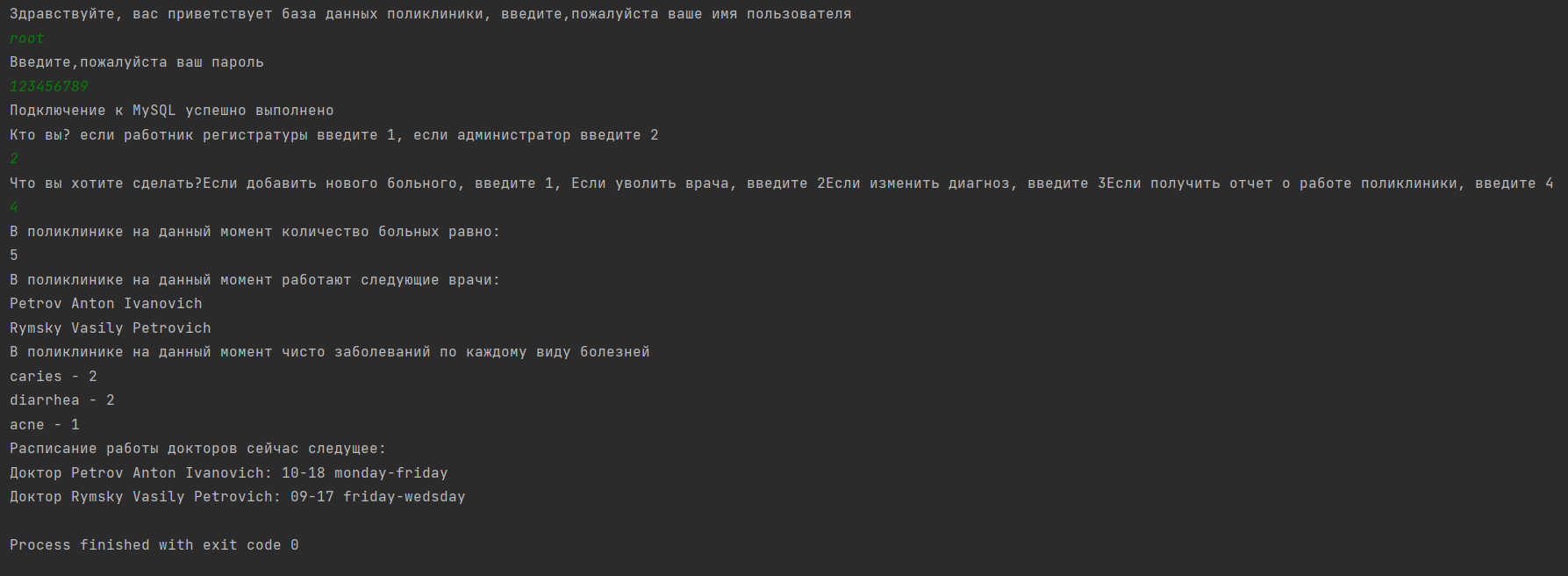


Рисунок 2 – Интерфейс администратора поликлиники

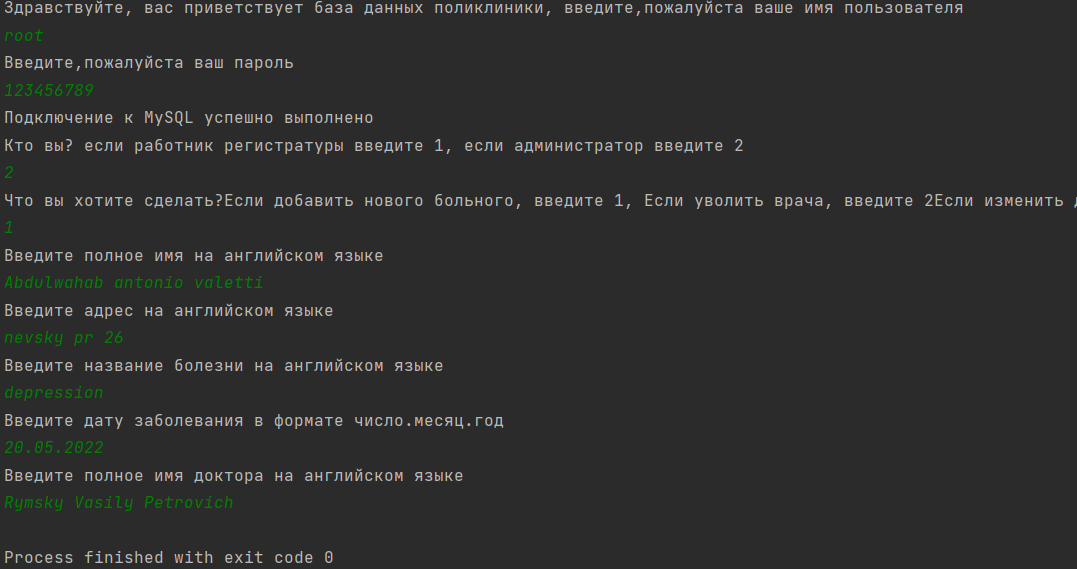


Рисунок 3 – Добавление нового пациента

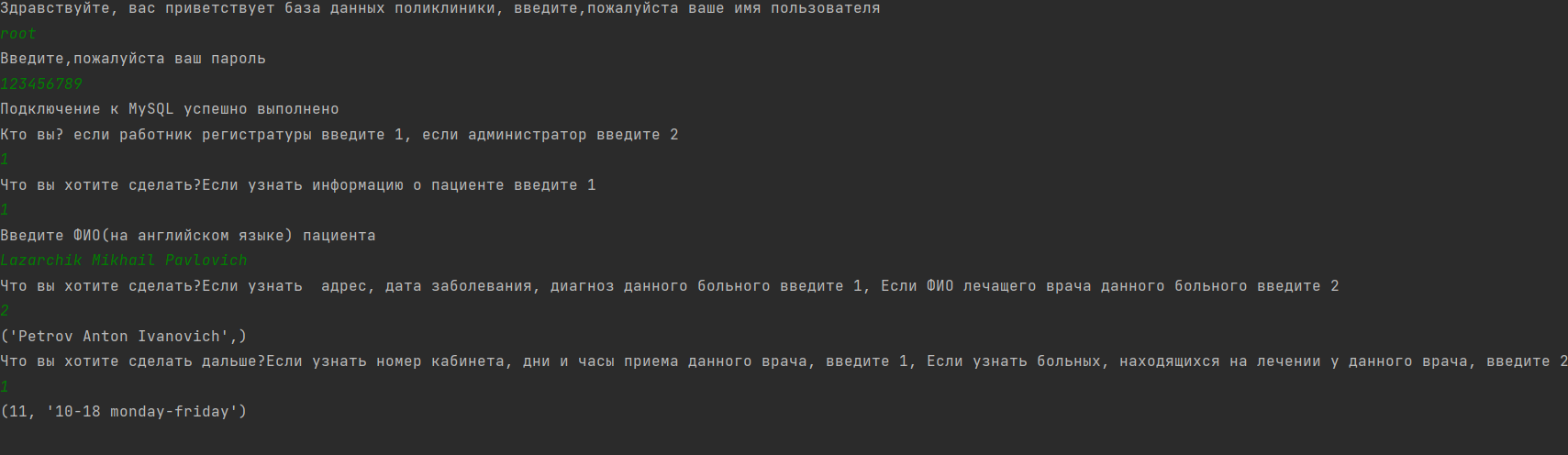


Рисунок 4 – Интерфейс работника регистратуры

### ПРИЛОЖЕНИЕ Б. ЛИСТИНГ ПРОГРАММНОГО КОДА

**Записываем информацию необходимую для подключения к субд**

config.py:

db\_config = {  
 "mysql": {  
 "host": "localhost",  
 "user": "root",  
 "pass": "123456789"  
 }  
}

