МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО
ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г.
ШУХОВА» (БГТУ им. В.Г. Шухова)

Кафедра программного обеспечения вычислительной техники и
автоматизированных систем

Курсовая работа

по дисциплине: Базы данных
тема: «Создание приложения для работы с БД»

Автор работы_____Игнатьев Артур Олегович ПВ-223
Руководитель проекта_____Панченко Максим Владимирович

Оценка _____

Белгород 2024 г.

Содержание

Введение

Курсовая работа представляет собой разработку десктопной системы управления библиотекой с СУБД PostgreSQL. ORM - не используется. Взаимодействие с БД происходит напрямую через библиотеку psycopg2. Для разработки интерфейса выбрана библиотека tkinter. Для хэширования паролей библиотека bcrypt. Для резервного копирования на удалённое хранилище используются библиотеки ftplib и paramico.

Целью данной работы является создание функциональной системы управления библиотекой, которая позволит автоматизировать процессы, связанные с работой библиотеки.

Для достижения поставленной цели были решены следующие задачи:

1. Разработка схемы базы данных на PostgreSQL.
2. Создание интерфейса пользователя с использованием Tkinter для взаимодействия с системой.
3. Реализация функций управления книгами, пользователями и выдачей книг.
4. Реализация разграничения доступа на роли (администратор, библиотекарь, пользователь).
5. Реализация функционала бэкапа и восстановления базы данных.

1. Теоретические основы разработки системы управления библиотекой

1.1. Предметная область

Данная система управления библиотекой призвана автоматизировать основные процессы, происходящие в библиотеке. Эти процессы включают в себя: формирование библиотечного фонда, обработку и каталогизацию новых ресурсов, хранение библиотечного фонда, обслуживание читателей и регистрацию новых читателей и выдачу книг.

В системе выделяются три основные роли, каждая из которых имеет свой набор обязанностей и функционала.

1. Читатель, или обычный пользователь, имеет может просматривать свою историю выданных книг.
2. Библиотекарь ведет учет библиотечного фонда, осуществляет выдачу и прием книг, отвечает за каталогизацию.
3. Администратор — это пользователь с полными правами. В его обязанности входит управление пользователями, назначение ролей и, в том числе, управление резервным копированием базы данных, что бы обеспечить сохранность информации.

В основе предметной области лежат следующие объекты:

1. Книга: Представляет собой основной ресурс библиотеки. Книга имеет название, автор(ов), жанр, год издания, количество экземпляров и состояние (доступна или нет).
2. Автор: Создатель книги. Автор имеет имя, фамилию, а так же может иметь дату рождения.
3. Жанр: Определяет тематическую принадлежность книги (фантастика, детектив, роман и т.д.)
4. Читатель: Пользователь библиотеки, имеющий имя, фамилию,

контактную информацию, ID. Читатель может иметь или не иметь связи с пользователем системы.

5. Выдача книг: Представляет собой связь между книгой и читателем. Содержит информацию о дате выдачи, сроке возврата и дате фактического возврата (если есть).

1.2. Требования к системе

Стояла задача создать полноценную систему, которая помогла бы в работе библиотеки. Поэтому, были выделены несколько ключевых моментов, которые должны быть в системе в плане функциональности, и то, как она должна работать в принципе.

Функциональные требования:

Управление пользователями: система должна давать возможность регистрировать новых пользователей. Все логины, пароли, имена, фамилии, а так же телефоны, если это нужно – все это должно где то хранится, ну и, конечно же, нельзя забыть про безопасность. Система также должна позволять пользователям входить, используя логин и пароль, чтобы убедиться что это именно тот пользователь.

Помимо этого, в системе должны быть роли (администратор, библиотекарь, обычный пользователь), которые определяют, что каждый из них может делать в системе. Должна быть возможность удалять ненужных пользователей.

Управление книгами: библиотекарь должен уметь добавлять в каталог информацию о новых книгах. Это значит, что нужно будет добавлять название книги, авторов, год издания, жанр, количество копий и т.д. Должна быть возможность редактировать информацию о книгах, если что-то изменилось или если нашли ошибку. Также нужно удалять записи о книгах, которых уже нет в фонде.

Управление выдачей книг: система должна давать возможность выдавать книги читателям, регистрировать дату выдачи и срок возврата. Должна быть возможность отмечать возврат книг и вестись история выдачи.

Бэкап и восстановление: должна быть возможность восстановить базу данных из бэкапа.

