СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 5](#_Toc26483221)

[1. ПОСТРОЕНИЕ ИНФОЛОГИЧЕСКОЙ КОНЦЕПТУАЛЬНОЙ МОДЕЛИ 6](#_Toc26483222)

[1.1 Анализ предметной области и выявление необходимого набора сущностей 6](#_Toc26483223)

[1.2 Обоснование требуемого набора атрибутов для каждой сущности и выделение идентифицирующих атрибутов 6](#_Toc26483224)

[1.3 Определение связей между объектами 8](#_Toc26483227)

[1.4 Описание полученной модели на языке инфологического проектирования 9](#_Toc26483228)

[2. ПОСТРОЕНИЕ СХЕМЫ РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ 10](#_Toc26483229)

[2.1 Построение набора необходимых отношений базы данных 10](#_Toc26483230)

[2.2 Задание первичных и внешних ключей определенных отношений 11](#_Toc26483232)

[Окончание таблицы 2.1 12](#_Toc26483233)

[2.3 Третья нормальная форма 12](#_Toc26483234)

[2.4 Определение ограничений целостности для внешних ключей отношений и для отношений в целом 13](#_Toc26483235)

[2.5 Графическое представление связей между внешними и первичными ключами 14](#_Toc26483236)

[3. СОЗДАНИЕ СПРОЕКТИРОВАННОЙ БАЗЫ ДАННЫХ 15](#_Toc26483237)

[4 ЗАПИСЬ ВЫРАЖЕНИЙ УКАЗАННЫХ В ВАРИАНТЕ ЗАДАНИЯ ТИПОВ ЗАПРОСОВ НА ЯЗЫКЕ SQL 18](#_Toc26483238)

[4.1 Запросы используемые в спроектированной базе данных 18](#_Toc26483239)

[4.2 Триггеры используемые в спроектированной базе данных 19](#_Toc26483246)

[5 ВЫБОР И ОСНОВАНИЕ СРЕДСТВ РАЗРАБОТКИ ПРИЛОЖЕНИЯ 21](#_Toc26483247)

[6 РЕАЛИЗАЦИЯ ЗАКОНЧЕННОГО ПРИЛОЖЕНИЯ, РАБОТАЮЩЕГО С СОЗДАННОЙ БАЗОЙ ДАННЫХ 22](#_Toc26483248)

[6.1 Разработка и построение интерфейса главной и рабочих страниц 22](#_Toc26483249)

[6.2 Построение главного навигационного меню сайта 22](#_Toc26483250)

[6.3 Выполнение программного кода на Spring Boot 23](#_Toc26483251)

[ЗАКЛЮЧЕНИЕ 25](#_Toc26483252)

[Список использованных источников 26](#_Toc26483254)

[ПРИЛОЖЕНИЕ А: КОНЦЕПТУАЛЬНАЯ СХЕМА БД 27](#_Toc26483255)

[ПРИЛОЖЕНИЕ Б: СХЕМА РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ 28](#_Toc26483256)

[ПРИЛОЖЕНИЕ В: ГЛАВНАЯ И РАБОЧИЕ ФОРМЫ ПРИЛОЖЕНИЯ 29](#_Toc26483257)

[ПРИЛОЖЕНИЕ Г: ЛИСТИНГ ПРОГРАММЫ 30](#_Toc26483258)

## ВВЕДЕНИЕ

Тема курсовой работы – проектирование реляционной базы данных интернет-магазина ритуальных товаров и разработка web-приложения для взаимодействия с созданной базой данных.

В соответствии с предметной областью работы, будущая база данных должна хранить сведения о заказах, клиентах, товарах, сообщениях, продажах и обеспечивать целостность этих данных.

Для обеспечения функционала, а также для удобства пользования сайтом необходимо разработать административную часть приложения, котороя позволит добавлять, удалять, редактировать товары, заказы клиентов, выводить информацию о товарах, продажах, распределениях товара. Приложение должно быть простым в использовании, которым могли бы пользоваться даже неквалифицированные сотрудники.

Аналогов данного приложения немало и не только в данной области. Актуальность работы так же не вызывает вопросов, так как со временем все переходит в более удобный и мобльный формат. Создаваемый интернет-магазин, для удобства, будет представлять абстрактное бюро ритуальных услуг.

Для создания базы данных будет использоваться СУДБ Postgresql. Для создания приложения – Spring Framework 4.

## ПОСТРОЕНИЕ ИНФОЛОГИЧЕСКОЙ КОНЦЕПТУАЛЬНОЙ МОДЕЛИ

1.1 Анализ предметной области и выявление необходимого набора сущностей

Для построения информационной системы требуется для начала выделить необходимы набор сущностей, которые описывают эту систему. Данный набор должен удовлетворять всем условиям на проектирование системы.

Определим минимальный набор сущностей, необходимый для проектирования информационной системы регистратуры поликлиники. Для определения первичного набора сущностей будет проведён анализ технического задания и предметной области.

Данной информационной системой может пользоваться не только покупаель, но также и администратор, который вносит изменение в базу данных. Отсюда следует необходимость выделения сущности – *Role* которая будет хранить роли*.*

Так как данный проект является веб-приложением, то должно предусматривается авторизация для каждой из ролей. Отсюда следует необходимость выделения сущности – *Users* которая будет хранить идентификационные данные каждой из ролей.

Данные каждого клиента представлены в виде имя, пароль, активность.

труСктура данных о товарах выглядит следующим образом: каждый товар имеет категорию (*Category*), стоимость, название, описание, изображение(*Files*). Следовательно, информация о данных структурах будет храниться в соответствующих сущностях.

