

Министерство образования и науки РФ
Пермский национальный исследовательский политехнический университет
Электротехнический факультет
Кафедра Информационные технологии и автоматизированные
системы

Базы данных
Отчет по курсовому проекту

Тема: «Информационная система аптеки»

Выполнила: студентка группы
РИС-19-16
Степанова А. А.
Проверил: доцент кафедры
ИТАС
Петренко А.А.

Пермь, 2021

Содержание

Введение	3
1 Анализ предметной области	4
1.1 Сравнительный анализ сред реализации приложения	6
1.2 Сравнительный анализ языков программирования	7
2 Структура базы данных	8
2.1 Концептуальная модель данных	8
2.2 Логическая модель	9
2.3 Физическая модель	10
3 Технология реализации	11
3.1 Создание базы данных	11
3.2 Разработка веб-приложения	12
3.3 Дизайн графического интерфейса	14
Заключение	16
Список использованной литературы	17
Приложение А. Постановка задачи	18
Приложение Б. Запросы создания процедур	21

Введение

В настоящее время жизнь людей зависит от различного рода информации, которую необходимо хранить и, возможно, всячески видоизменять. В такие моменты на помощь приходят базы данных и информационные системы, взаимодействующие с ними. Использование баз данных и информационных систем становится неотъемлемой составляющей деловой деятельности современного человека и функционирования преуспевающих организаций.

База данных (БД) представляет собой совокупность специальным образом организованных данных, хранимых в памяти вычислительной системы и отображающих состояние объектов и их взаимосвязей в рассматриваемой предметной области. [1]

Целью курсовой работы является разработка информационной системы аптеки.

Для достижения данной цели необходимо решить следующие задачи:

1. провести анализ предметной области;
2. разработать структуру базы данных;
3. реализовать приложение для работы с разработанной базой данных;

1 Анализ предметной области

Предметной областью является деятельность аптеки. Детально предметная область описана в тексте постановки задачи варианта № 72:

“Аптека продает *медикаменты* и изготавливает их по *рецептам*. Лекарства могут быть разных *типов*:

1. готовые лекарства: таблетки, мази, настойки;
2. изготавливаемые аптекой: микстуры, мази, растворы, настойки, порошки.

Различие в типах лекарств отражается в наборе атрибутов, их характеризующих. Микстуры и порошки изготавливаются только для внутреннего применения, растворы для наружного, внутреннего применения и для смешивания с другими лекарствами и мази только для наружного применения. Лекарства различны также по способу приготовления и по времени приготовления. Порошки и мази изготавливаются смешиванием различных компонент. При изготовлении растворов и микстур ингредиенты не только смешивают, но и отстаивают с последующей фильтрацией лекарства, что увеличивает время изготовления.

В аптеке существует справочник *технологий приготовления* различных лекарств. В нем указываются: идентификационный номер технологии, название лекарства и сам способ приготовления. На складе на все медикаменты устанавливается критическая норма, т.е. когда какого-либо вещества на складе меньше критической нормы, то составляются заявки на данные вещества и их в срочном порядке привозят с оптовых складов медикаментов.

Для изготовления аптекой лекарства больной должен принести *рецепт* от лечащего врача. В рецепте должно быть указано: ФИО, подпись и печать врача, ФИО, возраст и диагноз пациента, а также количество лекарства и способ применения. Больной отдает рецепт регистратору, он принимает *заказ*

и смотрит, есть ли компоненты заказываемого лекарства. Если не все компоненты имеются в наличии, регистратор делает *заявки* на оптовые склады лекарств и фиксирует ФИО, телефон и адрес необслуженного покупателя, чтобы сообщить ему, когда доставят нужные компоненты. Такой больной пополняет справочник заказов – это те заказы, которые находятся в процессе приготовления, с пометкой, что не все компоненты есть для заказа. Если все компоненты имеются, то они резервируются для лекарства больного. Покупатель выплачивает цену лекарства, ему возвращается рецепт с пометкой о времени изготовления. Больной также пополняет справочник заказов в производстве. В назначенное время больной приходит и по тому же рецепту получает готовое лекарство. Такой больной пополняет список отданных заказов.

Ведется статистика по объемам используемых медикаментов. Через определенный промежуток времени производится инвентаризация склада. Это делается для того, чтобы определить, есть ли лекарства с критической нормой или вышел срок хранения или недостача.”

Исходя из постановки задачи можно выделить шесть главных сущностей:

1. **Медикамент** (лекарство). Необходимо знать название медикамента, его количество на складе, цену и критическую норму.
2. **Тип** медикамента. Тип медикамента указывает способ применения лекарства.
3. **Рецепт** на лекарство. В рецепте указана информация о пациенте, лекарстве и его количестве.
4. **Технология приготовления** — информация о способе и времени приготовления.
5. **Заявка** на поставку медикаментов с оптовых складов.
6. **Покупатель**

Концептуальная, логическая и физическая модели базы данных будут нарисованы с использованием веб-сервиса draw.io. Концептуальная модель будет создана в соответствии с нотации Питера Чена, а логическая и физическая — в нотации Мартина (“Воронья лапка”).

