МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ  
Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ   
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра информационных систем и технологий

Специальность 1-40 01 01 «Программное обеспечение информационных технологий»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА КУРСОВОГО ПРОЕКТА**

по дисциплине «Базы данных»

Тема: «Реализация база данных поликлиники с применением технологии шифрование и маскирование»

**Исполнитель**

студент 3 курса 5 группы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д. Ю. Окулич

подпись, дата

**Руководитель**

ассистент \_\_\_\_\_\_\_\_\_\_\_\_ О. А. Нистюк

(должность, уч. звание) (подпись, дата)

Допущен(а) к защите \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

дата, подпись

Курсовой проект защищен с оценкой

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_ О. А. Нистюк

подпись дата инициалы и фамилия

Минск 2023

**Содержание**

[Введение 4](#_Toc149334900)

[1 Постановка задачи 5](#_Toc149334901)

[1.1 Аналитический обзор аналогов и литературных решений 5](#_Toc149334902)

[2.1 Анализ и разработка функциональных требований 7](#_Toc149334903)

[2 Проектирование базы данных 9](#_Toc149334904)

[3 Разработка необходимых объектов 11](#_Toc149334905)

[3.1 Таблицы 11](#_Toc149334906)



# Введение

Цель данной работы заключается в создании реляционной базы данных для веб-приложения поликлиники, которое позволит поциентам удобно записываться на приемы, а врачам удобно анализировать и вести учёт болезней, диагнозов, сиптомов и лекарств.

База данных - это организованное собрание данных, которое обычно хранится в электронном виде в компьютерной системе. БД используются для хранения, организации и управления большим объемом структурированных и неструктурированных данных. Реляционная база данных является наиболее распространенной формой организации данных, в которой данные представлены в виде таблиц, состоящих из строк и столбцов, где каждый столбец представляет атрибут, а каждая строка представляет кортеж или запись. В данной работе для управления базой данных была выбрана СУБД Oracle, поскольку эта система обладает большим функционалом, что позволяет обеспечить эффективное хранение, обработку и управление данными.

Также необходимо разработать приложение для демонстрации функциональности базы данных и взаимодействия с ней. Приложение было реализовано с использованием фреймворка Spring boot на языке Java и HTML c JavaScript.

Для гарантированной безопасности пользователей приложения в курсовой работе применяется метод шифрования паролей перед их сохранением в базу данных. Так же будет использоваться маскирование некоторых данных что бы пользователю было доступна часть информации.

Основные требования к приложению:

* Реализация ролей администратора, врача и пациента;
* Открытие, закрытие диагноза;
* Регистрация и легирование;
* Оформление приёма;
* взаимодействие с базой данных при помощи хранимых процедур и функций.

Содержание данной пояснительной записки отражает этапы выполнения курсового проекта.

1. Постановка задачи

Задача проекта: разработать архитектуру базы данных, создать процедуры и функции, взаимодействие с которыми будет понятно любому пользователю.

В соответствии с заданием курсового проекта следует не только создать базу данных, но и разработать программное средство, которое должно в полной мере демонстрировать возможности базы данных.

Для того, чтобы сформировать окончательные требования к проектируемому программному средству сначала рассмотрим прототипы из той же области.

1. Аналитический обзор аналогов и литературных решений

Немаловажным этапом в разработке программного продукта является аналитический обзор прототипов и литературных источников.

На сегодняшний день аналогичных приложений или web-сервисов для контроля базы данных поликлиники в открытом доступе не много. Поэтому было рассмотрено всего пара аналогов.

Мобильное приложение «BestDoctor» ­­– приложение, предназначенное для записи к врачу, а также помогаем в выборе специалиста и предлагает оптимальные решения для проблем, связанных с здоровьем.

Интерфейса приведён на рисунке 1.1.