Для хранения и учета всех заказов необходима описание такой сущности как *Deal.*

* 1. Обоснование требуемого набора атрибутов для каждой сущности и выделение идентифицирующих атрибутов

Для построения инфологической концептуальной модели необходимо для каждой сущности, выявленной в предыдущем пункте, определить требуемый набор атрибутов. Атрибутом является поименованная характеристика сущности. Его наименование должно быть уникальным для конкретного типа сущности, но может быть одинаковым для различного типа сущностей. Атрибуты используются для определения того, какая информация должна быть собрана о сущности.

В таблице 1.1 представлены сущности, определенные для них атрибуты и описание атрибутов, а также ключи.

Таблица 1.1 – Описание сущностей

|  |  |  |  |
| --- | --- | --- | --- |
| **Таблица** | **Поле** | **Ключ** | **Описание** |
| customer | id | PK | Идентификационный номер таблицы " customer ". |
| dtype |  | Атрибут типа данных. |
| username |  | Login учетной записи. |
| password |  | Пароль учетной записи. |
| role | id | PK | Идентификационный номер таблицы " role ". |
| name |  | Имя уровня доступа. |
| product | id | PK | Идентификационный номер таблицы " product ". |
| category\_id | FK | Идентификационный номер таблицы " category ". |
| Name |  | Название продукта |
| Cost |  | Цена товара |
| Description |  | Описание товара |
| Src |  | Ссылка на изображение. |
| category | id | PK | Идентификационный номер таблицы " category ". |
| Name |  | Название категории. |
| Message | id | PK | Идентификационный номер таблицы " Message ". |
| id\_customer | FK | Идентификационный номер таблицы " customer ". |
| text |  | Текст собщения |
| header |  | Заголовок сообщения |
| Date |  | Дата отправки сообщения |
| Basket | id | PK | Идентификационный номер таблицы "stret". |

Окончание таблицы 1.1

|  |  |  |  |
| --- | --- | --- | --- |
| Basket | id\_ customer | FK | Идентификационный номер таблицы " customer ". |
| User\_roles | id | PK | Идентификационный номер таблицы " User\_roles ". |
| id\_customer | FK | Идентификационный номер таблицы " customer ". |
| id\_roles |  | Идентификационный номер таблицы " Deal ". |
| Deal | id | PK | Идентификационный номер таблицы "tickets". |
| Id\_product | FK | Идентификационный номер таблицы " product ". |
| id\_customer | FK | Идентификационный номер таблицы " customer ". |
| date |  | Дата начала приёма. |
| bought |  | Факт покупки |
| Basket\_deal | Basket\_id | FK | Идентификационный номер таблицы " Basket ". |
| Deal\_id | FK | Идентификационный номер таблицы " Deal ". |

* 1. Определение связей между объектами

Кроме атрибутов каждой сущности модель данных должна определять связи между сущностями. На концептуальном уровне связи представляют собой простые ассоциации между сущностями.

Связь – это ассоциирование двух или более сущностей. Если бы назначением базы данных было только хранение отдельных, не связанных между собой данных, то ее структура могла бы быть очень простой. Однако, одно из основных требований к организации базы данных – это обеспечение возможности отыскания одних сущностей по значениям других, для чего необходимо установить между ними определенные связи. А так как в реальных базах данных нередко содержатся десятки или даже сотни сущностей, то между ними может быть установлено великое множество связей. Наличие такого множества связей и определяет сложность инфологических моделей.

Для реализации информационной системы регистратуры поликлиники необходимо установить все связи между объектами. А именно, нужно рассмотреть всю информационную систему в совокупности и определить взаимное влияние объектов, составляющих систему.

1. Customer — Role : (M:M).
2. Customer – Basket : (1:M).
3. Customer – Messages : (1:M).
4. Customer – User\_roles : (1:M).
5. Role – User\_roles : (1:M).
6. Category – Product : (1:M).
7. Product – Deal : (1:M).
8. Deal – Customer: (1:M).
9. Deal – Basket : (M:M).
10. Deal –Basket\_deals: (1:M).
11. Basket – Basket\_deals : (1:M).

1.4 Описание полученной модели на языке инфологического проектирования

Проектирование инфологической модели предметной области – частично формализованное описание объектов предметной области в терминах некоторой семантической модели, например, в терминах ER-модели (*англ.* entity-relationship model).

По правилам построения ER-диаграмм сущность изображается в виде прямоугольника. Связь изображается линией, которая связывает две сущности, участвующие в отношении. Степень конца связи указывается графически, множественность связи изображается в виде «вилки» на конце связи. Модальность связи так же изображается графически — необязательность связи помечается кружком на конце связи. Атрибуты сущности записываются внутри прямоугольника, изображающего сущность.

Таким образом, на основании результатов предыдущих пунктов, получим ER-диаграмму, проектируемой базы данных, представленную в приложении A, рисунок А.1.

## 2. ПОСТРОЕНИЕ СХЕМЫ РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ

2.1 **Построение набора необходимых отношений базы данных**

Чтобы построить схему реляционной базы данных необходимо определить совокупность отношений, которые составляют базу данных. Эта совокупность отношений будет содержать всю информацию, которая должна храниться в базе данных.

В предыдущем пункте описана инфологическая концептуальная модель базы данных «Бюро ритуальных услуг», построенной с помощью IDEF1X. На основе полученной концептуальной модели можно определить набор необходимых отношений базы данных. На рисунке 2.1 представлены отношения для базы данных.