1.1 Сравнительный анализ сред реализации приложения

Приложение можно реализовать как для смартфонов, персональных компьютеров, так и для веб-браузеров, и тогда оно будет доступно практически на любом устройстве. Соответственно выбор производится среди мобильных, десктопных и веб приложений. В таблице 1 описаны достоинства и недостатки каждого из приложений .

Таблица 1 — Достоинства и недостатки типов приложения

Тип приложения	Достоинства	Недостатки
Десктопное приложение	<ul style="list-style-type: none"> - скорость - возможность работы с локальной или с удаленной базами данных 	<ul style="list-style-type: none"> - необходимость реализации приложения для различных ОС - отсутствие мобильности - необходимость в многопоточности
Мобильное приложение	<ul style="list-style-type: none"> - мобильность - доступ к ресурсам устройства (камера, геолокация и т.д.) 	<ul style="list-style-type: none"> - необходимость реализации приложения для различных ОС - сложнее в реализации - необходимость в многопоточности
Веб-приложение	<ul style="list-style-type: none"> - возможность использования на любом устройстве, где есть браузер - не нуждается в установке - отсутствие необходимости в многопоточности - проще в реализации - возможность работы с любой СУБД 	<ul style="list-style-type: none"> - требуется постоянное подключение к интернету

Стоит отметить, что веб-приложения не требуют установки, в отличие от десктопных и мобильных, а также при внесении каких-либо изменений в проект они тут же отобразятся на стороне пользователя. Опираясь на это и на данные из таблицы было принято решение разрабатывать именно сайт информационной системы аптеки.

1.2 Сравнительный анализ языков программирования

В таблице 2 указаны самые популярные языки программирования (ЯП) для веб-разработки, их достоинства и недостатки.

Таблица 2 — Достоинства и недостатки языков программирования

Язык программирования	Достоинства	Недостатки
PHP	- кроссплатформенность	- отсутствие полноценного ООП - медленен в исполнении
JavaScript	- простота кода - скорость выполнения	- пониженный уровень безопасности
Python	- простота кода - кроссплатформенность - большое количество различных фреймворков	- медленен в исполнении

Наилучшим вариантом ЯП является Python, так как он имеет простой синтаксис, множество различных фреймворков, что упрощает дальнейшую работу. Одним из таких фреймворков является Flask. Именно он был выбран в качестве дополнительного инструмента для разработки веб-приложения.

Таким образом на основе анализа предметной области было решено реализовывать именно сайт, используя язык Python вместе с фреймворком Flask.

2 Структура базы данных

2.1 Концептуальная модель данных

Концептуальной моделью называют отражение предметной области, для которой разрабатывается база данных. Модель содержит информацию о сущностях и отношениях между ними. [3]

На рисунке 1 представлена концептуальная модель, выполненная в нотации Питера Чена.

