СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc26534017)

[1. ПОСТРОЕНИЕ ИНФОЛОГИЧЕСКОЙ КОНЦЕПТУАЛЬНОЙ МОДЕЛИ 5](#_Toc26534018)

[1.1 Анализ предметной области и выявление необходимого набора сущностей 5](#_Toc26534019)

[1.2 Обоснование требуемого набора атрибутов для каждой сущности и выделение идентифицирующих атрибутов 6](#_Toc26534020)

[1.3 Определение связей между объектами 10](#_Toc26534021)

[1.4 Описание полученной модели на языке инфологического проектирования 12](#_Toc26534022)

[2. ПОСТРОЕНИЕ СХЕМЫ РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ 13](#_Toc26534023)

[2.1 Построение набора необходимых отношений базы данных 13](#_Toc26534024)

[2.2 Задание первичных и внешних ключей определенных отношений 14](#_Toc26534025)

[2.3 Третья нормальная форма 17](#_Toc26534026)

[2.4 Определение ограничений целостности для внешних ключей отношений и для отношений в целом 18](#_Toc26534027)

[2.5 Графическое представление связей между внешними и первичными ключами 20](#_Toc26534028)

[3. СОЗДАНИЕ СПРОЕКТИРОВАННОЙ БАЗЫ ДАННЫХ 21](#_Toc26534029)

[4 ЗАПИСЬ ВЫРАЖЕНИЙ УКАЗАННЫХ В ВАРИАНТЕ ЗАДАНИЯ ТИПОВ ЗАПРОСОВ НА ЯЗЫКЕ SQL 27](#_Toc26534030)

[5 ВЫБОР И ОСНОВАНИЕ СРЕДСТВ РАЗРАБОТКИ ПРИЛОЖЕНИЯ 32](#_Toc26534031)

[6 РЕАЛИЗАЦИЯ ЗАКОНЧЕННОГО ПРИЛОЖЕНИЯ, РАБОТАЮЩЕГО С СОЗДАННОЙ БАЗОЙ ДАННЫХ 35](#_Toc26534032)

[6.1 Разработка и построение интерфейса главной и рабочих форм 35](#_Toc26534033)

[6.2 Построение главного меню и кнопок панели инструментов 35](#_Toc26534034)

[6.3 Выполнение программного кода на Spring Boot 36](#_Toc26534035)

[ЗАКЛЮЧЕНИЕ 38](#_Toc26534036)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 39](#_Toc26534037)

[ПРИЛОЖЕНИЕ А: КОНЦЕПТУАЛЬНАЯ СХЕМА БД 40](#_Toc26534038)

[ПРИЛОЖЕНИЕ Б: СХЕМА РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ 41](#_Toc26534039)

[ПРИЛОЖЕНИЕ В: ГЛАВНАЯ И РАБОЧИЕ ФОРМЫ ПРИЛОЖЕНИЯ 42](#_Toc26534040)

[ПРИЛОЖЕНИЕ Г: ЛИСТИНГ ПРОГРАММЫ 44](#_Toc26534041)

## ВВЕДЕНИЕ

Мы живём во время широкого распространения интернета и современным людям не понять все аспекты жизни в 1980-2000г. В то время для получения заветной игры нужно было чуть больше, чем интернет и пластиковая карточка. Для получения игры нужно было как минимум выйти из дома и дойти до магазина. Многие скажут, что это очень долгий и сложный путь, но в этом было своё преимущество, а именно – мы ценили, то что у нас есть. А в данный момент, когда интернет слишком распространён, мы не ценим игры. Чтобы купить игру сейчас нужно просто оплатить игру в интернет-магазине карточкой или вообще купить подписку на сервис, который будет предоставлять вами игры за ежемесячную плату. Игр в таком случае вы получите очень много, за цену меньше, чем за целую игру. Можно играть во всё что угодно и не ждать пока носитель с нужной игрой привезут в магазин или не привезут, если игра предназначена для другого рынка. Если так много преимуществ, то для чего вообще нужен магазин «РЕТРО ИГОРЫ»? – А нужен он исключительно для ностальгии по старым временам, по былым мечтам и по беззаботному прошлому.

Как бы настоящее ни было продвинутое и хорошее, в прошлом было что-то хорошее о чём можно было бы вспомнить.