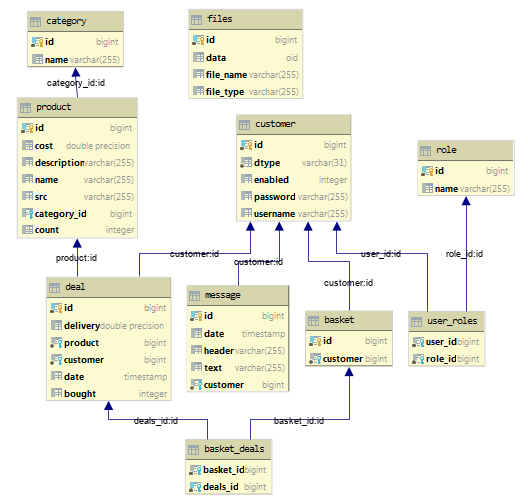


Рисунок 2.1 – Набор необходимых отношений базы данных

## 

## 2.2 **Задание первичных и внешних ключей определенных отношений**

Поле первичного ключа служит уникальным определением записи. Оно также служит для связи таблиц. В связанных таблицах первичный ключ родительской таблицы становится внешним ключом в дочерней таблице. Внешний ключ дочерней таблицы отсылает к сведениям родительской таблицы.

Первичные ключи будут иметь постфикс PK, вторичные – FK. Также в подавляющем числе случаев названия вторичных ключей будут соответствовать названию связанных таблиц (без множественного числа), при несоответствии укажем название связанной таблицы во избежание путаницы. Также следует упомянуть, что в контексте данной работы первичные ключи будут являться единственным полем в таблице.

Таблица 2.1 – Первичные и внешние ключи отношений

|  |  |  |
| --- | --- | --- |
| **Таблица** | **Поле** | **Ключ** |
| Message | id | PK |
| id\_customer | FK |
| text |  |
| header |  |
| Date |  |
| Basket | id | PK |
| id\_ customer | FK |
| Deal | id | PK |
| Id\_product | FK |
| id\_customer | FK |
| date |  |
| bought |  |
| Basket\_deal | Basket\_id | FK |
|  |  |
| Message | id | PK |
| id\_customer | FK |
| text |  |
| header |  |
| Date |  |
| Basket | id | PK |
| id\_ customer | FK |

Окончание таблицы 2.1

|  |  |  |
| --- | --- | --- |
| **Таблица** | **Поле** | **Ключ** |
| Basket | id | PK |
| id\_ customer | FK |
| User\_roles | id | PK |
| id\_customer | FK |
| id\_roles |  |
| Deal | id | PK |
| Id\_product | FK |
| id\_customer | FK |
| date |  |
| bought |  |
| Basket\_deal | Basket\_id | FK |
| Deal\_id | FK |

2.3 Третья нормальная форма

Процесс преобразования базы данных к виду, отвечающему нормальным формам, называется нормализацией. Нормализация предназначена для приведения структуры базы данных к виду, обеспечивающему минимальную избыточность, то есть нормализация не имеет целью уменьшение или увеличение производительности работы или же уменьшение, или увеличение объёма БД. Конечной целью нормализации является уменьшение потенциальной противоречивости хранимой в БД информации.

Нормальная форма — свойство  в , характеризующее его с точки зрения избыточности, которая потенциально может привести к логически ошибочным результатам выборки или изменения данных. Нормальная форма определяется как совокупность требований, которым должно удовлетворять отношение.

Процесс преобразования отношений базы данных к виду, отвечающему нормальным формам, называется нормализацией.

Так как все атрибуты наших отношений атомарны, а каждое отношение имеет первичный ключ, то это означает, что отношения базы находятся в первой нормальной форме (1НФ).

Так как зависимости неключевых атрибутов от части составного ключа отсутствуют (все ключи в вышеописанных отношениях несоставные), а отношения базы находятся в 1НФ, то можно утверждать, что отношения базы удовлетворяют требованиям второй нормальной формы (2НФ).

Так как в отношениях базы отсутствуют зависимости неключевых атрибутов от других неключевых атрибутов, а присутствие 2НФ указано выше, можно сказать, что отношения базы находятся в третьей нормальной форме (3НФ).

2.4 Определение ограничений целостности для внешних ключей отношений и для отношений в целом

Ограничение целостности отношений заключается в том, что в любом отношении должны отсутствовать записи с одним и тем же значением первичного ключа. Конкретно требование состоит в том, что любая запись любого отношения должна быть отличной от любой другой записи этого отношения. Это требование автоматически удовлетворяется, если в системе не нарушаются базовые свойства отношений.

Ограничение целостности для внешних ключей состоит в том, что значение внешнего ключа должно быть равным значению первичного ключа цели; либо быть полностью неопределенным, т.е. каждое значение атрибута, участвующего во внешнем ключе должно быть неопределенным.

Условиями целостности называется набор правил, используемых для поддержания допустимых межтабличных связей и запрета на случайное изменение или удаление связанных данных. Следует устанавливать целостность данных только при выполнении следующих условий: связываемое поле из главной таблицы является полем первичного ключа и имеет уникальный индекс, связанные поля имеют один и тот же тип данных.

Для автоматического обновления связанных полей (удаления записей) при обновлении (удалении) в главной таблице, следует устанавливать обеспечение целостности данных и каскадное обновление связанных полей (каскадное удаление связанных записей).

Ограничение целостности, накладываемые на разрабатываемую систему:

* ключевое поле отношения должно быть уникальным;
* внешний ключ должен быть повторяющимся, то есть соответствовать уникальному ключу в своем отношении.

Для удовлетворения требования ограничения целостности для внешних ключей отношений и для отношений в целом необходимо, чтобы выполнялось соответствие между типами вводимых данных и типами столбцов в таблицах, а так же чтобы были заполнены все обязательные поля в таблицах, т.е. те поля которые не могут содержать значения NULL.