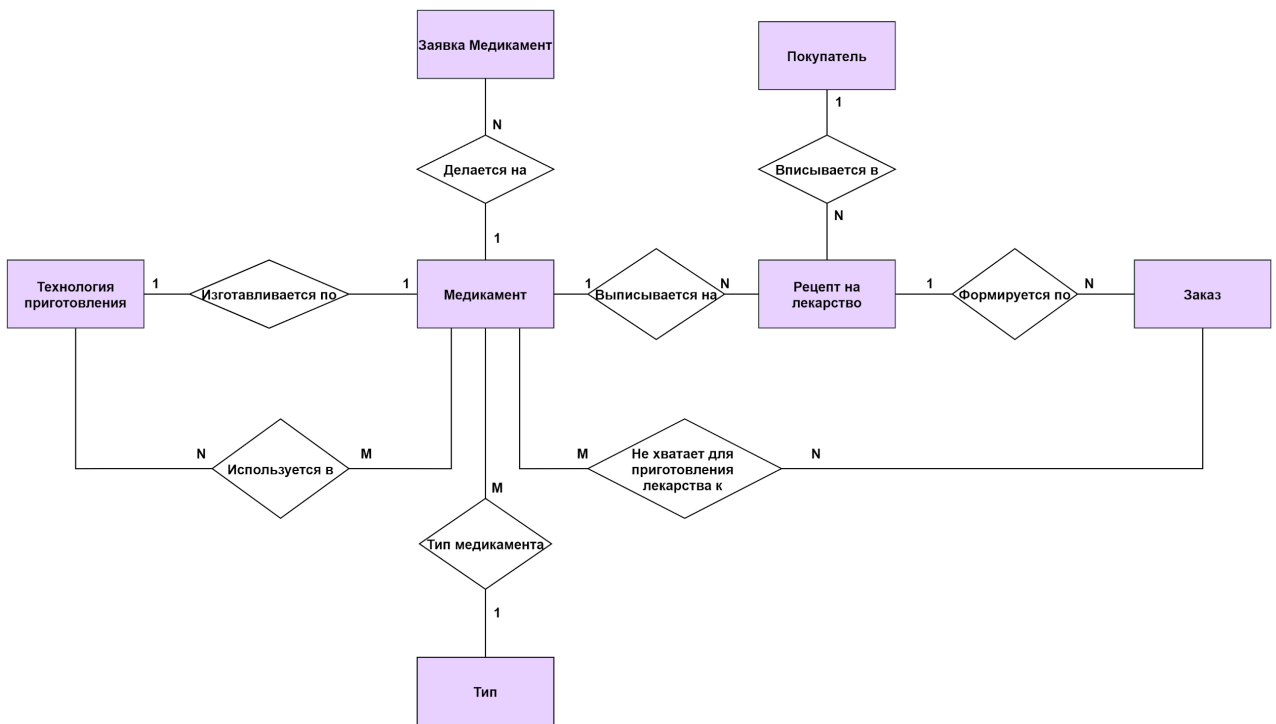


Рис. 1 — Концептуальная модель

Модель построена согласно анализу предметной области. На ней указаны все семь главных сущностей и связи между ними. Например, известно, что медикамент изготавливается по одной технологии, медикаменты могут быть разного типа (таблетки, порошки и так далее) и на них составляются заявки.

Помимо прочего, для покупки лекарства покупатель обязательно должен предоставить рецепт, по которому формируется заказ.

2.2 Логическая модель

Логическая модель — это схема базы данных, разработанная на основе конкретной модели данных. В логической модели также указывают сущности, связи между ними, с указанием первичных и вторичных ключей. Логическая модель базы данных курсового проекта представлена на рисунке 2.

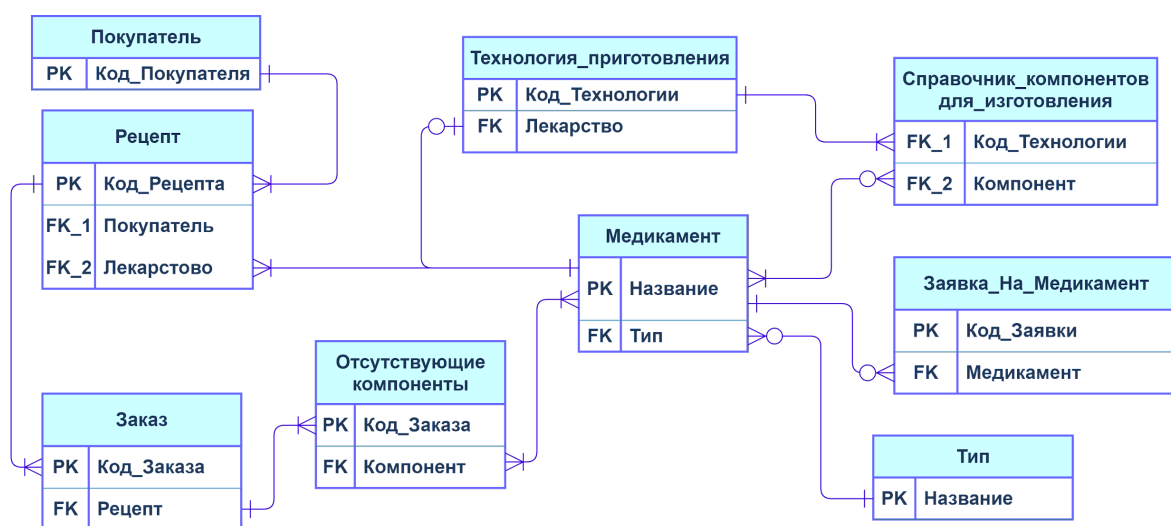


Рис. 2 — Логическая модель

Можно заметить, что к основным сущностям также добавилась вспомогательная таблица-справочник для реализации связи многие-ко-многим между сущностями Медикамент и Технология приготовления, которая будет содержать список требуемых компонентов для изготовления.

На основе данной логической модели будет построена физическая.

2.3 Физическая модель

Физическая модель базы данных строится на основе логической и содержит все детали, необходимые конкретной СУБД для создания базы: наименования таблиц и столбцов, типы данных, определения первичных и внешних ключей.