**Создаем базу данных в MySQL используя python**

Starting\_Data.py

import mysql.connector  
from mysql.connector import Error  
from config import db\_config  
  
def create\_connection\_mysql\_db(db\_host, user\_name, user\_password, db\_name = None):  
 connection\_db = None  
 try:  
 connection\_db = mysql.connector.connect(  
 host = db\_host,  
 user = user\_name,  
 passwd = user\_password,  
 database = db\_name  
 )  
 print("Подключение к MySQL успешно выполнено")  
 except Error as db\_connection\_error:  
 print("Возникла ошибка: ", db\_connection\_error)  
 return connection\_db  
  
  
conn = create\_connection\_mysql\_db(db\_config["mysql"]["host"],  
 db\_config["mysql"]["user"],  
 db\_config["mysql"]["pass"])  
cursor = conn.cursor()  
create\_db\_sql\_query = 'CREATE DATABASE {}'.format('Poliklinika')  
cursor.execute(create\_db\_sql\_query)  
cursor.close()  
conn.close()  
  
conn = create\_connection\_mysql\_db(db\_config["mysql"]["host"],  
 db\_config["mysql"]["user"],  
 db\_config["mysql"]["pass"],  
 "Poliklinika")  
try:  
 # создание таблицы с пациентами  
 cursor = conn.cursor()  
 create\_table\_query = '''  
 CREATE TABLE IF NOT EXISTS patients (  
 id INT AUTO\_INCREMENT,   
 fullname TEXT NOT NULL,   
 address TEXT NOT NULL,  
 diagnosis TEXT NOT NULL,  
 date\_of\_disease TEXT NOT NULL,  
 doctor TEXT NOT NULL,   
 PRIMARY KEY (id)  
 ) ENGINE = InnoDB'''  
 cursor.execute(create\_table\_query)  
 conn.commit()  
  
 # создание таблицы с докторами  
 cursor = conn.cursor()  
 create\_table\_query = '''  
 CREATE TABLE IF NOT EXISTS doctors (  
 id INT AUTO\_INCREMENT,   
 fullname TEXT NOT NULL,   
 room\_number INT,  
 zone\_number INT,  
 date\_and\_time\_of\_work TEXT,   
 PRIMARY KEY (id)  
 ) ENGINE = InnoDB'''  
 cursor.execute(create\_table\_query)  
 conn.commit()  
  
 # создание таблицы с болезнями  
 cursor = conn.cursor()  
 create\_table\_query = '''  
 CREATE TABLE IF NOT EXISTS diseases (  
 id INT AUTO\_INCREMENT,   
 name TEXT NOT NULL,  
 symptoms TEXT NOT NULL,   
 medicine TEXT NOT NULL,   
 PRIMARY KEY (id)  
 ) ENGINE = InnoDB'''  
 cursor.execute(create\_table\_query)  
 conn.commit()  
  
 # вставка данных в таблицу с пациентами изначальной инфы  
 insert\_patients\_table\_query = '''  
 INSERT INTO  
 `patients` (`fullname`, `address`, `diagnosis`, `date\_of\_disease`, `doctor`)   
 VALUES  
 ('Pavlova Antonina Mikhailovna', 'nevsky prospect 24', 'herpes', '24.05.2023', 'Petrov Anton Ivanovich'),  
 ('Germanenko Pytr Vasilyevich', 'ligovsky prospect 33', 'diarrhea', '26.05.2023', 'Rymsky Vasily Petrovich'),  
 ('Timiryazev Antonin Pavlovich', 'petrovsky prospect 4', 'acne', '20.05.2023', 'Petrov Anton Ivanovich'),  
 ('Klimashin Igor Pethovich', 'oktyabrskaya embankment 15', 'diarrhea', '24.05.2023', 'Rymsky Vasily Petrovich');'''  
 cursor.execute(insert\_patients\_table\_query)  
 conn.commit()  
  
