Тема проекта

Система управления медицинскими записями

Описание:

Разработка системы для управления медицинскими записями пациентов в клиниках. Врачи смогут создавать и просматривать медицинские записи, назначать лечение, а пациенты смогут отслеживать свои визиты и назначения.

### Функциональные требования

1. \*\*Регистрация и авторизация пользователей:\*\*

- Регистрация новых пользователей (пациентов, врачей, администраторов).

- Авторизация зарегистрированных пользователей в системе.

- Восстановление пароля.

2. \*\*Управление пользователями (CRUD):\*\*

- Создание, чтение, обновление и удаление записей пользователей.

- Управление профилями пациентов и врачей.

- Назначение ролей пользователям.

3. \*\*Система ролей:\*\*

- Роли пользователей (пациент, врач, администратор).

- Ограничение доступа к определённым функциям на основе роли.

4. \*\*Управление медицинскими записями:\*\*

- Создание и редактирование медицинских записей врачами.

- Просмотр медицинских записей пациента врачом и пациентом.

5. \*\*Назначение и отслеживание лечения:\*\*

- Врачи могут назначать лечение и отслеживать его выполнение.

- Пациенты могут видеть свои назначения и статус их выполнения.

6. \*\*Журналирование действий пользователя:\*\*

- Ведение логов действий пользователей (создание, редактирование, удаление записей, авторизация).

7. \*\*Управление приемами:\*\*

- Запись пациентов на прием к врачу.

- Управление расписанием врачей.

- Отслеживание статуса приема (назначен, завершен, отменен).

8. \*\*Управление лабораторными анализами:\*\*

- Назначение врачом лабораторных анализов.

- Хранение и просмотр результатов анализов.

9. \*\*Управление рецептами:\*\*

- Врачи могут выписывать рецепты пациентам.

- Пациенты могут просматривать свои рецепты и следовать инструкциям.

### Сущности базы данных

В системе управления медицинскими записями будет 10 сущностей, которые покрывают все основные аспекты функциональных требований.

1. Пользователь (User)

- \*\*Поля:\*\*

- `user\_id` (INT, PRIMARY KEY, AUTO\_INCREMENT): уникальный идентификатор пользователя.

- `username` (VARCHAR(100), UNIQUE, NOT NULL): имя пользователя для авторизации.

- `password\_hash` (VARCHAR(255), NOT NULL): хэш пароля пользователя.

- `email` (VARCHAR(255), UNIQUE, NOT NULL): адрес электронной почты пользователя.

- `created\_at` (TIMESTAMP, DEFAULT CURRENT\_TIMESTAMP): дата и время регистрации.

- `role\_id` (INT, FOREIGN KEY, NOT NULL): ссылка на роль пользователя.

- \*\*Связи:\*\*

- Один ко многим с сущностью `Role` (у каждого пользователя одна роль, но одна роль может быть у многих пользователей).

- Один к одному с `Patient` и `Doctor` (каждый пользователь связан либо с пациентом, либо с врачом).

2. \*\*Роль (Role)\*\*

- \*\*Поля:\*\*

- `role\_id` (INT, PRIMARY KEY, AUTO\_INCREMENT): уникальный идентификатор роли.

- `role\_name` (VARCHAR(50), UNIQUE, NOT NULL): название роли (например, пациент, врач, администратор).

- \*\*Связи:\*\*

- Один ко многим с сущностью `User`.

3. \*\*Пациент (Patient)\*\*

- \*\*Поля:\*\*

- `patient\_id` (INT, PRIMARY KEY, AUTO\_INCREMENT): уникальный идентификатор пациента.

- `user\_id` (INT, FOREIGN KEY, NOT NULL): ссылка на пользователя.

- `first\_name` (VARCHAR(100), NOT NULL): имя пациента.

- `last\_name` (VARCHAR(100), NOT NULL): фамилия пациента.

- `date\_of\_birth` (DATE, NOT NULL): дата рождения пациента.

- `gender` (ENUM('Male', 'Female', 'Other'), NOT NULL): пол пациента.

- `insurance\_id` (INT, FOREIGN KEY, NULL): ссылка на медицинскую страховку.

- \*\*Связи:\*\*

- Один к одному с `User`.

- Один ко многим с `MedicalRecord`, `Appointment`, `Prescription`, `LabTest`.