1.3. Выбор технологий

Первым и самым важным выбором была база данных. После рассмотрения нескольких вариантов, было решено остановиться на PostgreSQL. Почему именно она? PostgreSQL – это мощная и надежная реляционная СУБД, которая предоставляет все необходимые инструменты для хранения и управления данными, что очень важно для такого приложения как библиотека. Она имеет открытый исходный код, что очень важно для некоммерческих проектов, а так же имеет очень хороший функционал. Так как она реляционная база данных, то это позволяет эффективно организовывать данные в таблицы и устанавливать связи между ними, что очень важно для создания системы управления библиотекой.

Для создания графического интерфейса пользователя и основной логики приложения был выбран Python с библиотекой Tkinter. Python – это простой, понятный, и при этом достаточно мощный язык программирования, который хорошо подходит для создания десктопных приложений. А библиотека Tkinter предоставляет все необходимые инструменты для создания графического интерфейса, так же она очень проста в использовании и изучении.

Для работы с базой данных PostgreSQL используется библиотека psycopg2. Она предоставляет удобный и эффективный интерфейс для выполнения SQL-запросов и управления данными. Это позволяет мне напрямую общаться с базой данных, и при этом не усложнять работу с ней.

Также стоит отметить, что для создания резервных копий базы данных используется утилита командной строки pg_dump, и psql для их востановления. Это надежный способ, позволяющий сохранять резервные копии в виде SQL-файлов и гарантирующий что данные не будут утеряны.

Для работы с протоколом FTP я использовал стандартную библиотеку Python ftplib. Так же эта библиотека довольно проста в использовании.

Для защиты данных и хранения паролей используется библиотека bcrypt.

Она обеспечивает надежное хеширование паролей, что делает систему более безопасной.

2. Проектирование и реализация системы

2.5. Модель данных

В базе данных были использованы несколько таблиц, каждая из которых отвечает за хранение определенного типа данных. Первая и основная таблица – это users, которая содержит информацию о пользователях системы. У каждого пользователя есть свой уникальный user_id, а так же username (логин), password_hash (хешированный пароль), role_id для определения прав пользователя, и member_id что бы пользователь был связан с читателем. Так как пользователь может быть и не участником библиотеки то для member_id стоит возможность хранить NULL. Связь с таблицей roles по role_id.

Далее идёт таблица members, в которой хранится информация о членах библиотеки. У каждого члена есть свой member_id, имя (first_name) и фамилия (last_name), а так же необязательная контактная информация (contact_info), и дата начала членства (membership_start_date). Также имеется поле membership_end_date для отображения даты окончания членства, так как эта дата не всегда есть, то оно имеет возможность хранить NULL. Связь с таблицей users по member_id.

Таблица books хранит информацию о книгах, где для каждой книги есть уникальный book_id, название (title), связь с автором через author_id, связь с жанром через genre_id, год публикации (publication_year), а так же общее количество копий (total_copies) и доступное количество копий (available_copies). Связи с таблицами authors по author_id и genres по genre_id

Информация о авторах хранится в таблице authors, где для каждого автора есть свой author_id, имя (first_name), фамилия (last_name), и, по желанию, дата рождения (date_of_birth). Эта таблица нужна, чтобы избежать повторения данных об авторах.

Так же есть таблица genres, где хранятся все жанры книг, где каждому жанру соответствует свой genre_id, и название (genre_name). Эта таблица так же необходима для устранения повторений данных.

Для фиксации выданных книг и для связи между книгами и членами библиотеки я создал таблицу loans. Она содержит информацию о выдаче книги, с полями loan_id, book_id, member_id, дату выдачи loan_date, срок возврата due_date, и датой фактического возврата return_date (которая может быть NULL). Связи с таблицами books по book_id и members по member_id. Так как читатель может взять множество книг, а одна книга может быть выдана множеству читателей то эта таблица представляет связь многие ко многим.

И таблица roles, где хранятся роли пользователей, где есть role_id и название role_name. Эта таблица, как и таблица жанров, нужны для устранения повторений и связей.

Для обеспечения целостности данных используются внешние ключи. Например, author_id в таблице books ссылается на author_id в таблице authors. Для более надежной системы при удалении данных в таблицах loans и users используются ограничения ON DELETE CASCADE и ON DELETE SET NULL соответственно.

2.2. Проектирование пользовательского интерфейса

Форма аутентификации и регистрации:

Начало работы с системой всегда начинается с форм аутентификации и регистрации. Форма аутентификации – это окно для входа уже зарегистрированного пользователя. Она содержит поля для ввода логина и пароля, а так же кнопки "Войти" и "Зарегистрироваться". Так как у нас много ролей, то используется одна форма для всех пользователей, и в зависимости от роли будет открыт нужный интерфейс.

Форма регистрации предназначена для тех, кто впервые пользуется системой. Она содержит поля для ввода имени, фамилии, логина, пароля, номера телефона (по желанию) и кнопки "Зарегистрироваться" и "На главную".