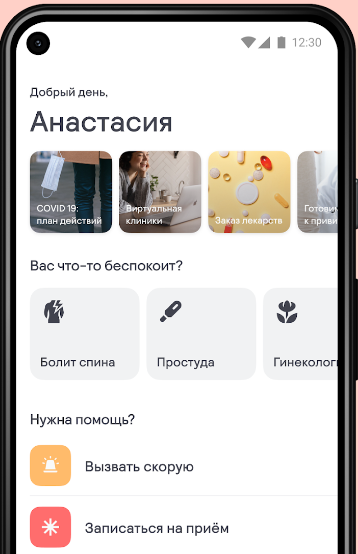


Рисунок 1.1 – Интерфейс приложения «BestDoctor»

Проанализировав «BestDoctor», можно выделить основные минусы и плюсы данного программного средства.

Основные плюсы:

* удобный интерфейс;
* бесплатный;
* имеется чат для общения с врачом;
* имеется карта поликлиник;

Основные минусы:

* навязчивая реклама;
* мало функционала для врачей;

Мобильное приложение «справочник врача» – приложение является мобильным справочником для врачей с встроенным калькулятором анализов.

Интерфейс приведён на рисунке 1.2.

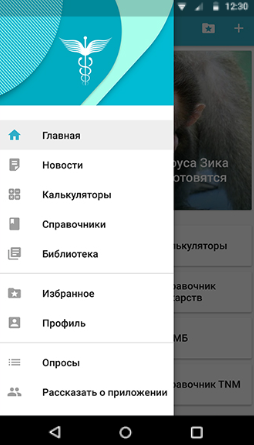


Рисунок 1.2 интерфейс приложения «Справочник врача»

Проанализировав «Справочник врача», можно выделить основные минусы и плюсы данного программного средства.

Основные плюсы:

* есть калькулятор анализов;
* большой объем справочников;

Основные минусы:

* не удобный интерфейс;
* нет функционала для пациентов;

Таким образом был выполнен анализ необходимого функционала, предоставленный аналогами разрабатываемого приложения.

1. Анализ и разработка функциональных требований

Анализ требований – процесс сбора требований к программному обеспечению, их систематизации, документирования, анализа, выявления противоречий, неполноты, разрешения конфликтов в процессе разработки программного обеспечения.

Цель анализа требований в проектах – получить максимум информации о заказчике и специфике его задач, уточнить рамки проекта, оценить возможные риски.

На этом этапе происходит идентификация принципиальных требований методологического и технологического характера, формулируются цели и задачи проекта.

Определение и описание требований – шаги, которые во многом определяют успех всего проекта, поскольку именно они влияют на все остальные этапы.

Различают три уровня требований к проекту:

* бизнес-требования;
* пользовательские требования;
* функциональные требования;

Бизнес-требования содержат высокоуровневые цели организации или заказчиков системы. Как правило, их высказывают те, кто финансируют проект, покупатели системы, менеджер реальных пользователей, отдел маркетинга.

Курсовой проект не подразумевает наличие заказчика, который мог бы выдвинуть бизнес-требования, поэтому в качестве таких высокоуровневых требований будут рассматривать общие требования к разрабатываемому курсовому проекту.

К их этим требованиям относятся:

* использование архитектурных шаблонов проектирования;
* использование системы управления базами данных (СУБД);
* доступ к данным должен осуществляться только через соответствующие процедуры.

Следующими требованиями являются требования пользователей приложения, а именно администратора, врача и пациента.

Данные требования описывают цели и задачи, которые пользователям позволит решить система. Таким образом, в пользовательских требованиях указано, что клиенты смогут делать с помощью системы.

Пользователь данного программного решения должен иметь возможности, соответствующие его роли.