## ПОСТРОЕНИЕ ИНФОЛОГИЧЕСКОЙ КОНЦЕПТУАЛЬНОЙ МОДЕЛИ

1.1 Анализ предметной области и выявление необходимого набора сущностей

Знакомство покупателей с магазином начинается с витрины. На ней размещены все игры для всех платформ.

Для быстрого выбора необходимой игры все игры разделены по жанрам и платформам. Если покупатель приходит в магазин, за игрой, то скорее всего у него есть какая-то игровая платформа.

Так как на сайте организована аренда игр, а не продажа, покупателю нужно выбрать количество дней для аренды. Покупатель может либо оплатить игры и прийти в магазин и забрать их, либо заказать доставку.

Работа администратора же заключается в просмотре заказов всех пользователей, добавлении игр, платформ, жанров.

Если товар был удалён из магазина, а у кого-то он был в корзине, то он удаляется из корзины.

В ходе анализа знаний и разработке базы данных были выявлены следующие основные сущности:

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 проката игр. Для определения первичного набора сущностей будет проведён анализ технического задания и предметной области.

Для данной информационной системы предусмотрено два вида полномочий – администратор, который добавляет и удаляет игры, платформы и просматривает заказы, но и покупатель, который может добавить товар в корзину, удалить его из корзины, если он там уже есть, получить купленный продукт. Так как у нас есть несколько видов полномочий, то нам нужна таблица Roles, которая хранит роли.

Так как для заказа и манипуляции с данными нужно зарегистрироваться и затем войти, нам нужна таблица с Users, которая будет хранить e-mail, пароль, количество заказов, адрес.

У нас присутствуют игровые платформы, для них тоже нужна таблица Platforms. Она содержит модель процессора, описание, поколение, производитель, название, количество проданных штук, дата начала продаж, история, изображение.

Мы предлагаем купить, поэтому нужна таблица с описанием товара (Products). В ней мы храним название, описание, полную цены, цену за день, количество, возрастной лимит, жанр, изображение, язык, платформа, издатель.

При добавлении в корзину товара создаётся запрос (Requests). Он хранит данные о продукте и выбранное количество дней. В данной таблице есть, ссылка на товар, дата, количество дней, цена за один день, цена за выбранное количество дней, ссылка на корзину, к которой этот заказ относится, ссылка на финальный заказ, к которому относится данная заявка (если товар не удалят из корзины).

Корзина(Basket) для хранения текущих заявок покупателя. Она хранит только ссылку на все заявки и цену за все товары в корзине.

Заказ (Final\_order) нужен для дублирования данных из корзины. Когда заказ полностью сформирован, то мы удаляем корзину и переносим данные из корзины и добавляем данные о доставке. Таблица хранит ссылку на заявки, дату, цену, ссылку на доставку.

Доставка (Delivery) хранит данные о доставке, ссылку на заказ, дату, адрес, тип доставки.

## 1.2 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** |
| user | id | PK | Идентификационный номер таблицы "user". |
| dtype |  | Служебное поле |
| enabled |  | Состояние учётной записи. |
| password |  | Пароль учетной записи. |
| username |  | Login для входа. |
| address\_id | FK | Идентификационный номер таблицы "address". |
| roles | id | PK | Идентификационный номер таблицы "roles". |
| name |  | Имя уровня доступа. |
| address | id | PK | Идентификационный номер таблицы "address". |
|  | city |  | Город пользователя. |
|  | street |  | Улица пользователя. |
|  | flat\_number |  | Номер квартиры пользователя. |
| platforms | id | PK | Идентификационный номер таблицы " platforms". |
|  | cpu |  | Информация о процессоре. |
|  | description |  | Описание игровой платформы. |
|  | generation |  | Поколение. |
|  | manufacturer |  | Производитель |
|  | name |  | Название игровой платформы. |
|  | pieces\_sold |  | Количество проданных штук. |
|  | release\_date |  | Дата выхода. |
|  | story |  | История о платформе. |
|  | imagest\_id | FK | Идентификационный номер таблицы "imagest". |
| imagest | id | PK | Идентификационный номер таблицы "imagest". |
|  | content\_type |  | Полный тип изображения. |
|  | data |  | Само изображение в Base64 |
|  | extension |  | Расширение изображения. |
|  | name |  | Название изображения. |