2.5 Графическое представление связей между внешними и первичными ключами

По результатам нормализации, определении первичных и внешних ключей, связей между сущностями, была получена схема реляционной базы данных, представленная в приложении B на рисунке B.2. На ней изображаются все отношения базы данных, а также связей между внешними и первичными ключами. Первичные ключи обозначаются буквами PK (*от англ.* Primary Key – первичный ключ), внешние ключи обозначаются FK (*от англ.* Foreign Key – внешний ключ).

## 3. СОЗДАНИЕ СПРОЕКТИРОВАННОЙ БАЗЫ ДАННЫХ

Для реализации спроектированной базы данных была выбрана система управления базами данных PostgreSQL 9.6. Это обусловлено тем, что, во-первых, данная СУБД получилась широкое распространение, а во-вторых, имеются свободно распространяемые сборки.

Произведем описание структуру каждой из таблиц с описанием типа полей.

Таблица *Customer* хранит авторотационные данные учетных записей. Её структура приведена в таблице 3.1.

Таблица 3.1 – Характеристика атрибутов таблицы Customer.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| id | INT | Идентификационный номер таблицы " Customer ". |
| dtype | VARCHAR | Служебный тип данных. |
| login | VARCHAR | login учетной записи. |
| password | VARCHAR | Пароль учетной записи. |
| enabled | BOOLEAN | Активность пользователя. |

Таблица *Role* содержит основные роли, предоставленные для работы с приложением. Её структура приведена в таблице 3.2.

Таблица 3.2 – Характеристика атрибутов таблицы Role.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| id | INT | Идентификационный номер таблицы " Role ". |
| Role | VARCHAR | Имя роли. |

Таблица *Product* содержит основные сведения о реализуемых продуктах и услугах. Её структура приведена в таблице 3.3.

**Таблица 3.3 –** Характеристика атрибутов таблицы Product.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| id | INT | Идентификационный номер таблицы " Product ". |
| Cost | INT | Цена продукта. |
| Description | VARCHAR | Описание продукта. |
| Name | VARCHAR | Название продукта. |
| Src | VARCHAR | Ссылка на изображение продукта. |

Окончание таблицы 3.3

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| Category\_id | BIGINT | Ссылка на таблицу “Category” |

Таблица *Deal* содержит сведения опокупках. Её структура приведена в таблице 3.4.

**Таблица 3.4 –** Характеристика атрибутов таблицы Deal.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| id | INT | Идентификационный номер таблицы "deal". |
| product\_id | INT | Идентификационный номер таблицы " product ". |
| customer\_id | INT | Идентификационный номер таблицы " customer ". |
| date | date | Дата сделки. |
| Bought | INT | Факт совершения сделки. |

Таблица *Basket* данные о корзине покупателя. Её структура приведена в таблице 3.5.

**Таблица 3.5 –** Характеристика атрибутов таблицы Basket.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| id | INT | Идентификационный номер таблицы " Basket ". |
| customer\_id | INT | Идентификационный номер таблицы "Cuestomer". |

Таблица *Basket\_deals* содержит список покупок пользователя. Её структура приведена в таблице 3.6.

**Таблица 3.6 –** Характеристика атрибутов таблицы Basket\_deals.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| id | INT | Идентификационный номер таблицы " Basket ". |
| Deal\_id | INT | Идентификационный номер таблицы " Deal ". |

Таблица *Category* содержит список всех доступных категорий. Её структура приведена в таблице 3.7.

**Таблица 3.7** – Характеристика атрибутов таблицы Category.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| id | INT | Идентификационный номер таблицы "category". |
| category | VARCHAR | Название категории. |

Таблица *User\_roles* содержит информацию о правах доступа каждого пользователя. Её структура приведена в таблице 3.8.

**Таблица 3.8 –** Характеристика атрибутов таблицы User\_roles.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| id | INT | Идентификационный номер таблицы " User\_roles ". |
| Role\_id | INT | Идентификационный номер таблицы " User\_roles ". |

Таблица *Message* содержит график работы всех врачей. Её структура приведена в таблице 3.9.

**Таблица 3.9** – Характеристика атрибутов таблицы Message.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| id | INT | Идентификационный номер таблицы " Message ". |
| Customer\_id | INT | Идентификационный номер таблицы " Customer ". |
| Header | VARCHAR | Заголовок сообщения. |
| Text | VARCHAR | Текст сообщения. |
| date | date | Дата и время отправки сообщения. |

Таблица *Files* содержит основные сведения о пациенте. Её структура приведена в таблице 3.10.

**Таблица 3.10** – Характеристика атрибутов таблицы Files.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| id | INT | Идентификационный номер таблицы " Files ". |
| Data | Bytes | Хранимые данные |
| File\_name | VARCHAR | Имя файла. |
| File\_type | VARCHAR | Тип файла. |

## 4 ЗАПИСЬ ВЫРАЖЕНИЙ УКАЗАННЫХ В ВАРИАНТЕ ЗАДАНИЯ ТИПОВ ЗАПРОСОВ НА ЯЗЫКЕ SQL

4.1 Запросы используемые в спроектированной базе данных

## Запрос, возвращающий список авторизационных данных учетных записей.

select role0\_.id as id1\_9\_, role0\_.name as name2\_9\_ from role role0\_;

1. Запрос, возвращающий список всех продуктов заданной категории.