 # вставка данных в таблицу с докторами изначальной инфы  
 insert\_doctors\_table\_query = '''  
 INSERT INTO  
 `doctors` (`fullname`, `room\_number`, `zone\_number`, `date\_and\_time\_of\_work` )  
 VALUES  
 ('Petrov Anton Ivanovich', 11, '1', '10-18 monday-friday'),  
 ('Rymsky Vasily Petrovich', 15, '2', '09-17 friday-wedsday');'''  
 cursor.execute(insert\_doctors\_table\_query)  
 conn.commit()  
  
 # вставка данных в таблицу с болезнями изначальной инфы  
 insert\_diseases\_table\_query = '''  
 INSERT INTO  
 `diseases` (`name`,`symptoms`,`medicine`)  
 VALUES  
 ('herpes', 'small red dots', 'allomedin'),  
 ('diarrhea', 'irritable bowel', 'smekta'),  
 ('acne','big red dots', 'basiron');'''  
 cursor.execute(insert\_diseases\_table\_query)  
 conn.commit()  
  
except Error as error:  
 print(error)  
finally:  
 cursor.close()  
 conn.close()

**Создаем интерфейс для удобства пользования работниками**

Interface.py

import mysql.connector  
from mysql.connector import Error  
  
db\_host="localhost"  
print("Здравствуйте, вас приветствует база данных поликлиники, введите,пожалуйста ваше имя пользователя")  
user\_name=input()  
print("Введите,пожалуйста ваш пароль")  
user\_password=input()  
  
def create\_connection\_mysql\_db(db\_host, user\_name, user\_password, db\_name = None):  
 connection\_db = None  
 try:  
 connection\_db = mysql.connector.connect(  
 host = db\_host,  
 user = user\_name,  
 passwd = user\_password,  
 database = db\_name  
 )  
 print("Подключение к MySQL успешно выполнено")  
 except Error as db\_connection\_error:  
 print("Возникла ошибка: ", db\_connection\_error)  
 return connection\_db  
conn = create\_connection\_mysql\_db(db\_host,  
 user\_name,  
 user\_password,  
 "Poliklinika")  
cursor = conn.cursor()  
  
print("Кто вы? если работник регистратуры введите 1, если администратор введите 2")  
post=int(input())  
if post==1:  
 print("Что вы хотите сделать?"  
 "Если узнать информацию о пациенте введите 1")  
 what1 = int(input())  
 if what1 == 1:  
 print("Введите ФИО(на английском языке) пациента")  
 what2 = str(input())  
 print("Что вы хотите сделать?"  
 "Если узнать адрес, дата заболевания, диагноз данного больного введите 1, "  
 "Если ФИО лечащего врача данного больного введите 2")  
 what3 = int(input())  
 if what3 == 1:  
 select\_patient\_query ="SELECT address, date\_of\_disease, diagnosis FROM patients WHERE fullname = %s"  
 cursor.execute(select\_patient\_query, ((what2,)))  
 query\_result = cursor.fetchall()  
 for x in query\_result:  
 z=x  
 print(x)  
 print("Что вы хотите сделать?"  
 "Cимптомы данного заболевания и рекомендуемое лекарство введите 1")  
 what34 = int(input())  
 if what3 == 1:  
 select\_patient\_query ="SELECT symptoms, medicine FROM diseases WHERE name = %s"  
 cursor.execute(select\_patient\_query, ((z[2],)))  
 query\_result = cursor.fetchall()  
 for x in query\_result:  
 print(x)  
  