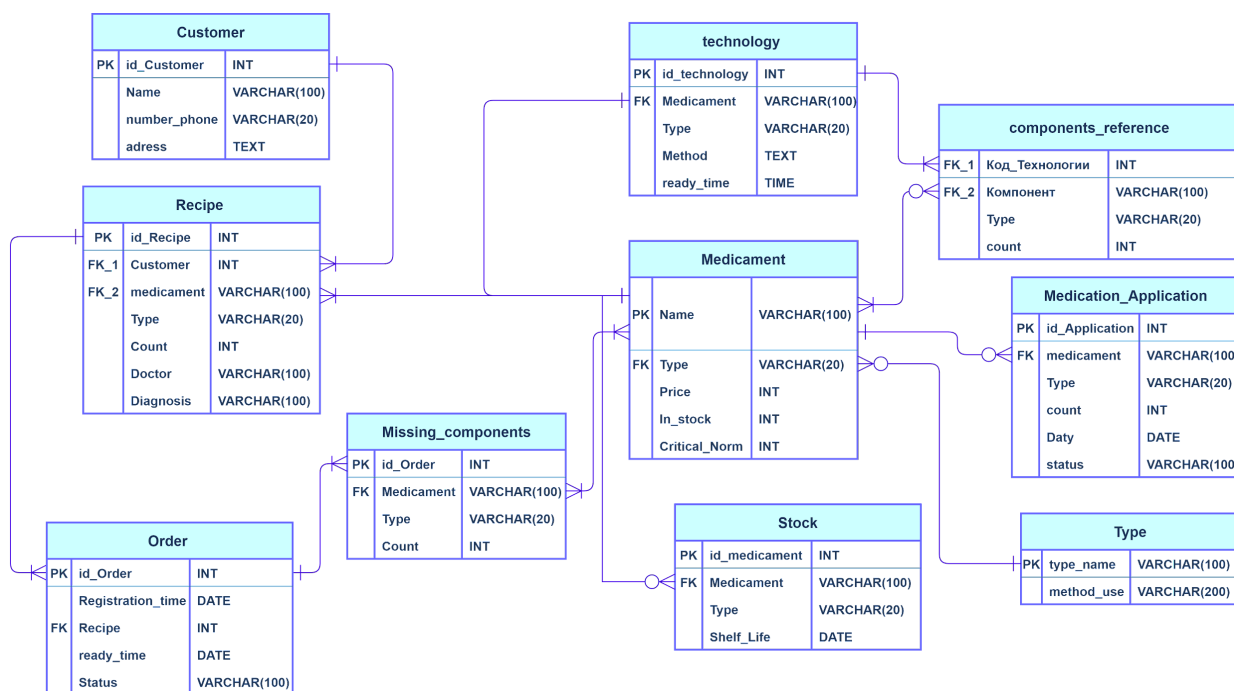


Рис. 3 — Физическая модель

На этом этапе построены концептуальная, логическая и физическая модели. На основе этих моделей можно переходить к реализации базы данных.

3 Технология реализации

3.1 Создание базы данных

База данных, содержащая все ранее описанные таблицы была создана в СУБД Postgresql. Для реализации некоторых запросов были созданы представления, например:

- `critical_norm_list` — предоставляет список медикаментов, запас которых ниже установленной критической нормы.
- `customer_exp` — посредством объединения таблиц `customer`, `recipe`, и `order`, выводит информацию о покупателях, которые не подошли в назначенной время за заказом.

Также для автоматизации управления базой данных были созданы хранимые процедуры, позволяющие уменьшить использование прямого sql кода в тексте программы веб-приложения. Описания некоторых из них:

- `accepted_request()` — в процедуре происходит изменение статуса заявки по переданному `id` заявки и вместе с этим добавляет в таблицу `stock` указанное в заявке количество записей о медикаментах. Данные, такие как тип и срок годности берутся из таблицы `medicament`.
- `add_tecnology()` — получая необходимые данные добавляет запись о технологии производства определенного медикамента, а также вносит в таблицу `components_referens` записи о требуемых компонентах для данной технологии.
- `add_recipe()` — процедура добавляет данные о рецепте. Она вызывается из описанной далее `register_order()` при регистрации заказа.
- `register_order()` — создает запись о рецепте, получая его `id` чтобы далее использовать его при создании самого заказа. Внутри процедуры устанавливается время регистрации заказа. В зависимости от статуса

заказа, определяется время его готовности. Например, для изготавливаемого лекарства учитывается время его приготовления. А при наличии уже готового медикамента, нужное его количество резервируется с общего склада медикаментов.

- `upd_orders()` — процедура вызывается из приложения при обращении к странице со списком заказов, в ней каждый проверяется наличие медикаментов из заказов. Если на склад прибыло нужное количество определенного лекарства, процедура производит резерв и обновляет статус заказа.

Дополнительно были созданы процедуры для добавления записей в таблицы медикаментов, заявок и общего склада.

Таким образом, хранимые процедуры позволяют уменьшить использование прямого `sql` кода в программе приложения и позволяет инкапсулировать процессы базы данных. Подробное описание создания всех процедур представлено в Приложении Б.