Возможности администратора:

* управление базой данных болезней;
* управление базой данных симптомов;
* управление базой данных лекарств;
* управление базой данных посещений;

Возможности врача:

* подтверждение/отмена посещения;
* создание/редактирование диагноза;
* изменение состояния диагноза;
* просмотр активных посещений;
* просмотр активных диагнозов;

Возможности пациента:

* запись на посещение;
* отмена посещения;
* просмотр своих диагнозов;
* просмотр своих посещений;

После проведения анализа были выявлены следующие функциональные требования программного средства:

* вся информация должна храниться в базе данных;
* приложение должно производить валидацию вводимых пользователем данных;
* приложение должно корректным образом обрабатывать возникающие исключительные ситуации: отображать понятное для пользователя сообщение о возникшей ошибке;
* приложение должно предоставлять пользователям возможность создания нового аккаунта в виде регистрационной формы;
* приложение должно предоставлять возможность пользователям проходить аутентификацию и входить в систему под соответствующим введенным данным пользовательским именем;

Таким образом, был проведен тщательный анализ требований к программному средству, который позволил разработать список функциональных требований. Разработка данной программной системы должна проводиться в соответствии с сформированными списком.

1. Проектирование базы данных

Проектирование баз данных представляет собой процесс создания структуры базы данных и определения необходимых ограничений целостности.

Основные задачи проектирования базы данных:

* обеспечение хранения в БД всей необходимой информации;
* обеспечение возможности получения данных по всем необходимым запросам;
* сокращение избыточности и дублирования данных;
* обеспечение целостности базы данных.

Проектирование базы данных проходит через два основных этапа: концептуальное и логическое проектирование.

Концептуальное проектирование представляет собой создание семантической модели предметной области, то есть информационной модели на высоком уровне абстракции.

Концептуально на данном этапе было выделено 6 сущностей:

* пользователь;
* посещение;
* болезнь;
* симптом;
* лекарство;
* диагноз;

Так же были определены необходимые связи. Например, между сущностями «пользователь» и «посещение» установлена связь один-ко-многим. Так же были выделены атрибуты для сущностей. Например, для сущности «пользователь» были выделены атрибуты такие, как «имя», «фамилия», «отчество», «дата рождения», «пароль», «мобильный номер», «роль».

Логическое проектирование – создание схемы базы данных на основе конкретной модели данных, например, реляционной модели данных.

Для реляционной модели данных логическая модель – набор схем отношений, обычно с указанием первичных ключей, а также «связей» между отношениями, представляющих собой внешние ключи.

На этапе логического проектирования учитывается специфика конкретной модели данных, но может не учитываться специфика конкретной СУБД.

Логическая модель базы со структурой связей представлена на рисунке 2.1.