- Один ко многим с `Insurance` (у пациента может быть несколько страховок, но для упрощения модели предполагается, что текущая активная страховка одна).

4. \*\*Врач (Doctor)\*\*

- \*\*Поля:\*\*

- `doctor\_id` (INT, PRIMARY KEY, AUTO\_INCREMENT): уникальный идентификатор врача.

- `user\_id` (INT, FOREIGN KEY, NOT NULL): ссылка на пользователя.

- `first\_name` (VARCHAR(100), NOT NULL): имя врача.

- `last\_name` (VARCHAR(100), NOT NULL): фамилия врача.

- `specialization` (VARCHAR(255), NOT NULL): специализация врача.

- \*\*Связи:\*\*

- Один к одному с `User`.

- Один ко многим с `MedicalRecord`, `Appointment`, `Prescription`, `LabTest`.

5. \*\*Медицинская запись (MedicalRecord)\*\*

- \*\*Поля:\*\*

- `record\_id` (INT, PRIMARY KEY, AUTO\_INCREMENT): уникальный идентификатор медицинской записи.

- `patient\_id` (INT, FOREIGN KEY, NOT NULL): ссылка на пациента.

- `doctor\_id` (INT, FOREIGN KEY, NOT NULL): ссылка на врача.

- `diagnosis` (TEXT, NOT NULL): диагноз.

- `treatment\_plan` (TEXT, NOT NULL): план лечения.

- `created\_at` (TIMESTAMP, DEFAULT CURRENT\_TIMESTAMP): дата и время создания записи.

- \*\*Связи:\*\*

- Один ко многим с `Patient`.

- Один ко многим с `Doctor`.

6. \*\*Назначение лечения (Treatment)\*\*

- \*\*Поля:\*\*

- `treatment\_id` (INT, PRIMARY KEY, AUTO\_INCREMENT): уникальный идентификатор назначения.

- `record\_id` (INT, FOREIGN KEY, NOT NULL): ссылка на медицинскую запись.

- `treatment\_description` (TEXT, NOT NULL): описание назначенного лечения.

- `start\_date` (DATE, NOT NULL): дата начала лечения.

- `end\_date` (DATE, NULL): дата окончания лечения (может быть пустой, если лечение продолжается).

- `status` (ENUM('Pending', 'Ongoing', 'Completed', 'Cancelled'), NOT NULL): статус лечения.

- \*\*Связи:\*\*

- Один ко многим с `MedicalRecord`.

7. \*\*Журнал (Log)\*\*

- \*\*Поля:\*\*

- `log\_id` (INT, PRIMARY KEY, AUTO\_INCREMENT): уникальный идентификатор записи журнала.

- `user\_id` (INT, FOREIGN KEY, NOT NULL): ссылка на пользователя.

- `action` (VARCHAR(255), NOT NULL): описание действия (например, "Создание записи", "Обновление профиля").

- `timestamp` (TIMESTAMP, DEFAULT CURRENT\_TIMESTAMP): дата и время действия.

- \*\*Связи:\*\*

- Один ко многим с `User`.

8. \*\*Прием (Appointment)\*\*

- \*\*Поля:\*\*

- `appointment\_id` (INT, PRIMARY KEY, AUTO\_INCREMENT): уникальный идентификатор приема.

- `patient\_id` (INT, FOREIGN KEY, NOT NULL): ссылка на пациента.

- `doctor\_id` (INT, FOREIGN KEY, NOT NULL): ссылка на врача.

- `appointment\_date` (DATETIME, NOT NULL): дата и время приема.

- `status` (ENUM('Scheduled', 'Completed', 'Cancelled'), NOT NULL): статус приема.

- \*\*Связи:\*\*

- Один ко многим с `Patient`.

- Один ко многим с `Doctor`.

9. \*\*Лабораторный анализ (LabTest)\*\*

- \*\*Поля:\*\*

- `lab\_test\_id` (INT, PRIMARY KEY, AUTO\_INCREMENT): уникальный идентификатор анализа.

- `patient\_id` (INT, FOREIGN KEY, NOT NULL): ссылка на пациента.

- `doctor\_id` (INT, FOREIGN KEY, NOT NULL): ссылка на врача.

- `test\_type` (VARCHAR(255), NOT NULL): тип анализа (например, кровь, моча и т.д.).