Интерфейс пользователя:

Основная цель интерфейса пользователя – предоставить ему возможность просматривать информацию о себе и о взятых книгах.

Интерфейс пользователя представляет собой окно, в верхней части которого находится заголовок с ролью пользователя, далее идет информация о пользователе. Затем пользователь видит список взятых им книг, в виде таблицы. Так же имеется кнопка для выхода, которая расположена справа вверху.

Интерфейс библиотекаря:

Интерфейс библиотекаря предназначен для выполнения задач, связанных с управлением книгами и пользователями, и так же для обработки выдачи книг.

В верхней части интерфейса библиотекаря расположен заголовок "Кабинет библиотекаря" и кнопка "Выход", расположенная справа вверху. Далее идёт строка с полем для ввода member_id и кнопка "Найти читателя".

Ниже расположены область с информацией о читателе и таблица с данными о выданных книгах. Так же внизу имеется панель кнопок управления книгами и выдачей, это кнопки "Добавить книгу", "Редактировать данные", и "Удалить запись", а так же блок кнопок для управления книгами в библиотеке "Добавить книгу в библиотеку", "Редактировать книгу", и "Удалить книгу".

Интерфейс администратора:

Интерфейс администратора предназначен для управления пользователями и их правами, а так же для управления резервными копиями.

Интерфейс администратора является копией интерфейса библиотекаря, но с заголовком "Кабинет Админа". В верхней части экрана расположена кнопка "Выход", и кнопка "Привелегии аккаунтов", позволяющая перейти к управлению ролями пользователей, а так же кнопка "Удалить пользователя". В низу окна, под таблицей со списком книг расположена кнопка "Создать бэкап".

2.3. Описание основных модулей

db_utils.py (Модуль работы с базой данных):

Этот модуль, является самым важным, ведь в нём содержится весь код, необходимый для взаимодействия с базой данных PostgreSQL. В этом модуле я создал функции connect_to_db для подключения к базе данных, и execute_query для выполнения SQL-запросов. Так же в этом файле находится функционал для создания бэкапа (create_backup), и восстановления бд из бэкапа (restore_backup). Иными словами этот файл управляет работой с базой данных.

auth_form.py (Модуль аутентификации):

Этот модуль отвечает за отображение и работу окна аутентификации (то есть входа). Тут написан код для проверки логина и пароля, а так же перенаправления на другие окна. Модуль создает интерфейс для авторизации, позволяющий пользователям войти в систему. Тут же происходит обработка ввода логина и пароля, проверка на корректность, и вызов других форм в зависимости от роли пользователя.

reg_form.py (Модуль регистрации):

Этот модуль управляет процессом регистрации новых пользователей. Здесь происходит создание интерфейса, обработка ввода данных, хеширование паролей, и добавление нового пользователя в базу данных. Тут же происходит связь нового пользователя с новым участником библиотеки, если пользователь при регистрации не указал номер телефона, и так же если он указал номер телефона.

user_form.py (Модуль интерфейса пользователя):

Здесь находится код, который отвечает за отображение главного окна пользователя. Этот модуль отображает информацию о текущем пользователе,

и список выданных книг. Также здесь расположена кнопка выхода из аккаунта.

librarian_form.py (Модуль интерфейса библиотекаря):

В этом модуле реализуется окно и функции для роли библиотекаря. Это и обработка поиска пользователя по member_id, добавления и редактирования выдачи книг, а также управление каталогом книг. Этот модуль отвечает за все действия библиотекаря в системе.

admin_form.py (Модуль интерфейса администратора):

Здесь содержится код, для создания интерфейса и логики для администратора, включая функционал управления пользователями, их ролями, а так же бэкапами и восстановлением.

admin_privileges_form.py (Модуль управления правами пользователей):

В этом модуле я реализовал логику и пользовательский интерфейс для изменения ролей пользователей, так как это функция предназначена только для администратора, то я вынес это в отдельный модуль.

delete_user_form.py (Модуль удаления пользователя):

Этот модуль создан специально для удаления пользователя из системы, и так же для удаления связанных с ним данных.

add_loan_form.py (Модуль добавления выдачи):

Этот модуль предназначен для добавления новой выдачи книги. Тут происходит связь с таблицей books через ComboBox, а так же происходит добавление данных о выдаче книги.

edit_loan_form.py (Модуль редактирования выдачи):

Этот модуль предназначен для редактирования данных о выданных книгах. Он позволяет изменить дату выдачи, дату возврата.

add_book_form.py (Модуль добавления книг):

Модуль, в котором я реализовал логику и интерфейс для добавления новой книги в библиотеку.

edit_book_form.py (Модуль редактирования книг):

Этот модуль предоставляет интерфейс для редактирования информации о книгах в библиотеке.

delete_book_form.py (Модуль удаления книг):

Здесь находится логика для удаления книги из каталога библиотеки.