select product0\_.id as id1\_8\_, product0\_.category\_id as category6\_8\_, product0\_.cost as cost2\_8\_, product0\_.description as descript3\_8\_, product0\_.name as name4\_8\_, product0\_.src as src5\_8\_ from product product0\_ where product0\_.category\_id=?;

1. Запрос, возвращающий файл изображения конкретного рподукта.

select dbfile0\_.id as id1\_6\_0\_, dbfile0\_.data as data2\_6\_0\_, dbfile0\_.file\_name as file\_nam3\_6\_0\_, dbfile0\_.file\_type as file\_typ4\_6\_0\_ from files dbfile0\_ where dbfile0\_.id=?;

1. Запрос, возвращающий список всех товаров и услуг содержащих в названии некоторые символы с нормализацией.

select products0\_.category\_id as category1\_3\_0\_, products0\_.products\_id as products2\_3\_0\_, product1\_.id as id1\_8\_1\_, product1\_.category\_id as category6\_8\_1\_, product1\_.cost as cost2\_8\_1\_, product1\_.description as descript3\_8\_1\_, product1\_.name as name4\_8\_1\_, product1\_.src as src5\_8\_1\_, category2\_.id as id1\_2\_2\_, category2\_.name as name2\_2\_2\_ from category\_products products0\_ inner join product product1\_ on products0\_.products\_id=product1\_.id left outer join category category2\_ on product1\_.category\_id=category2\_.id where products0\_.category\_id=?;

## Запрос, возвращающий информацию о продукте.

select product0\_.id as id1\_8\_0\_, product0\_.category\_id as category6\_8\_0\_, product0\_.cost as cost2\_8\_0\_, product0\_.description as descript3\_8\_0\_, product0\_.name as name4\_8\_0\_, product0\_.src as src5\_8\_0\_, category1\_.id as id1\_2\_1\_, category1\_.name as name2\_2\_1\_ from product product0\_ left outer join category category1\_ on product0\_.category\_id=category1\_.id where product0\_.id=?;

## Запрос, возвращающий корзину конкретного пользователя.

select basket0\_.id as id1\_0\_, basket0\_.customer as customer2\_0\_ from basket basket0\_ where basket0\_.customer=?;

## Запрос, возвращающий список покупок конкретного пользователя.

select deal0\_.id as id1\_5\_, deal0\_.bought as bought2\_5\_, deal0\_.date as date3\_5\_, deal0\_.product as product4\_5\_, deal0\_.customer as customer5\_5\_ from deal deal0\_ where deal0\_.customer=?

## Запрос, возвращающий сообщения конкретного пользователя.

select message0\_.id as id1\_7\_, message0\_.date as date2\_7\_, message0\_.header as header3\_7\_, message0\_.text as text4\_7\_, message0\_.customer as customer5\_7\_ from message message0\_ where message0\_.customer=?

## Запрос, возвращающий список покупок всех пользователей.

select deal0\_.id as id1\_5\_, deal0\_.bought as bought2\_5\_, deal0\_.date as date3\_5\_, deal0\_.product as product4\_5\_, deal0\_.customer as customer5\_5\_ from deal

4.2 Триггеры используемые в спроектированной базе данных

Триггер, рассылающий зарегистрированным пользователя сообщения при поступлении в ассортимент новых товаров:

create or replace function sendmessage() returns trigger as $sendmessage$

declare cust record;

begin

for cust in select id from customer

loop

insert into message(date, header, text, customer)

values (now(),'Пополнение ассортимента', (select 'В категорию '||

(select c.name from category c

right join product cp on c.id = cp.category\_id

where cp.id=NEW.id

limit 1)

|| ' поступили новые товары. ' as text) ,

cust.id);

end loop;

if new.id is null then raise notice 'nol'; end if;

return NULL ;

end

$sendmessage$

LANGUAGE plpgsql;

CREATE TRIGGER sendmessage after INSERT ON product

EXECUTE PROCEDURE sendmessage();

Триггер отправляющий в личный кабинет пользователя сообщение по факту совершения заказа:

create or replace function buy() returns trigger as $buy$

begin

if new.bought= 1 and old.bought=0 then

insert into message(date, header, text, customer)

values (now(),'Вы сделали заказ', (select 'Вы сделали заказ на '||

(select p.name from product p where p.id = new.product)

|| '. Курьер свяжется с вами в ближайшее время. ' as text) ,

new.customer);

end if;

return NULL ;

end

$buy$

LANGUAGE plpgsql;

CREATE TRIGGER buy after update ON deal

for each row EXECUTE PROCEDURE buy();

## 5 ВЫБОР И ОСНОВАНИЕ СРЕДСТВ РАЗРАБОТКИ ПРИЛОЖЕНИЯ

Для реализации задачи по разработке информационной системы туристического клуба была выбрана СУБД PostgreSQL 9.6. Данная СУБД является одной из наиболее популярных систем управления и обслуживанию баз данных. В ней применяются новейшие разработки и последние достижения в области проектирования, построения и обслуживания баз данных.

Для создания приложения используется Intellij IDEA Ultimate 2019.3, а языком написания приложения был выбран – Java, Spring Boot.

Spring Boot — это полезный проект, целью которого является упрощение создания приложений на основе Spring. Он позволяет наиболее простым способом создать web-приложение, требуя от разработчиков минимум усилий по его настройке и написанию кода

Spring Boot обладает большим функционалом, но его наиболее значимыми особенностями являются: управление зависимостями, автоматическая конфигурация и встроенные контейнеры сервлетов

ORM — Object-Relational Mapping или в переводе на русский объектно-реляционное отображение. Это технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования.

ORM — это по сути концепция о том, что Java объект можно представить как данные в БД (и наоборот). Она нашла воплощение в виде спецификации JPA — Java Persistence API.