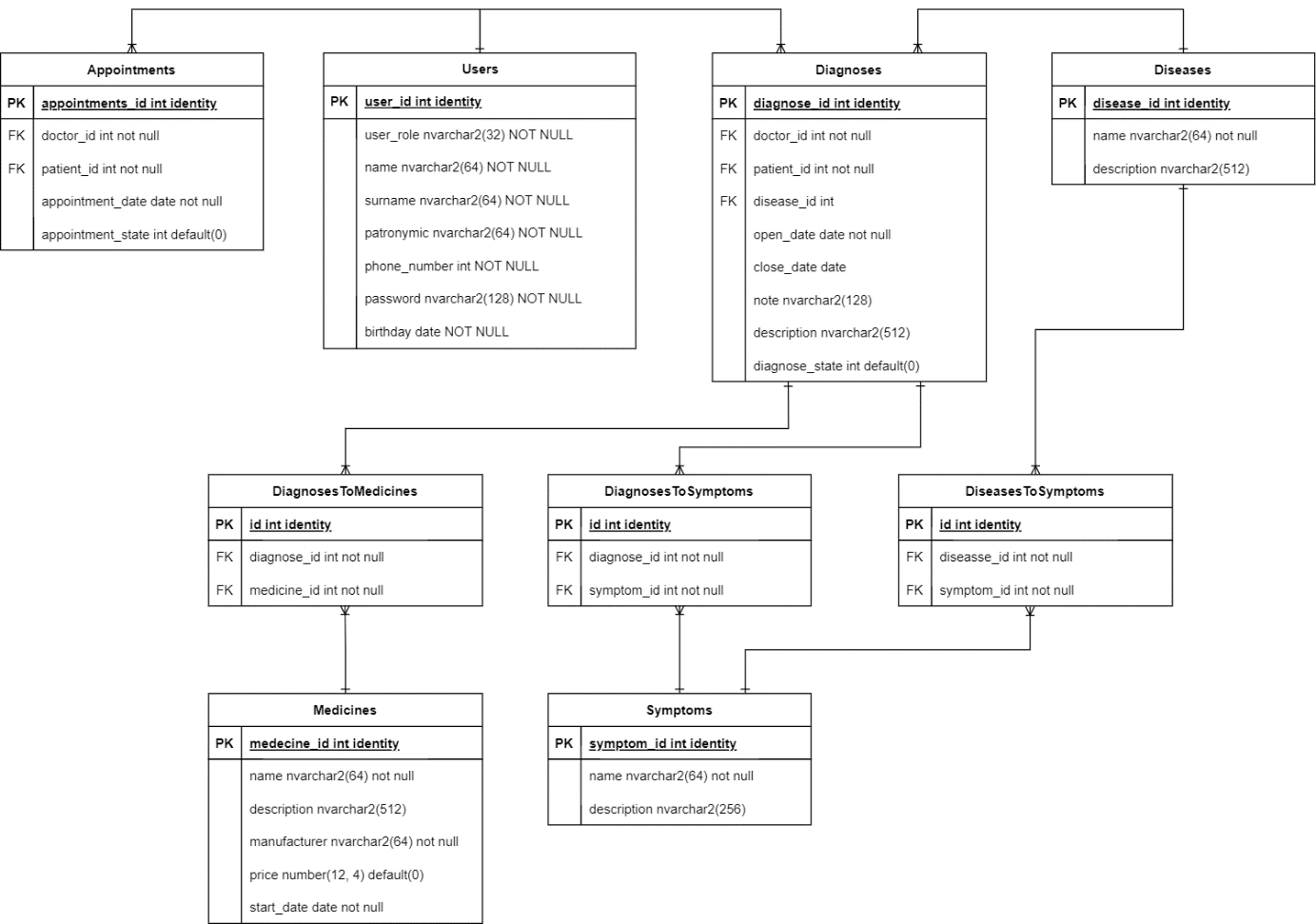


Рисунок 2.1 логическая модель базы данных

Всего в базе данных содержится 9 таблиц.

Таблица Users предназначена для хранения данных зарегистрированных пользователей.

Таблица Appointments предназначена для хранения данных посещений.

Таблица Medicines предназначена для хранения данных лекарств.

Таблица Symptoms предназначена для хранения данных симптомов.

Таблица Diseases предназначена для хранения данных болезней.

Таблица Diagnoses предназначена для хранения данных диагнозов.

Таблица DiagnosesToMedicines предназначена для создания связи многие-ко-многим между таблицами Diagnoses и Medicines.

Таблица DiagnosesToSymptoms предназначена для создания связи многие-ко-многим между таблицами Diagnoses и Symptoms.

Таблица DiseasesToSymptoms предназначена для создания связи многие-ко-многим между таблицами Diseases и Symptoms.

1. Разработка необходимых объектов
2. Таблицы

Таблица – это совокупность связанных данных, хранящихся в структурированном виде в базе данных. Она состоит из столбцов и строк.

Столбцы таблицы называют полями; каждое поле характеризуется своим именем (названием соответствующего свойства) и типом данных, отражающих значения данного свойства. Каждое поле обладает определенным набором свойств (размер, формат и др.).

Поле базы данных – это столбец таблицы, включающий в себя значения определенного свойства.

В каждой таблице должно быть, по крайней мере, одно ключевое поле, содержимое которого уникально для любой записи в этой таблице.