Продолжение таблицы 1.1

|  |  |  |  |
| --- | --- | --- | --- |
| **Таблица** | **Поле** | **Ключ** | **Описание** |
| products | id | PK | Идентификационный номер таблицы "specialty". |
| Name |  | Название продукта. |
| description |  | Описание |
| full\_price |  | Полная цена |
| one\_day\_price |  | Цена за день |
| idd | FK | Для изменения платформы у игры, если она удалена. |
| quantity |  | Количество |
| year\_of\_issue |  | Год выпуска |
| age\_limits\_id | FK | Идентификационный номер таблицы "age\_limits". |
| genres\_id | FK | Идентификационный номер таблицы "genres". |
| imagesp\_id | FK | Идентификационный номер таблицы "imagesp". |
| languages\_id | FK | Идентификационный номер таблицы "languages". |
| platforms\_id | FK | Идентификационный номер таблицы "platforms". |
| publishers\_id | FK | Идентификационный номер таблицы "publishers". |
| languages | id | PK | Идентификационный номер таблицы "languages". |
| name |  | Язык |
| publishers | id | PK | Идентификационный номер таблицы "publishers". |
| name |  | Издатель |
| age\_limits | id | PK | Идентификационный номер таблицы "age\_limits". |
| year |  | Возраст |

Продолжение таблицы 1.1

|  |  |  |  |
| --- | --- | --- | --- |
| **Таблица** | **Поле** | **Ключ** | **Описание** |
| genres | id | PK | Идентификационный номер таблицы "genres". |
| name |  | Жанр игры. |
| imagesg\_id | FK | Идентификационный номер таблицы "imagesg". |
|  |  |  |
| imagesg | id | PK | Идентификационный номер таблицы "imagesg". |
| content\_type |  | Полный тип изображения. |
| data |  | Само изображение в Base64 |
| extension |  | Расширение изображения. |
| name |  | Название изображения. |
| imagesp | id | PK | Идентификационный номер таблицы " imagesp". |
| content\_type |  | Полный тип изображения. |
| data |  | Само изображение в Base64 |
| extension |  | Расширение изображения. |
| name |  | Название изображения. |
| requests | id | PK | Идентификационный номер таблицы "genres". |
| date |  | Дата запроса. |
| number\_of\_days |  | Количество дней аренды. |
| price |  | Цена за день. |
| basket\_id | FK | Идентификационный номер таблицы "basket". |
| final\_order\_id | FK | Идентификационный номер таблицы "final\_order". |
| products\_id | FK | Идентификационный номер таблицы "products". |

Окончание таблицы 1.1

|  |  |  |  |
| --- | --- | --- | --- |
| **Таблица** | **Поле** | **Ключ** | **Описание** |
| basket | id | PK | Идентификационный номер таблицы "basket". |
| final\_price |  | Общая стоимость. |
| final\_order | id | PK | Идентификационный номер таблицы "final\_order". |
| final\_price |  | Общая стоимость. |
| data |  | Дата оформления заказа. |
| delivery\_id | FK | Идентификационный номер таблицы "delivery". |
| delivery | id | PK | Идентификационный номер таблицы "delivery". |
| date |  | Дата доставки. |
| addressd\_id | FK | Идентификационный номер таблицы "addressd". |
| type\_of\_delivery\_id | FK | Идентификационный номер таблицы "type\_of\_delivery". |
| type\_of\_delivery | id | PK | Идентификационный номер таблицы "type\_of\_delivery". |
| type |  | Тип доставки |
| addressd | id | PK | Идентификационный номер таблицы "address". |
| city |  | Город пользователя. |
| street |  | Улица пользователя. |
| flat\_number |  | Номер квартиры пользователя. |

* 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. User – Roles: (М:1).
2. User – Address: (1:1).
3. Delivery – Type\_of\_delivery: (М:1).
4. Delivery – Addressd: (М:1).
5. Delivery – Final\_order: (1:1).
6. Final\_order – Requests: (1: М).
7. Requests – Basket: (М:1).
8. Requests – Products: (М:1).
9. Platforms – Imagest: (1:1).
10. Genres – Imagesg: (1:1).
11. Product – Publishers: (М:1).
12. Product – Platforms: (M:1).
13. Product – Genres: (M:1).
14. Product – Age\_limits: (M:1).
15. Product – Languages: (M:1).
16. Product – Imagesp: (1:1).