При создании приложения используется реализация JPA — Hibernate. Это — одна из наиболее популярных реализаций ORM-модели. Целью Hibernate является освобождение разработчика от значительного объёма сравнительно низкоуровневого программирования при работе в объектно-ориентированных средствах в реляционной базе данных. Hibernate является библиотекой, созданной для работы со всеми видами баз данных.

## 6 РЕАЛИЗАЦИЯ ЗАКОНЧЕННОГО ПРИЛОЖЕНИЯ, РАБОТАЮЩЕГО С СОЗДАННОЙ БАЗОЙ ДАННЫХ

6.1 **Разработка и построение интерфейса главной и рабочих страниц**

Главная страница приложения представляет собой HTLM страницу и использованием элементов библиотеки Thymeleaf. Страница включает в себя несколько шаблонов, используемых во всех страницах приложения, таких как подвал и меню навигации. Для стилизации используется Bootstrap, что позволяет сделать приложене непротиворечивым по стилю и разместить всю информацию и элементы управления в соответсвии с правилами хорошй вёрстки.

Для представления данных большинства созданных сущностей свёрстаны отдельные страницы, что упрощает понимание и просмотр информации, т.к. она представляется в удобном, интуитивно доходчивом виде.

Основной упор при проектировании интерфейса приложения был сделан на привлекательность и понятность для конечного пользователя. Поэтому были заменены многие стандартные элементы оформления на доработанные.

При проектировании приложения были учтены все возможные случаи некорректной работы программы, поэтому большинство нештатных ситуаций сопровождается оповещениями с описанием проблемы.

Скриншоты главной и некоторых диалоговых окон представлены в приложении C.

6.2 **Построение главного навигационного меню сайта**

Главное навигационное меню сайта представлено ссылками на категории продаваемых товаров, строкой поиска и кнопками управления авторизацией. Данные пункты выполнены в виде одной панели, которая реагирует на нажатия пользователя и сопровождает все страницы приложения.

6.3 **Выполнение программного кода на Spring Boot**

Для построения и работы приложения используеться паттерн проектирования MVC. Реализация паттерна поставляется библиотеками Spring Boot. Это значит, что веб-сервер оперирует данными в виде классов-сушностей. Обрабатываются они в классах-сервисах и классах контроллеров, которые кроме того, отвечают за обработку HTTP запросов и навигацию по сайту. Главный класс приложения объединяющий компоненты и запускающий работу сервера:

1. import org.springframework.boot.CommandLineRunner;
2. import org.springframework.boot.SpringApplication;
3. import org.springframework.boot.autoconfigure.SpringBootApplication;
4. import org.springframework.context.ApplicationContext;
5. import org.springframework.context.annotation.Bean;
6. import java.io.IOException;
7. @SpringBootApplication
8. public class Application {
9. public static void main(String[] args) throws IOException {
10. SpringApplication.run(Application.class, args);
11. }
12. }

Подключение к серверу базы данных организуется посредством Hibernate, автоматически после настройки необходимых конфигураций:

1. spring.jpa.database=POSTGRESQL
2. spring.datasource.platform=postgres
3. spring.datasource.url=jdbc:postgresql://localhost:5432/postgres
4. spring.datasource.username=postgres

Все запросы по работе с базой данных вызываються специализированными интерфейсами JpaRepository. В следствии чего упрощается построение запросов, вызов процедур и фильтрация результатов. Пример интерфейсов с нестандартными пользовательскими запросами выглядит следующим образом:

1. import com.pyro.entities.Category;
2. import com.pyro.entities.Product;
3. import org.springframework.data.jpa.repository.JpaRepository;
4. import org.springframework.stereotype.Repository;
5. import java.util.List;
6. @Repository
7. public interface ProductRepository extends JpaRepository<Product, Long> {
8. Product findByName(String name);
9. List<Product> findByCategory(Category category);
10. List<Product> findByNameContainsIgnoreCase(String letter);
11. }

Результат выполненния представленных запросов представляет собой последовательность список элементов класса нужной сущности, который можно назначить в качестве источника данных для таблиц, и прочих элементов управления.

Функционал по работе с отчетами реализован в классе *Toxls*, в котором представлены методы для создания отчета.

## ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы была реализована база данных интернет-магазина ритуальных товаров, а также программный продукт, позволяющий пользователю взаимодействовать с базой данных. База данных была разработана спроектирована и разработана в среде Intellij IDEA 2019.3. Программный продукт реализован при помощи языков разметки HTML, CSS с использованием фреймворка Bootstrap, языка программирования Java с использованием фреймворка Spring Boot.

Приложение позволяет:

* Просматривать список товаров и услуг;
* Производить заказ товаров и услуг;
* Просматривать список заказов и список сообщений администрации;
* Отменить заказ продукта или услуги;
* Добавлять информацию о продуктах и услугах;
* Изменять информацию о продуктах и услугах;
* Удалять информацию о продуктах и услугах;
* Генерировать отчет о реализуемых продуктах и услугах

В процессе выполнения данной курсовой работы были закреплены навыки в программировании на Java с использованием Spring Framework, проектировании баз данных и реализации их в СУБД PostgreSQL, использования паттерна проектирования MVC.