- `result` (TEXT, NULL): результат анализа.

- `test\_date` (DATE, NOT NULL): дата проведения анализа.

- `status` (ENUM('Pending', 'Completed', 'Cancelled'), NOT NULL): статус анализа.

- \*\*Связи:\*\*

- Один ко многим с `Patient`.

- Один ко многим с `Doctor`.

10. \*\*Рецепт (Prescription)\*\*

- \*\*Поля:\*\*

- `prescription\_id` (INT, PRIMARY KEY, AUTO\_INCREMENT): уникальный идентификатор рецепта.

- `patient\_id` (INT, FOREIGN KEY, NOT NULL): ссылка на пациента.

- `doctor\_id` (INT, FOREIGN KEY, NOT NULL): ссылка на врача.

- `medication\_name` (VARCHAR(255), NOT NULL): название препарата.

- `dosage`

(VARCHAR(255), NOT NULL): дозировка.

- `instructions` (TEXT, NOT NULL): инструкции по приему.

- `issue\_date` (DATE, NOT NULL): дата выдачи рецепта.

- \*\*Связи:\*\*

- Один ко многим с `Patient`.

- Один ко многим с `Doctor`.

### Схематичное изображение не нормализованной схемы БД

Не нормализованная инфологическая модель базы данных может быть создана с помощью таких инструментов, как draw.io, Miro.com, Microsoft Word, PowerPoint, или любой другой графический редактор. Модель должна включать все перечисленные сущности и связи между ними, демонстрируя все виды отношений (1:1, 1:M, M:M).

### Описание сущностей

Каждая сущность была подробно описана выше, включая ее поля, типы данных, ограничения и связи с другими сущностями. Эти описания могут быть использованы для создания базы данных в любой реляционной СУБД, такой как MySQL, PostgreSQL или SQL Server.

### Теоретические знания

Для выполнения лабораторной работы потребуются знания в следующих областях:

- Основы реляционных баз данных.

- Типы связей между таблицами (1:1, 1:M, M:M).

- Проектирование и нормализация базы данных.

- Основы SQL: создание таблиц, определение связей, использование внешних ключей и индексов.

- Основы безопасности данных, включая хэширование паролей.

- Принципы журналирования и ведения логов.

### Пример README.md файла:

```markdown

# Система управления медицинскими записями

## Описание проекта

Система управления медицинскими записями предназначена для клиник и позволяет врачам вести медицинские записи пациентов, назначать лечение, а пациентам отслеживать свои визиты и назначения.

## Автор

ФИО: [Ваше ФИО]

Группа: [Номер группы]

## Функциональные требования

1. Регистрация и авторизация пользователей.

2. Управление пользователями (CRUD).

3. Система ролей.

4. Управление медицинскими записями.

5. Назначение и отслеживание лечения.

6. Журналирование действий пользователя.

7. Управление приемами.

8. Управление лабораторными анализами.

9. Управление рецептами.

```

Этот пример README.md файла должен быть дополнен и адаптирован под конкретные требования вашей лабораторной работы.

### Сущность: Администратор (Admin)

Сущность "Администратор" представляет пользователя, который управляет системой, настраивает параметры, контролирует действия других пользователей и выполняет административные функции.

### Описание сущности

- \*\*Поля:\*\*

- `admin\_id` (INT, PRIMARY KEY, AUTO\_INCREMENT): уникальный идентификатор администратора.

- `user\_id` (INT, FOREIGN KEY, NOT NULL): ссылка на пользователя.

- `first\_name` (VARCHAR(100), NOT NULL): имя администратора.

- `last\_name` (VARCHAR(100), NOT NULL): фамилия администратора.

- `created\_at` (TIMESTAMP, DEFAULT CURRENT\_TIMESTAMP): дата и время создания записи.

- `updated\_at` (TIMESTAMP, DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP): дата и время последнего обновления записи.

- \*\*Связи:\*\*

- Один к одному с `User` (каждый администратор связан с одним пользователем).