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.

## 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 «Пункт аренды игр», построенной с помощью IDEF1X. На основе полученной концептуальной модели можно определить набор необходимых отношений базы данных [1]. На рисунке 2.1 представлены отношения для базы данных.

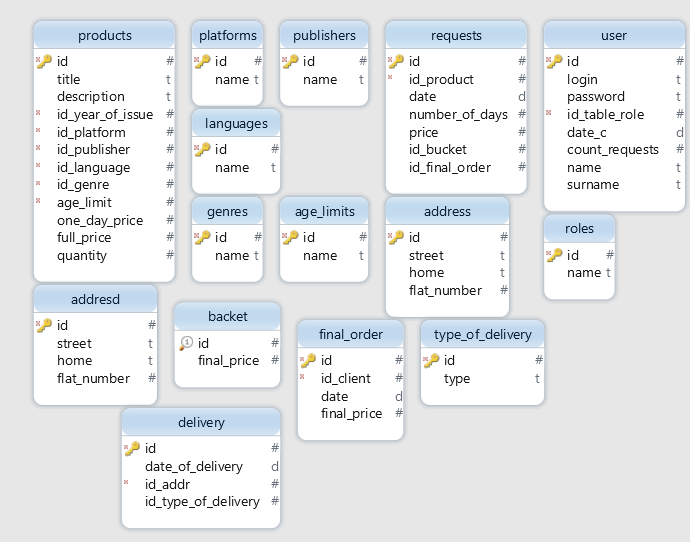


Рисунок 2.1 – Набор необходимых отношений базы данных

## 

## 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** |
| user | id | PK |
| dtype |  |
| enabled |  |
| password |  |
| username |  |
| count\_requests |  |
| address\_id | FK |
| roles | id | PK |
| name |  |
| address | id | PK |
| city |  |
| street |  |
| flat\_number |  |
| platforms | id | PK |
| cpu |  |
| description |  |
| generation |  |

Продолжение таблицы 2.1

|  |  |  |
| --- | --- | --- |
| **Таблица** | **Поле** | **Ключ** |
| platforms | manufacturer |  |
| name |  |
| pieces\_sold |  |
| release\_date |  |
| story |  |
| imagest\_id | FK |
| imagest | id | PK |
| content\_type |  |
| data |  |
| extension |  |
| name |  |
| products | id | PK |
| Name |  |
| description |  |
| full\_price |  |
| one\_day\_price |  |
| idd | FK |
| quantity |  |
| year\_of\_issue |  |
| age\_limits\_id | FK |
| genres\_id | FK |
| imagesp\_id | FK |
| languages\_id | FK |
| platforms\_id | FK |
| publishers\_id | FK |
| languages | id | PK |
| name |  |
| publishers | id | PK |
| name |  |
| age\_limits | id | PK |
| year |  |

Продолжение таблицы 2.1

|  |  |  |
| --- | --- | --- |
| **Таблица** | **Поле** | **Ключ** |
| genres | id | PK |
| name |  |
| imagesg\_id | FK |
|  |  |
| imagesg | id | PK |
| content\_type |  |
| data |  |
| extension |  |
| name |  |
| imagesp | id | PK |
| content\_type |  |
| data |  |
| extension |  |
| name |  |
| requests | id | PK |
|  | date |  |
|  | number\_of\_days |  |
|  | price |  |
|  | basket\_id | FK |
|  | final\_order\_id | FK |
|  | products\_id | FK |
| basket | id | PK |
|  | final\_price |  |
| final\_order | id | PK |
|  | final\_price |  |
|  | data |  |
|  | delivery\_id | FK |
| delivery | id | PK |
|  | date |  |
|  | addressd\_id | FK |
|  | type\_of\_delivery\_id | FK |