3.2 Разработка веб-приложения

Для взаимодействия приложения с базой данных необходимо установить к ней подключение. За подключение отвечает отдельная функция, в которой заранее прописаны данные пользователя — логин и пароль, адрес хоста, на котором будет работать приложение, и название базы данных, к которой производится подключение. Созданная база данных имеет имя `Pharmacy`.

Листинг 1 — функция подключения к бд.

```
def connection():
    conn = psycopg2.connect(dbname='Pharmacy',
                             user='postgres', password='Kraterept', host='localhost')
    return conn
```

Программа состоит из функций-контроллеров, которые вызываются при переходе пользователя по определенному интернет-адресу. В этих функциях содержится основной функционал программы. В ответ на запрос браузера функция-контроллер выполняет описанный в ней код и обязательно возвращает html страницу, сформированную на основе полученных результатов ее работы и затем она выводится в окне браузера.

Для примера рассмотрим функцию `show_catalog`, которая отображает таблицу с информацией о медикаментах:

Листинг 2 — функция `show_catalog`.

```
@app.route('/catalog')
def show_catalog():
    con = connection()
    cursor=con.cursor()
    cursor.execute("select * from all_catalog;")
    result = cursor.fetchall()
    return render_template('catalog.html', result=result)
```

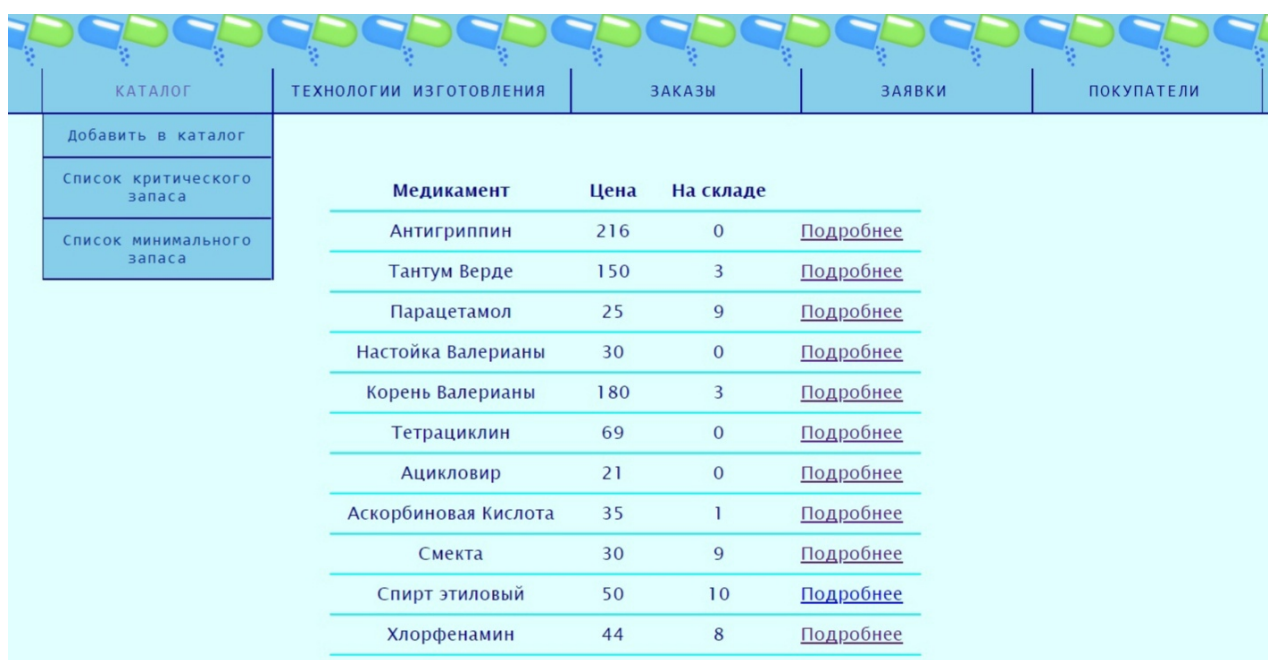
Данный код выполнится при переходе пользователя по адресу `"/catalog"`, для начала программа должна получить объект подключения к базе данных. Затем, при помощи этого объекта получается результат запроса, преобразуется в массив. Функция `render_template()` вызывает механизм шаблонов Jinja2, который поставляется в комплекте с Flask. Jinja2 заменяет блоки `{{...}}` соответствующими значениями, заданными аргументами, указанными в вызове `render_template().[2]`

Подводя итог, можно сказать что веб-приложение разрабатывалось на языке Python с использованием фреймворка Flask, который имеет дополнительный механизм шаблонов Jinja2, облегчающий вывод результатов выполнения функций-контроллеров при переходе пользователя по соответствующим интернет-адресам.