2.4. Реализация основных функций

Аутентификация и авторизация:

Для входа в систему я использовал форму AuthForm, где пользователи вводят свой логин и пароль. Для проверки пароля я использовал библиотеку bcrypt, что бы не хранить пароли в открытом виде, а хешировать их. При правильном вводе логина и пароля, система определяет роль пользователя (администратор, библиотекарь или обычный пользователь) и открывает соответствующий интерфейс. Это делается в файле main.py, где в зависимости от роли происходит вызов нужной функции, которая в свою очередь открывает нужную форму. Для этого я использую функцию show_after_auth_form которая открывает формы UserForm или LibrarianForm, или AdminForm в зависимости от роли.

Так же в модуле auth_form.py происходит запрос к БД, и если пользователь есть в БД то вызывается соответствующая форма.

Управление пользователями:

Для управления пользователями (добавление, изменение и удаление), а так же для их регистрации, я создал отдельные модули reg_form.py, admin_privileges_form.py, и delete_user_form.py.

В reg_form.py новый пользователь регистрируется в системе, при чем если он не указал номер телефона, то для него создается новая запись в members, а так же новый пользователь добавляется в таблицу users, с присвоением роли "пользователь".

В admin_privileges_form.py администратор может менять ролль пользователям.

В delete_user_form.py я реализовал удаление пользователя, и, так же, если пользователь имеет member_id то так же удаляется и запись из таблицы members

Управление книгами:

Для управления книгами я создал отдельные модули, такие как add_book_form.py, edit_book_form.py и delete_book_form.py.

В add_book_form.py содержится логика для добавления информации о новой книге в базу данных, а так же добавления новых авторов если их еще нет.

В edit_book_form.py реализовано редактирование данных о книге, включая автора, название, и год издания.

В delete_book_form.py реализовано удаление книг из каталога.

Управление выдачей книг:

В модуле librarian_form.py есть функции для вызова форм add_loan_form.py и edit_loan_form.py.

В add_loan_form.py создается новый запрос в базу данных с информацией о выданной книге, и id члена библиотеки.

В edit_loan_form.py реализовано редактирование информации о выдаче.

Создание и восстановление резервных копий:

Для резервного копирования и восстановления базы данных я использовал утилиты pg_dump и psql. Для этого я написал в db_utils.py функции create_backup и restore_backup, которые можно вызывать из интерфейса администратора.

Функция create_backup создает копию базы данных в формате SQL, и отправляет её на FTP сервер.

Функция restore_backup восстанавливает базу данных с помощью sql файла.

SQL-запросы:

Почти все взаимодействия с БД происходит через SQL запросы.

Использование subprocess:

В функциях create_backup и restore_backup используется модуль subprocess, что позволяет вызывать из кода программы сторонние утилиты, такие как pg_dump.exe и psql.exe. Это помогло мне в реализации бэкапа.

Заключение

В рамках курсовой работы была разработана десктопная система управления библиотекой, которая автоматизирует основные процессы: аутентификация, управление пользователями, книгами, и их выдачей, а так же создание бэкапов.

В процессе разработки были использованы Python, Tkinter, PostgreSQL, а так же библиотеки psycopg2, bcrypt, ftplib, paramiko.

Система предоставляет функционал для разных ролей: администратора, библиотекаря и обычного пользователя. Администратор может управлять пользователями и резервными копиями. Библиотекарь - фондом и выдачей книг, а пользователь может просматривать информацию о своих книгах.

Список источников и литературы

1. PostgreSQL Documentation:

Официальная документация по PostgreSQL [Электронный ресурс] Режим доступа:

https://www.postgresql.org/docs/

Руководство по утилите pg_dump[Электронный ресурс] Режим доступа: https://www.postgresql.org/docs/current/app-pgdump.html

Руководство по утилите psql [Электронный ресурс] Режим доступа: https://www.postgresql.org/docs/current/app-psql.html

2. Python Documentation[Электронный ресурс] Режим доступа: https://docs.python.org/3/
3. Tkinter Documentation[Электронный ресурс] Режим доступа: https://docs.python.org/3/library/tkinter.html
4. psycopg2 Documentation[Электронный ресурс] Режим доступа: https://www.psycopg.org/docs/
5. bcrypt Documentation[Электронный ресурс] Режим доступа: https://pypi.org/project/bcrypt/
6. Paramiko Documentation[Электронный ресурс] Режим доступа: http://docs.paramiko.org/en/stable/
7. ftplib Documentation[Электронный ресурс] Режим доступа: https://docs.python.org/3/library/ftplib.html

Приложения

Github проекта: https://github.com/NTK-Hub/library_management_system