## Список использованных источников

1. Data flow diagram - Wikipedia, the free encyclopedia / Многоязычная общедоступная свободно распространяемая энциклопедия, публикуемая в Интернете Википедия. Режим доступа: http://en.wikipedia.org/wiki/Data\_flow\_diagram
2. Entity-relationship model – Wikipedia, the free encyclopedia / Многоязычная общедоступная свободно распространяемая энциклопедия, публикуемая в Интернете Википедия. Режим доступа: http://en.wikipedia.org/wiki/Entity-relationship\_model
3. MySQL 5.0 Reference Manual [Электронный ресурс] / Официальный сайт MySQL. Документация по MySQL. Режим доступа:

http://dev.mysql.com/doc/refman/5.0/en/index.html

1. Документация по языку Java от компании Sun [Электронный ресурс]. – Элек-трон. текстовые дан. (270 Мб). – Sun Microsystems, Inc., 2006
2. Документация по MySQL Connector/J [Электронный ресурс]. – Элек-трон. текстовые дан. (3 Мб). – Sun Microsystems, Inc., 2008
3. Unified Modeling Language – Wikipedia, the free encyclopedia / Многоязычная общедоступная свободно распространяемая энциклопедия, публикуемая в Интернете Википедия. Режим доступа: http://en.wikipedia.org/wiki/Unified\_Modeling\_Language

## ПРИЛОЖЕНИЕ Б: СХЕМА РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ

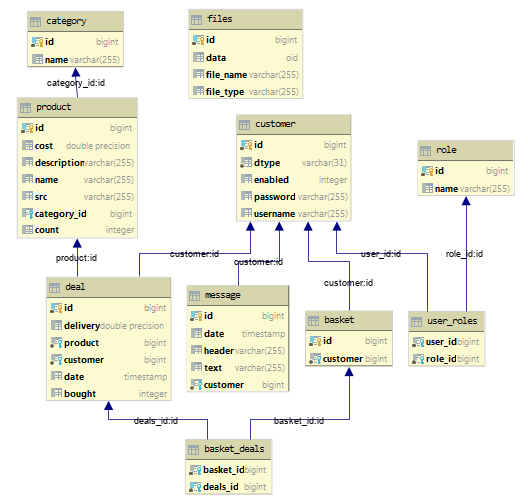


Рисунок Б.1 – Схема реляционной базы данных

## ПРИЛОЖЕНИЕ В: ГЛАВНАЯ И РАБОЧИЕ ФОРМЫ ПРИЛОЖЕНИЯ

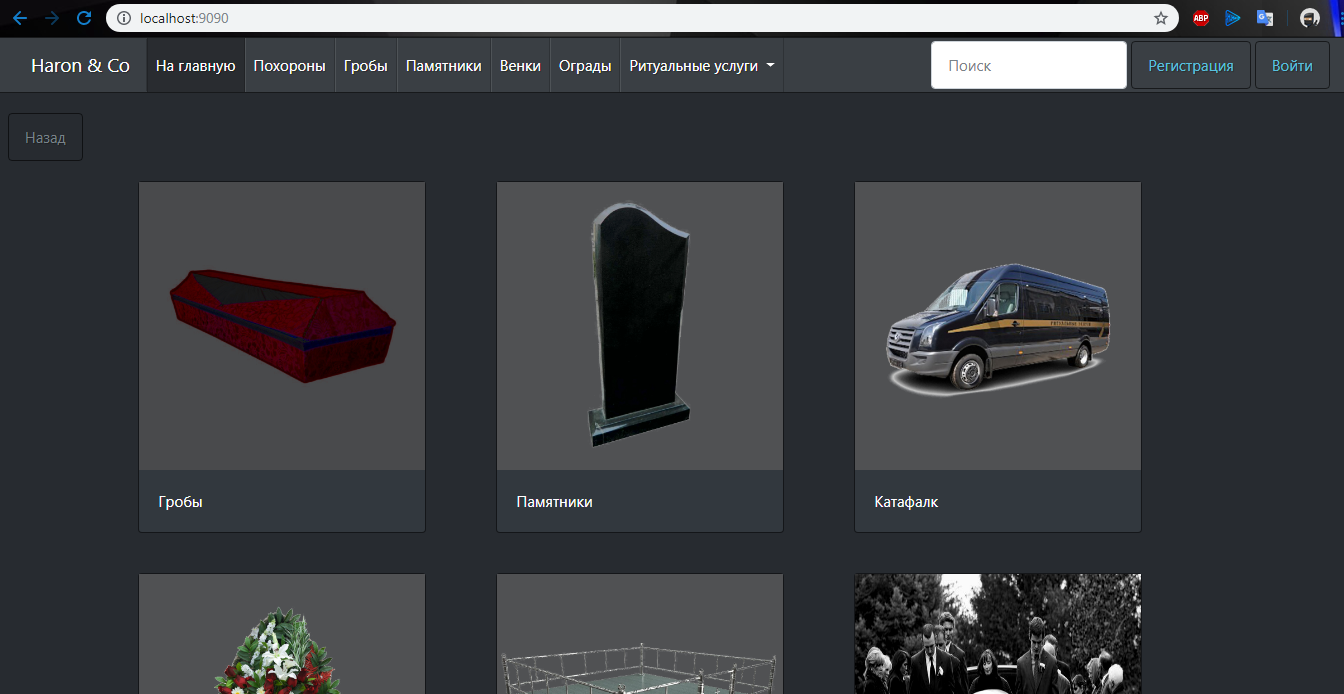


Рисунок В.1 – Главная страница.