3.3 Дизайн графического интерфейса

Интерфейс программы состоит из верхнего навигационного меню и области в которую выводятся результаты запросов. Для создания веб-интерфейса используется язык разметки html (HyperText Markup Language — «язык гипертекстовой разметки») он отвечает за визуальную часть и логическое расположение данных на странице .

Навигационное меню — всплывающий список, разделенный на блоки. Все разделы меню доступны на каждой странице приложения. На рисунке 1 представлена страница с перечнем медикаментов, а также подменю “КАТАЛОГ”.



КАТАЛОГ	ТЕХНОЛОГИИ ИЗГОТОВЛЕНИЯ	ЗАКАЗЫ	ЗАЯВКИ	ПОКУПАТЕЛИ
Добавить в каталог				
Список критического запаса				
Список минимального запаса				
Медикамент	Цена	На складе		
Антигриппин	216	0	Подробнее	
Тантум Верде	150	3	Подробнее	
Парацетамол	25	9	Подробнее	
Настойка Валерианы	30	0	Подробнее	
Корень Валерианы	180	3	Подробнее	
Тетрациклин	69	0	Подробнее	
Ацикловир	21	0	Подробнее	
Аскорбиновая Кислота	35	1	Подробнее	
Смекта	30	9	Подробнее	
Спирт этиловый	50	10	Подробнее	
Хлорфенамин	44	8	Подробнее	

Рис. 4 — Страница “Каталог”.

Анимация всплывания подменю реализована с помощью средств CSS (Cascading Style Sheets - «каскадные таблицы стилей»). Пункты меню — ссылки, которые при нажатии перенаправляют пользователя на соответствующий url-адрес.

На рисунке 2 отображена форма для ввода данных о технологии производства. Кнопки “+” и “-” позволяют настроить количество полей, необходимых для ввода компонентов.

КАТАЛОГ | ТЕХНОЛОГИИ ИЗГОТОВЛЕНИЯ | ЗАКАЗЫ | ЗАЯВКИ | ПОКУПАТЕЛИ

Добавление технологии производства

МЕДИКАМЕНТ: Антигриппин

ТЕХНОЛОГИЯ ПРОИЗВОДСТВА:

ВРЕМЯ ПРИГОТОВЛЕНИЯ: minuts

КОМПОНЕНТ 1:

КОМПОНЕНТ 2:

КОМПОНЕНТ 3:

+ -

Добавить

Рис. 5 — Форма для добавления технологии производства.

Поля для ввода являются обязательными для заполнения, что обеспечивает защиту от некорректного ввода данных.

В итоге был разработан веб-интерфейс, функционал которого отвечает поставленным задачам. В реализации были использованы такие средства, как html, для создания “скелета” страниц, и инструмент CSS для стилизации и анимации различных элементов.

Заключение

Для выполнения курсовой работы были использованы СУБД Postgresql, язык программирования Python, фреймворк Flask и шаблонизатор гипертекстовых страниц Jinja.

Был проведен анализ предметной области, разработана база данных, предварительно определив ее структуру, реализовано приложение для работы с полученной БД. Иными словами, поставленные задачи — решены.

Соответственно, цель курсовой работы — реализация информационной системы аптеки — также достигнута.

При дальнейшей доработке приложения, в перспективе, данная информационная система может использоваться для учета работы аптечной организации.

Список использованной литературы

1. Хомоненко А. Д. Базы данных: Учебник для высших учебных заведений/ В.М. Цыганков, М. Г. Мальцев, под ред. проф. А. Д. Хомоненко— СПб.: КОРОНА принт, 2002 — 672 с.
2. Гринберг М. Разработка веб-приложение с использованием Flask на языке Python / пер. с англ. А. Н. Киселева. — М: ДМК Пресс, 2014. — 272 с.
3. Голицына О. Л., Максимов Н. В., Попов И. И. Базы данных: учеб. пособие — 2-е изда. испр. и доп.. — М: ФОРУМ: ИНФРА-М, 2009. — 400 с.
4. Дейт К. Дж. Введение в системы баз данных, 8-е издание.: Пер. с англ. — М.: Издательский дом “Вильямс”, 2005. — 1328 с.

Приложение А. Постановка задачи

72(3). Информационная система аптеки.

Аптека продает медикаменты и изготавливает их по рецептам. Лекарства могут быть разных типов:

- 1) готовые лекарства: таблетки, мази, настойки;
- 2) изготавливаемые аптекой: микстуры, мази, растворы, настойки, порошки.

Различие в типах лекарств отражается в наборе атрибутов, их характеризующих. Микстуры и порошки изготавливаются только для внутреннего применения, растворы для наружного, внутреннего применения и для смешивания с другими лекарствами и мази только для наружного применения. Лекарства различны также по способу приготовления и по времени приготовления. Порошки и мази изготавливаются смешиванием различных компонент. При изготовлении растворов и микстур ингредиенты не только смешивают, но и отстаивают с последующей фильтрацией лекарства, что увеличивает время изготовления.