 elif what3 == 2:  
 select\_patient\_query = "SELECT doctor FROM patients WHERE fullname = %s"  
 cursor.execute(select\_patient\_query, ((what2,)))  
 query\_result = cursor.fetchall()  
 for x in query\_result:  
 y=x  
 print(x)  
 print("Что вы хотите сделать дальше?"  
 "Если узнать номер кабинета, дни и часы приема данного врача, введите 1, "  
 "Если узнать больных, находящихся на лечении у данного врача, введите 2")  
 what4 = int(input())  
 if what4 == 1:  
 select\_patient\_query = "SELECT room\_number, date\_and\_time\_of\_work FROM doctors WHERE fullname = %s"  
 cursor.execute(select\_patient\_query, ((y[0],)))  
 query\_result = cursor.fetchall()  
 for x in query\_result:  
 print(x)  
  
 else:  
 select\_patient\_query = "SELECT fullname FROM patients WHERE doctor = %s"  
 cursor.execute(select\_patient\_query, ((y[0],)))  
 query\_result = cursor.fetchall()  
 for x in query\_result:  
 print(x)  
  
  
  
elif post==2:  
  
 print("Что вы хотите сделать?"  
 "Если добавить нового больного, введите 1, "  
 "Если уволить врача, введите 2"  
 "Если изменить диагноз, введите 3"  
 "Если получить отчет о работе поликлиники, введите 4")  
 what5 = int(input())  
 if what5 == 1:  
 print('Введите полное имя на английском языке')  
 fullname = str(input())  
 print('Введите адрес на английском языке')  
 address = str(input())  
 print('Введите название болезни на английском языке')  
 diagnosis = str(input())  
 print('Введите дату заболевания в формате число.месяц.год')  
 date\_of\_disease = str(input())  
 print('Введите полное имя доктора на английском языке')  
 doctor = str(input())  
 val = (fullname, address, diagnosis, date\_of\_disease, doctor)  
 insert\_patients\_table\_query = '''  
 INSERT INTO  
 `patients` (`fullname`, `address`, `diagnosis`, `date\_of\_disease`, `doctor`)   
 VALUES  
 (%s, %s, %s, %s, %s);'''  
 cursor.execute(insert\_patients\_table\_query, val)  
 conn.commit()  
 elif what5 == 2:  
 print('Введите полное имя на английском языке доктора, которого хотите уволить')  
 fulname = str(input())  
 delete\_doctor\_doctors\_query = '''  
 DELETE FROM doctors WHERE fullname = %s;  
 '''  
 cursor.execute(delete\_doctor\_doctors\_query, ((fulname, )))  
 conn.commit()  
 elif what5 == 3:  
 print('Введите имя больного диагноз, которое хотите изменить')  
 patient = str(input())  
 print('Введите диагноз, на который хотите изменить')  
 diag = str(input())  
 val2=(diag, patient)  
 update\_patient\_query = '''  
 UPDATE patients SET diagnosis = %s WHERE fullname = %s;  
 '''  
 cursor.execute(update\_patient\_query, val2)  
 conn.commit()  
 elif what5 == 4:  
 print('В поликлинике на данный момент количество больных равно:')  
 cursor.execute("SELECT COUNT(\*) FROM patients")  
 result = cursor.fetchone()[0]  
 print(result)  
 print('В поликлинике на данный момент работают следующие врачи:')  
 cursor.execute("SELECT fullname FROM doctors")  
 for row in cursor.fetchall():  
 print(row[0])  
 print('В поликлинике на данный момент чисто заболеваний по каждому виду болезней')  
 query = "SELECT diagnosis, COUNT(\*) AS count FROM patients GROUP BY diagnosis"  
 cursor.execute(query)  
 result = cursor.fetchall()  
 for row in result:  
 print(row[0], "-", row[1])  
 print('Расписание работы докторов сейчас следущее:')  
 query = ("SELECT fullname, date\_and\_time\_of\_work FROM doctors")  
 cursor.execute(query)  
 for (fullname, date\_and\_time\_of\_work) in cursor:  
 print(f"Доктор {fullname}: {date\_and\_time\_of\_work}")