Окончание таблицы 2.1

|  |  |  |
| --- | --- | --- |
| **Таблица** | **Поле** | **Ключ** |
| requests | id | PK |
| date |  |
| number\_of\_days |  |
| price |  |
| basket\_id | FK |
| final\_order\_id | FK |
| products\_id | FK |
| basket | id | PK |
| final\_price |  |
| final\_order | id | PK |
| final\_price |  |
| data |  |
| delivery\_id | FK |
| delivery | id | PK |
| date |  |
| addressd\_id | FK |
| type\_of\_delivery\_id | FK |
| Type\_of\_delivery | id | PK |
| type |  |
| addressd | id | PK |
| city |  |
| street |  |
| flat\_number |  |

* 1. 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 [отношения](http://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BD%D0%BE%D1%88%D0%B5%D0%BD%D0%B8%D0%B5_(%D1%80%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C)) в [реляционной модели данных](http://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), 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].

Таким образом, отношения БД находятся в 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.

Для хранения информации студенте необходимо обязательно заполнить такие данные как, информация об образовании, паспортные данные и семейное положение, для этого между таблицами Students и Education, Students и MaritalStatus, Students и Passports были установлены связи один-к-одному.

Т.к. студент не обязательно может иметь данные о работе, либо фотографию, то контроль ограничения целостности для отношений между таблицами Students и Work, Students и StudentPhoto отключен.

Так как для оформляемого документа может отсутствовать дополнительная информация, то для отношения между таблицами Documents и AdditionalDocumetnData ограничение контроля целостности отключено, а кардинальность связи установлена один-к одному.

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 11. PostgreSQL предоставляет множество различных возможностей, достаточно надежна и имеет хорошие характеристики по производительности. Она работает практически на всех UNIX-платформах, включая UNIX-подобные системы, такие как FreeBSD и Linux.

Кроме того, PostgreSQL свободно распространяется и имеет открытый исходный код. PostgreSQL выгодно отличается от многих других СУБД.

Произведем описание структуру каждой из таблиц с описанием типа полей.

Таблица *User* хранит данные о пользователе, которые служат для авторизации пользователя. Её структура приведена в таблице 3.1.

Таблица 3.1 – Характеристика атрибутов таблицы Users.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| id | bigint | Идентификационный номер таблицы "user". |
| dtype | varchar(31) | Служебное поле |
| enabled | integer | Состояние учётной записи. |
| password | varchar(255) | Пароль учетной записи. |
| username | varchar(255) | Login для входа. |
| address\_id | varchar(255) | Идентификационный номер таблицы "address". |

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

Таблица 3.2 – Характеристика атрибутов таблицы Access.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| id | bigint | Идентификационный номер таблицы "roles". |
| name | varchar(255) | Имя уровня доступа. |

Таблица Address содержит адреса пользователей. Её структура приведена в таблице 3.3.

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

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| id | bigint | Идентификационный номер таблицы "address". |
| city | varchar(255) | Город пользователя. |
| street | varchar(255) | Улица пользователя. |
| flat\_number | varchar(255) | Номер квартиры пользователя. |

Таблица *Platforms* содержит сведения о враче. Её структура приведена в таблице 3.4.

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

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| id | bigint | Идентификационный номер таблицы " platforms". |
| cpu | varchar(255) | Информация о процессоре. |
| description | text | Описание игровой платформы. |
| generation | varchar(255) | Поколение. |
| manufacturer | varchar(255) | Производитель |
| name | varchar(255) | Название игровой платформы. |
| pieces\_sold | varchar(255) | Количество проданных штук. |
| release\_date | varchar(255) | Дата выхода. |
| story | text | История о платформе. |
| imagest\_id | bigint | Идентификационный номер таблицы "imagest". |

Таблица *Imagest* содержит список изображений относящихся к платформам. Её структура приведена в таблице 3.5.

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

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| id | bigint | Идентификационный номер таблицы "imagest". |
| content\_type | varchar(255) | Полный тип изображения. |
| data | text | Само изображение в Base64 |
| extension | varchar(255) | Расширение изображения. |
| name | varchar(255) | Название изображения. |

Таблица *Products* содержит список товаров. Её структура приведена в таблице 3.6.