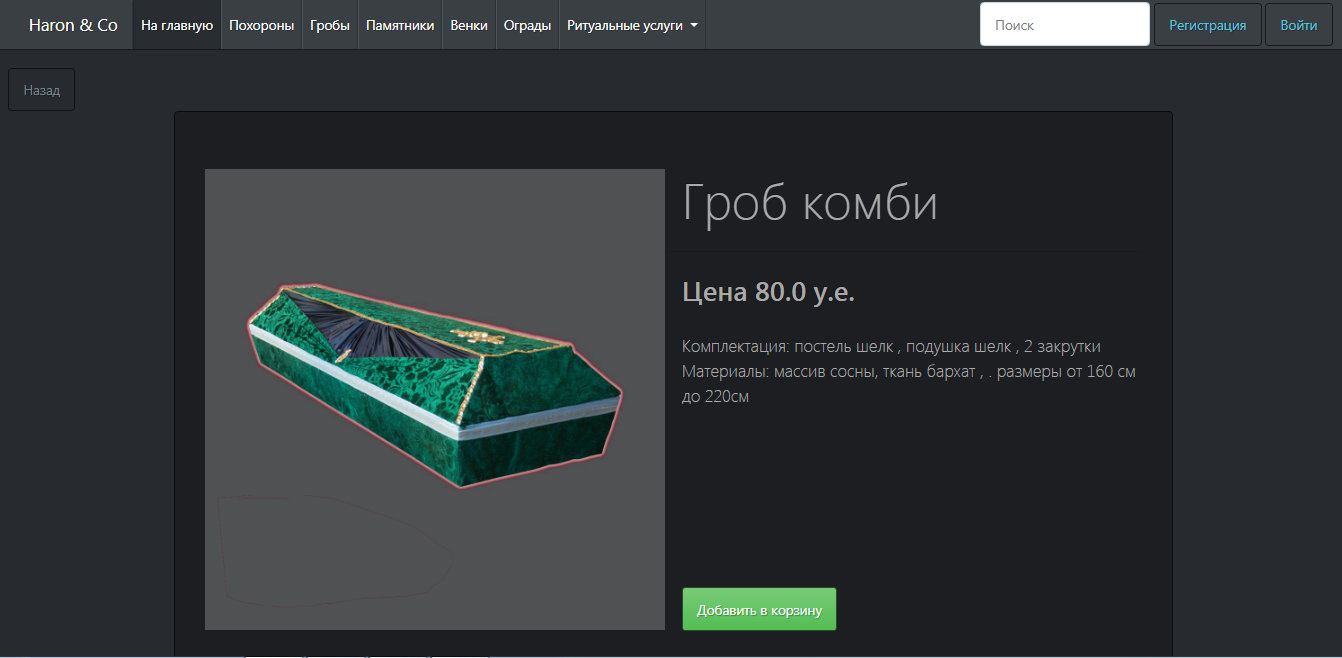


Рисунок В.2 – Страница продукта.

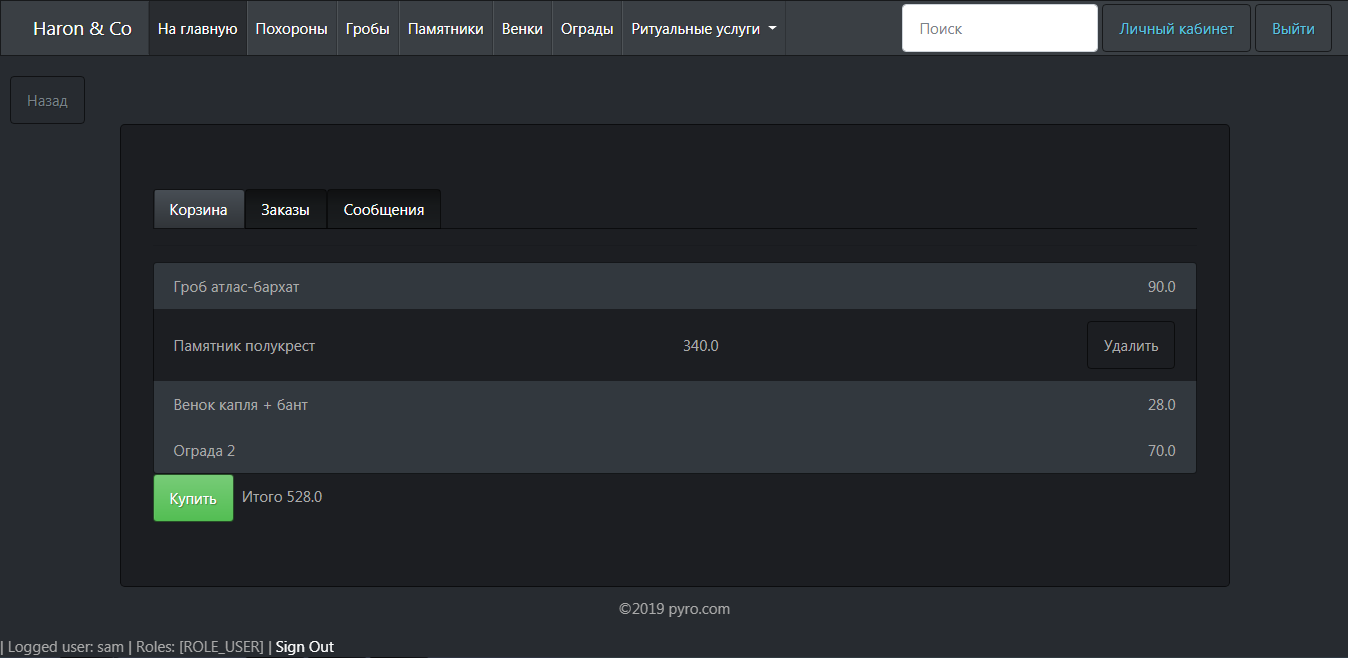


Рисунок В.3 – Страница личного кабинета.

## ПРИЛОЖЕНИЕ Г: ЛИСТИНГ ПРОГРАММЫ