Значения ключевого поля однозначно определяют каждую запись в таблице.

Для реализации базы данных «Реализация база данных поликлиники с применением технологии шифрование и маскирование» было разработано 8 таблиц: Users, Appointments, Medicines, Symptoms, Diseases, DiagnosesToMedicines, DiagnosesToSymptoms, DiseasesToSymptoms, Diagnoses. Они будут реализованы в СУБД Oracle DataBase 19c.

Таблица Users представляет список всех зарегистрированных пользователей (Таблица 3.1).

Таблица 3.1 – Столбцы таблицы Users

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| user\_id | Идентификатор пользователя | int |
| user\_role | Роль пользователя | nvarchar2 |
| name | Имя пользователя | nvarchar2 |
| surname | Фамилия пользователя | nvarchar2 |
| patronymic | Отчество пользователя | nvarchar2 |
| phone\_number | Телефонный номер пользователя | int |
| password | Пароль пользователя | nvarchar2 |
| birthday | Дата рождения пользователя | date |

Таблица Appointments представляет список всех посещений (Таблица 3.2).

Таблица 3.2 – Столбцы таблицы Appointments

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| appointments\_id | Идентификатор посещения | int |
| doctor\_id | Идентификатор врача | int |
| patient\_id | Идентификатор пациента | int |
| appointments\_date | Дата посещения | date |
| appointments\_state | Состояние посещения | int |

Таблица Diseases представляет список всех болезней (Таблица 3.3).

Таблица 3.3 – Столбцы таблицы Diseases

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| disease\_id | Идентификатор болезни | int |
| name | Название болезни | nvarchar2 |
| description | Описание болезни | nvarchar2 |

Таблица Diagnoses представляет список всех диагнозов (Таблица 3.4).

Таблица 3.4 – Столбцы таблицы Users

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| diagnose\_id | Идентификатор диагноза | int |
| doctor\_id | Идентификатор врача | int |
| patient\_id | Идентификатор пациента | int |
| disease\_id | Идентификатор болезни | int |
| open\_date | Дата открытия диагноза | date |
| close\_date | Дата закрытия диагноза | date |
| note | Заметка к диагнозу | nvarchar2 |
| description | Описание диагноза | nvarchar2 |
| diagnose\_state | Состояние диагноза | int |

Таблица Medicines представляет список всех лекарств (Таблица 3.5).

Таблица 3.5 – Столбцы таблицы Medicines

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| medicine\_id | Идентификатор лекарства | int |
| name | Название лекарства | nvarchar2 |
| description | Описание лекарства | nvarchar2 |
| manufacturer | Название производителя | nvarchar2 |
| price | Цена лекарства | number (12, 4) |
| start\_date | Дата начала производства | date |

Таблица Symptoms представляет список всех симптомов (Таблица 3.6).

Таблица 3.6 – Столбцы таблицы Symptoms

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| symptom\_id | Идентификатор симптома | int |
| name | Название симптома | nvarchar2 |
| description | Описание симптома | nvarchar2 |

Таблица DiagnosesToMedicines представляет список связей между таблицами Diagnoses и Medicines (Многие-ко-многим) (Таблица 3.7).

Таблица 3.7 – Столбцы таблицы DiagnosesToMedicines

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| id | Идентификатор связи | int |
| diagnose\_id | Идентификатор диагноза | int |
| medicine\_id | Идентификатор лекарств | int |

Таблица DiagnosesToSymptoms представляет список связей между таблицами Diagnoses и Symptoms (Многие-ко-многим) (Таблица 3.8).

Таблица 3.8 – Столбцы таблицы DiagnosesToSymptoms

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| id | Идентификатор связи | int |
| diagnose\_id | Идентификатор диагноза | int |
| symptoms\_id | Идентификатор симптома | int |

Таблица DiseasesToSymptoms представляет список связей между таблицами Diagnoses и Medicines (Многие-ко-многим) (Таблица 3.8).