**Таблица 3.6 –** Характеристика атрибутов таблицы Products.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| id | bigint | Идентификационный номер таблицы "products". |
| title | varchar(255) | Название продукта. |
| description | text | Описание |
| full\_price | double precision | Полная цена |
| one\_day\_price | double precision | Цена за день |
| idd | bigint | Для изменения платформы у игры, если она удалена. |
| quantity | integer | Количество |
| year\_of\_issue | integer | Год выпуска |
| age\_limits\_id | bigint | Идентификационный номер таблицы "age\_limits". |
| genres\_id | bigint | Идентификационный номер таблицы "genres". |
| imagesp\_id | bigint | Идентификационный номер таблицы "imagesp". |
| languages\_id | bigint | Идентификационный номер таблицы "languages". |
| platforms\_id | bigint | Идентификационный номер таблицы "platforms". |
| publishers\_id | bigint | Идентификационный номер таблицы "publishers". |

Таблица *Languages* содержит список всех доступных языков. Её структура приведена в таблице 3.7.

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

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| id | bigint | Идентификационный номер таблицы "languages". |
| name | varchar(255) | Язык игры. |

Таблица *Publishers* содержит список всех доступных издательств. Её структура приведена в таблице 3.8.

**Таблица 3.8 –** Характеристика атрибутов таблицы *Publishers*.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| id | bigint | Идентификационный номер таблицы "publishers". |
| name | varchar(255) | Издатель. |

Таблица *Age\_limits* содержит список возрастных ограничений для игр. Её структура приведена в таблице 3.9.

**Таблица 3.9** – Характеристика атрибутов таблицы *Age\_limits*.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| id | bigint | Идентификационный номер таблицы "age\_limits". |
| year | integer | Возраст. |

Таблица *Genres* содержит список жанров игр. Её структура приведена в таблице 3.10.

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

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| id | bigint | Идентификационный номер таблицы "genres". |
| name | varchar(255) | Жанр игры. |
| imagesg\_id | bigin | Идентификационный номер таблицы "imagesg". |

Таблица *Imagesg* содержит список изображений, относящихся к жанрам. Её структура приведена в таблице 3.11.

**Таблица 3.11 –** Характеристика атрибутов таблицы Imagesg.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| id | bigint | Идентификационный номер таблицы "imagesg". |
| content\_type | varchar(255) | Полный тип изображения. |
| data | text | Само изображение в Base64 |
| extension | varchar(255) | Расширение изображения. |
| name | varchar(255) | Название изображения. |

Таблица *Imagesp* содержит список изображений, относящихся к продуктам. Её структура приведена в таблице 3.12.

**Таблица 3.12 –** Характеристика атрибутов таблицы Imagesp.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| id | bigint | Идентификационный номер таблицы "imagesp". |
| content\_type | varchar(255) | Полный тип изображения. |
| data | text | Само изображение в Base64 |
| extension | varchar(255) | Расширение изображения. |
| name | varchar(255) | Название изображения. |

Таблица *Requests* содержит список заявок, которые создаются при добавлении игры в корзину. Её структура приведена в таблице 3.13.

**Таблица 3.13 –** Характеристика атрибутов таблицы *Requests*.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| id | bigint | Идентификационный номер таблицы "genres". |
| date | varchar(255) | Дата запроса. |
| number\_of\_days | integer | Количество дней аренды. |
| price | double precision | Цена за день. |
| basket\_id | bigint | Идентификационный номер таблицы "basket". |
| final\_order\_id | bigint | Идентификационный номер таблицы "final\_order". |
| products\_id | bigint | Идентификационный номер таблицы "products". |

Таблица *Basket* содержит продукты, добавленные в корзину. Её структура приведена в таблице 3.14.

**Таблица 3.14 –** Характеристика атрибутов таблицы *Basket*.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| id | bigint | Идентификационный номер таблицы "basket". |
| final\_price | double precision | Общая стоимость. |

Таблица *Final\_order* содержит список оформленных заказов. Её структура приведена в таблице 3.15.

**Таблица 3.15 –** Характеристика атрибутов таблицы *Final\_order*.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| id | bigint | Идентификационный номер таблицы "final\_order". |
| final\_price | double precision | Общая стоимость. |
| data | varchar(255) | Дата оформления заказа. |
| delivery\_id | bigint | Идентификационный номер таблицы "delivery". |

Таблица *Delivery* содержит список данных о доставках. Её структура приведена в таблице 3.16.