### Обновленный список сущностей базы данных:

1. \*\*Пользователь (User)\*\*

2. \*\*Роль (Role)\*\*

3. \*\*Пациент (Patient)\*\*

4. \*\*Врач (Doctor)\*\*

5. \*\*Администратор (Admin)\*\*

6. \*\*Медицинская запись (MedicalRecord)\*\*

7. \*\*Назначение лечения (Treatment)\*\*

8. \*\*Журнал (Log)\*\*

9. \*\*Прием (Appointment)\*\*

10. \*\*Лабораторный анализ (LabTest)\*\*

11. \*\*Рецепт (Prescription)\*\*

12. \*\*Медицинское учреждение (MedicalFacility)\*\*

### Виды связей:

Добавление сущности `Admin` не изменяет уже существующие связи, но вводит еще одну связь \*\*Один к одному (1:1)\*\* между `User` и `Admin`.

### Схематичное изображение

В схему теперь будет добавлена сущность `Admin`, связанная с `User` через связь один к одному.

### README.md

Добавьте описание новой сущности в README.md:

```markdown

## Описание сущностей

### Администратор (Admin)

Администратор управляет системой, выполняет административные функции и контролирует действия других пользователей. Связан с сущностью `User` через связь один к одному.

```

Теперь система управления медицинскими записями включает возможность администрирования, что делает ее более функциональной и управляемой.

Добавим еще одну сущность — \*\*Медицинское учреждение (MedicalFacility)\*\*, которая будет хранить информацию о клиниках или больницах, где работают врачи и обслуживаются пациенты.

### Сущность: Медицинское учреждение (MedicalFacility)

- \*\*Поля:\*\*

- `facility\_id` (INT, PRIMARY KEY, AUTO\_INCREMENT): уникальный идентификатор медицинского учреждения.

- `facility\_name` (VARCHAR(255), NOT NULL): название медицинского учреждения.

- `address` (VARCHAR(255), NOT NULL): адрес учреждения.

- `phone\_number` (VARCHAR(20), NULL): контактный номер телефона.

- `email` (VARCHAR(255), NULL): контактный адрес электронной почты.

- `website` (VARCHAR(255), NULL): веб-сайт медицинского учреждения.

- `created\_at` (TIMESTAMP, DEFAULT CURRENT\_TIMESTAMP): дата и время добавления записи.

- \*\*Связи:\*\*

- Один ко многим с `Doctor`: одно медицинское учреждение может иметь множество врачей.

- Один ко многим с `Patient`: один пациент может быть прикреплен к одному медицинскому учреждению (например, к поликлинике по месту жительства).

- Один ко многим с `Appointment`: приемы пациентов могут быть привязаны к определенному медицинскому учреждению.

- Один ко многим с `LabTest`: лабораторные анализы могут проводиться в определенном учреждении.

### Обновленный список сущностей базы данных:

1. \*\*Пользователь (User)\*\*

2. \*\*Роль (Role)\*\*

3. \*\*Пациент (Patient)\*\*

4. \*\*Врач (Doctor)\*\*

5. \*\*Медицинская запись (MedicalRecord)\*\*

6. \*\*Назначение лечения (Treatment)\*\*

7. \*\*Журнал (Log)\*\*

8. \*\*Прием (Appointment)\*\*

9. \*\*Лабораторный анализ (LabTest)\*\*

10. \*\*Рецепт (Prescription)\*\*

11. \*\*Медицинское учреждение (MedicalFacility)\*\*

### Виды связей:

С добавлением сущности `MedicalFacility` все виды связей остаются покрытыми:

- \*\*Один к одному (1:1):\*\* `User` — `Patient` и `User` — `Doctor`.

- \*\*Один ко многим (1:M):\*\* `Role` — `User`, `Patient` — `MedicalRecord`, `Doctor` — `MedicalRecord`, `MedicalFacility` — `Doctor`, и другие.

- \*\*Многие ко многим (M:M):\*\* `Patient` — `Doctor` через `Appointment`.

### Схематичное изображение

Теперь в схеме будет дополнительная сущность `MedicalFacility`, которая связывается с врачами, пациентами, приемами и лабораторными анализами.

### README.md

Добавьте описание новой сущности в README.md:

```markdown

## Описание сущностей

### Медицинское учреждение (MedicalFacility)

Медицинское учреждение хранит информацию о клиниках или больницах, где работают врачи и обслуживаются пациенты. Связана с врачами, пациентами, приемами и лабораторными анализами.

```

Таким образом, добавление этой сущности делает систему более комплексной и лучше отражает реальную структуру взаимодействия в медицинских учреждениях.