В аптеке существует справочник технологий приготовления различных лекарств. В нем указываются: идентификационный номер технологии, название лекарства и сам способ приготовления. На складе на все медикаменты устанавливается критическая норма, т.е. когда какого-либо вещества на складе меньше критической нормы, то составляются заявки на данные вещества и их в срочном порядке привозят с оптовых складов медикаментов.

Для изготовления аптекой лекарства больной должен принести рецепт от лечащего врача. В рецепте должно быть указано: ФИО врача, ФИО, возраст и диагноз пациента, а также количество лекарства. Больной отдает рецепт регистратору, он принимает заказ и смотрит, есть ли компоненты заказываемого лекарства. Если не все компоненты имеются в наличии, регистратор делает заявки на оптовые склады лекарств и фиксирует ФИО,

телефон и адрес необслуженного покупателя, чтобы сообщить ему, когда доставят нужные компоненты. Такой больной пополняет справочник заказов – это те заказы, которые находятся в процессе приготовления, с пометкой, что не все компоненты есть для заказа. Если все компоненты имеются, то они резервируются для лекарства больного. Покупатель выплачивает цену лекарства, ему возвращается рецепт с пометкой о времени изготовления. Больной также пополняет справочник заказов в производстве. В назначенное время больной приходит и по тому же рецепту получает готовое лекарство. Такой больной пополняет список отданных заказов.

Ведется статистика по объемам используемых медикаментов. Через определенный промежуток времени производится инвентаризация склада. Это делается для того, чтобы определить, есть ли лекарства с критической нормой или вышел срок хранения.

Запросы в информационной системе:

1) Получите сведения о покупателях, которые не пришли забрать свой заказ в назначенное им время, и общее число покупателей.

2) Получите перечень и общее число покупателей, которые ждут прибытия на склад нужных им медикаментов в целом и по указанной категории медикаментов.

3) Получите перечень десяти наиболее часто используемых медикаментов в целом и в указанной категории медикаментов.

4) Подсчитайте, какой объем указанных веществ использован за указанный период.

5) Получите перечень и общее число покупателей, заказывавших определенные типы лекарств за данный период.

6) Получите перечень и типы лекарств, достигших своей критической нормы или закончившихся.

7) Получите перечень лекарств с минимальным запасом на складе в целом и по указанной категории медикаментов.

8) Получите полный перечень и общее число заказов, находящихся в производстве или ожидающих прибытия медикаментов на склад.

9) Получите полный перечень и общее число препаратов, требующихся заказам, находящимся в производстве.

10) Получите все технологии приготовления лекарств указанных типов.

11) Получите сведения о ценах на указанное лекарство в готовом виде и ценах на все компоненты, требующиеся для этого лекарства.

12) Получите сведения о постоянных клиентах медикаментов.

13) Получите сведения о конкретном лекарстве (его тип, способ приготовления, названия всех компонент, цены, его количество на складе).

Приложение Б. Запросы создания процедур

```
CREATE OR REPLACE PROCEDURE public.accepted_request(IN id
integer)
    LANGUAGE 'plpgsql'

AS $BODY$
DECLARE nam VARCHAR; life interval; accept DATE;

begin
accept := now();
UPDATE request SET date_completed= accept where id_request = id;

nam := (SELECT medicament FROM request where id_request = id);
life := (SELECT shelf_life FROM medicament where name = nam);
accept := accept + life;

FOR i IN 1..(SELECT count FROM request where id_request = id)
LOOP

    CALL add_stock(nam, cast(now()+life as date));

END LOOP;

UPDATE medicament SET in_stock = in_stock + cast((SELECT count
FROM request where id_request = id) as integer)
    where name = nam;

end
$BODY$;
```

```
CREATE OR REPLACE PROCEDURE public.add_customer(IN nam character
varying)
    LANGUAGE 'sql'

AS $BODY$
insert into customer (name) values(nam);
$BODY$;
```

```
CREATE OR REPLACE PROCEDURE public.add_medicament(
    med character varying,
    typ character varying,
    pric integer,
    crit_norm integer,
    life interval)
LANGUAGE 'sql'
AS $BODY$
INSERT INTO medicament
VALUES (med,
    typ,
```

```

        (SELECT COUNT(*) FROM stock where stock.medicament =
med),
        pric,
        crit_norm, life);
$BODY$;

```

```

CREATE OR REPLACE PROCEDURE public.add_recipe(
    customer_id integer,
    med character varying,
    amount integer,
    doct character varying,
    diagnos character varying)
LANGUAGE 'plpgsql'
AS $BODY$
begin

insert into recipe (customer, medicament, count, doctor,
diagnosis)
                                values (customer_id, med, amount, doct,
diagnos);
end
$BODY$;