**Таблица 3.16 –** Характеристика атрибутов таблицы *Delivery*.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| id | bigint | Идентификационный номер таблицы "delivery". |
| date | varchar(255) | Дата доставки. |
| addressd\_id | bigint | Идентификационный номер таблицы "addressd". |
| type\_of\_delivery\_id | bigint | Идентификационный номер таблицы "type\_of\_delivery". |

Таблица *Type\_of\_Delivery* содержит список видов доставок. Её структура приведена в таблице 3.17.

**Таблица 3.17 –** Характеристика атрибутов таблицы *Type\_of\_Delivery*.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| id | bigint | Идентификационный номер таблицы "type\_of\_delivery". |
| type | varchar(255) | Тип доставки |

Таблица *Addressd* содержит адрес доставки, который формируется при оформлении заказа. Её структура приведена в таблице 3.18.

**Таблица 3.18 –** Характеристика атрибутов таблицы *Addressd*.

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| id | bigint | Идентификационный номер таблицы "address". |
| city | varchar(255) | Город пользователя. |
| street | varchar(255) | Улица пользователя. |
| flat\_number | varchar(255) | Номер квартиры пользователя. |

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

1. **Запрос возвращающий один продукт и все его данные.**

select products0\_.id as id1\_12\_1\_, products0\_.age\_limits\_id as age\_limi9\_12\_1\_, products0\_.description as descript2\_12\_1\_, products0\_.full\_price as full\_pri3\_12\_1\_, products0\_.genres\_id as genres\_10\_12\_1\_, products0\_.idd as idd4\_12\_1\_, products0\_.imagesp\_id as imagesp11\_12\_1\_, products0\_.languages\_id as languag12\_12\_1\_, products0\_.one\_day\_price as one\_day\_5\_12\_1\_, products0\_.platforms\_id as platfor13\_12\_1\_, products0\_.publishers\_id as publish14\_12\_1\_, products0\_.quantity as quantity6\_12\_1\_, products0\_.title as title7\_12\_1\_, products0\_.year\_of\_issue as year\_of\_8\_12\_1\_, imagesp1\_.id as id1\_8\_0\_, imagesp1\_.content\_type as content\_2\_8\_0\_, imagesp1\_.data as data3\_8\_0\_, imagesp1\_.extension as extensio4\_8\_0\_, imagesp1\_.name as name5\_8\_0\_ from cursovaya.products products0\_ left outer join cursovaya.imagesp imagesp1\_ on products0\_.imagesp\_id=imagesp1\_.id where products0\_.imagesp\_id=?

1. **Выборка определенного жанра.**

select genres0\_.id as id1\_6\_1\_, genres0\_.imagesg\_id as imagesg\_3\_6\_1\_, genres0\_.name as name2\_6\_1\_, imagesg1\_.id as id1\_7\_0\_, imagesg1\_.content\_type as content\_2\_7\_0\_, imagesg1\_.data as data3\_7\_0\_, imagesg1\_.extension as extensio4\_7\_0\_, imagesg1\_.name as name5\_7\_0\_ from cursovaya.genres genres0\_ left outer join cursovaya.imagesg imagesg1\_ on genres0\_.imagesg\_id=imagesg1\_.id where genres0\_.imagesg\_id=?

1. **Получение данных пользователя.**

select user0\_.id as id2\_19\_, user0\_.address\_id as address\_7\_19\_, user0\_.count\_reqests as count\_re3\_19\_, user0\_.enabled as enabled4\_19\_, user0\_.password as password5\_19\_, user0\_.username as username6\_19\_ from cursovaya.user user0\_ where user0\_.username=?

1. **Получение адреса пользователя.**

select address0\_.id as id1\_0\_, address0\_.city as city2\_0\_, address0\_.flat\_number as flat\_num3\_0\_, address0\_.street as street4\_0\_ from cursovaya.address address0\_