Таблица 3.8 – Столбцы таблицы DiseasesToSymptoms

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| id | Идентификатор связи | int |
| diseases\_id | Идентификатор болезни | int |
| medicine\_id | Идентификатор лекарств | int |

Скрипты создания таблиц и наложение ограничений целостности на них представлены в приложении А данной записки.

Приложение А

create table Users (  
 user\_id int generated always as identity (start with 1 increment by 1),  
 primary key (user\_id),  
  
 user\_role nvarchar2(32) not null,  
 name nvarchar2(64) not null,  
 surname nvarchar2(64) not null,  
 patronymic nvarchar2(64) not null,  
 password nvarchar2(128) not null,  
 birthday date not null  
) tablespace MEDKIT\_TS;  
  
create table Appointments (  
 appointment\_id int generated always as identity (start with 1 increment by 1),  
 primary key (appointment\_id),  
  
 doctor\_id int not null,  
 foreign key (doctor\_id) references Users(user\_id),  
  
 patient\_id int not null,  
 foreign key (patient\_id) references Users(user\_id),  
  
 appointment\_date date not null,  
 appointment\_state int default(0)  
) tablespace MEDKIT\_TS;

create table Diseases (  
 disease\_id int generated always as identity (start with 1 increment by 1),  
 primary key (disease\_id),  
  
 name nvarchar2(64) not null,  
 description nvarchar2(512)  
) tablespace MEDKIT\_TS;

create table Diagnoses (  
 diagnose\_id int generated always as identity (start with 1 increment by 1),  
 primary key (diagnose\_id),  
  
 doctor\_id int not null,  
 foreign key (doctor\_id) references Users(user\_id),  
  
 patient\_id int not null,  
 foreign key (patient\_id) references Users(user\_id),  
  
 disease\_id int,  
 foreign key (disease\_id) references Diseases(disease\_id),  
  
 open\_date date not null,  
 close\_date date,  
 note nvarchar2(128),  
 description nvarchar2(512),  
 diagnose\_state int default(0)  
) tablespace MEDKIT\_TS;

create table Symptoms (  
 symptom\_id int generated always as identity (start with 1 increment by 1),  
 primary key (symptom\_id),  
  
 name nvarchar2(64) not null,  
 description nvarchar2(256)  
) tablespace MEDKIT\_TS;  
  
create table Medicines (  
 medicine\_id int generated always as identity (start with 1 increment by 1),  
 primary key (medicine\_id),  
  
 name nvarchar2(64) not null,  
 description nvarchar2(512),  
 manufacturer nvarchar2(64) not null,  
 price number(12, 4) default(0),  
 start\_date date not null  
) tablespace MEDKIT\_TS;  
  
create table DiagnosesToMedicines (  
 id int generated always as identity (start with 1 increment by 1),  
 primary key (id),  
  
 diagnose\_id int not null,  
 foreign key (diagnose\_id) references Diagnoses(diagnose\_id),  
  
 medicine\_id int not null,  
 foreign key (medicine\_id) references Medicines(medicine\_id)  
) tablespace MEDKIT\_TS;

create table DiagnosesToSymptoms (  
 id int generated always as identity (start with 1 increment by 1),  
 primary key (id),  
  
 diagnose\_id int not null,  
 foreign key (diagnose\_id) references Diagnoses(diagnose\_id),  
  
 symptom\_id int not null,  
 foreign key (symptom\_id) references Symptoms(symptom\_id)  
) tablespace MEDKIT\_TS;  
  
create table DiseasesToSymptoms (  
 id int generated always as identity (start with 1 increment by 1),  
 primary key (id),  
  
 disease\_id int not null,  
 foreign key (disease\_id) references Diseases(disease\_id),  
  
 symptom\_id int not null,  
 foreign key (symptom\_id) references Symptoms(symptom\_id)  
) tablespace MEDKIT\_TS;