```

```

CREATE OR REPLACE PROCEDURE public.add_stock(IN med character
varying, IN life date)
    LANGUAGE 'sql'

AS $BODY$
insert into stock (medicament, shelf_life) values (med, life);
$BODY$;

```

```

CREATE OR REPLACE PROCEDURE public.add_tecnology(
    med character varying,
    methodd text,
    components character varying[],
    r_time interval)
LANGUAGE 'plpgsql'
AS $BODY$
DECLARE
id_tecn integer;
i varchar;
begin
insert into technology (medicament, method, ready_time) values
(med, methodd, r_time);
id_tecn:= (select id_technology from technology order by
id_technology desc limit 1);

FOREACH i IN array components LOOP
    insert into components_reference (medicament, technology)
values (i, id_tecn);

```

```
END LOOP;
```

```
end  
$BODY$;
```

```
CREATE OR REPLACE PROCEDURE public.create_request(  
    med character varying,  
    amount integer)  
LANGUAGE 'sql'  
AS $BODY$  
INSERT INTO request (medicament, count,date_registration)  
    VALUES (med, amount, NOW());  
$BODY$;
```

```
CREATE OR REPLACE PROCEDURE public.register_order(  
    med character varying,  
    amount integer,  
    customerr integer,  
    doctorr character varying,  
    diagnos character varying,  
    statuss character varying)  
LANGUAGE 'plpgsql'  
AS $BODY$  
DECLARE id_rec int;  
reg_time timestamp(0) without time zone;  
ready_time timestamp(0) without time zone;  
b int;  
id_del int;  
i int;  
miss int;  
id_orderr int;  
begin  
reg_time:= CURRENT_TIMESTAMP(0);  
call add_recipe(customerr, med, amount, doctorr, diagnos);  
id_rec := (SELECT recipe.id_recipe FROM recipe ORDER BY  
id_recipe DESC LIMIT 1);  
  
if statuss = 'Нет на складе' then  
    ready_time = NULL;  
    INSERT INTO "order" (registration_time, recipe, ready_time,  
status)  
        VALUES(reg_time, id_rec, ready_time, statuss);  
    id_orderr = (select id_order from "order" where  
recipe=id_rec);  
    miss = amount - (select count(name) from medicament where  
name=med group by name);  
    insert into missing_components (id_order, medicament,  
count) VALUES (id_orderr, med, miss);  
else  
    if statuss = 'В производстве' then
```

```

        ready_time = reg_time + (select technology.ready_time
from technology where medicament=med);
    end if;
    if status = 'Готов' then
        ready_time = reg_time;
        for j IN 1..amount LOOP
            id_del := (select id_medicament from stock where
medicament=med LIMIT 1);
            delete from stock where id_medicament=id_del;
            update medicament set in_stock = in_stock - 1
where name=med;
        end LOOP;
    end if;

    INSERT INTO "order" (registration_time, recipe, ready_time,
status)
    VALUES(reg_time, id_rec, ready_time, status);
end if;

end
$BODY$;

```

```

CREATE OR REPLACE PROCEDURE public.upd_orders(
    )
LANGUAGE 'plpgsql'
AS $BODY$
declare
id_orders int[];
id_rec int;
i int;
j int;
med varchar;
amount int;
id_del int;
begin
id_orders:= ARRAY(select id_order from "order" where status =
'Нет на складе');
foreach i in array id_orders LOOP
    id_rec := (select recipe from "order" where id_order = i);
    med := (select medicament from recipe where
id_recipe=id_rec);
    amount := (select "count" from recipe where
id_recipe=id_rec);
    if ((select count(medicament) from stock where
medicament=med) >= amount) then
        update "order" set status = 'Готов',
ready_time=(now()::timestamp(0)) where id_order = i;
        for j IN 1..amount LOOP
            id_del := (select id_medicament from stock where
medicament=med LIMIT 1);

```



```

        delete from stock where id_medicament=id_del;
    end LOOP;
    delete from missing_components where id_order=i;
end if;
end LOOP;
id_orders:= ARRAY(select id_order from "order" where status = 'B
производстве');
foreach i in array id_orders LOOP
    id_rec := (select recipe from "order" where id_order = i);
    med := (select medicament from recipe where
id_recipe=id_rec);
    amount := (select "count" from recipe where
id_recipe=id_rec);
    if ((select registration_time from "order" where
id_order=i) +
        (select ready_time from technology where
medicament = med) < now()) then
        update "order" set status = 'Готов', ready_time=now()
where id_order = i;
        for j IN 1..amount LOOP
            id_del := (select id_medicament from stock where
medicament=med LIMIT 1);
            delete from stock where id_medicament=id_del;
        end LOOP;
    end if;
end LOOP;
end;
$BODY$;

```