1. **Получение данных о доставке.**

select delivery0\_.id as id1\_4\_0\_, delivery0\_.addressd\_id as addressd3\_4\_0\_, delivery0\_.date as date2\_4\_0\_, delivery0\_.type\_of\_delivery\_id as type\_of\_4\_4\_0\_, addressd1\_.id as id1\_1\_1\_, addressd1\_.city as city2\_1\_1\_, addressd1\_.flat\_number as flat\_num3\_1\_1\_, addressd1\_.street as street4\_1\_1\_, finalorder2\_.id as id1\_5\_2\_, finalorder2\_.date as date2\_5\_2\_, finalorder2\_.delivery\_id as delivery4\_5\_2\_, finalorder2\_.final\_price as final\_pr3\_5\_2\_, finalorder2\_.user\_id as user\_id5\_5\_2\_ from cursovaya.delivery delivery0\_ left outer join cursovaya.addressd addressd1\_ on delivery0\_.addressd\_id=addressd1\_.id left outer join cursovaya.final\_order finalorder2\_ on delivery0\_.id=finalorder2\_.delivery\_id where delivery0\_.id=?

1. **Получение данных о заказе.**

select finalorder0\_.id as id1\_5\_2\_, finalorder0\_.date, finalorder0\_.delivery\_id, finalorder0\_.final\_price, finalorder0\_.user\_id, delivery1\_.id as id1\_4\_0\_, delivery1\_.addressd\_id as addressd3\_4\_0\_, delivery1\_.date as date2\_4\_0\_, delivery1\_.type\_of\_delivery\_id as type\_of\_4\_4\_0\_, addressd2\_.id as id1\_1\_1\_, addressd2\_.city as city2\_1\_1\_, addressd2\_.flat\_number as flat\_num3\_1\_1\_, addressd2\_.street as street4\_1\_1\_ from cursovaya.final\_order finalorder0\_ left outer join cursovaya.delivery delivery1\_ on finalorder0\_.delivery\_id=delivery1\_.id left outer join cursovaya.addressd addressd2\_ on delivery1\_.addressd\_id=addressd2\_.id where finalorder0\_.delivery\_id=?

1. **Получение данных о платформе.**

select platforms0\_.id as id1\_11\_1\_, platforms0\_.cpu as cpu2\_11\_1\_, platforms0\_.description as descript3\_11\_1\_, platforms0\_.generation as generati4\_11\_1\_, platforms0\_.imagest\_id as imagest10\_11\_1\_, platforms0\_.manufacturer as manufact5\_11\_1\_, platforms0\_.name as name6\_11\_1\_, platforms0\_.pieces\_sold as pieces\_s7\_11\_1\_, platforms0\_.relase\_date as relase\_d8\_11\_1\_, platforms0\_.story as story9\_11\_1\_, imagest1\_.id as id1\_9\_0\_, imagest1\_.content\_type as content\_2\_9\_0\_, imagest1\_.data as data3\_9\_0\_, imagest1\_.extension as extensio4\_9\_0\_, imagest1\_.name as name5\_9\_0\_ from cursovaya.platforms platforms0\_ left outer join cursovaya.imagest imagest1\_ on platforms0\_.imagest\_id=imagest1\_.id where platforms0\_.imagest\_id=?

1. **Вставка запроса товара.**

insert into cursovaya.requests (basket\_id, date, final\_order\_id, number\_of\_days, price, products\_id) values (?, ?, ?, ?, ?, ?)

1. **Получение данных о корзине.**

select basket0\_.id as id1\_3\_0\_, basket0\_.final\_price as final\_pr2\_3\_0\_ from cursovaya.basket basket0\_ where basket0\_.id=?

1. **Вставка данных в адрес доставки.**

insert into cursovaya.addressd (city, flat\_number, street, id) values (?, ?, ?, ?)

1. **Обновление данных о заказе.**

update cursovaya.final\_order set date=?, delivery\_id=?, final\_price=?, user\_id=? where id=?

Также было реализовано несколько триггеров.

1. **Изменение количества дней заказов у покупателя, при оформлении заказа.**

CREATE TRIGGER t\_user2

AFTER INSERT

ON cursovaya.final\_order

FOR EACH ROW

EXECUTE PROCEDURE add\_to\_log2();

CREATE OR REPLACE FUNCTION add\_to\_log2() RETURNS TRIGGER AS

$$

DECLARE

mstr varchar(30);

cn int;

cn2 int;

cnt int;

curcnt int;

BEGIN

cn := cursovaya.requests.basket\_id from cursovaya.requests order by id desc limit 1;

cn2 := cursovaya.requests.final\_order\_id from cursovaya.requests order by id desc limit 1;

cnt := sum(cursovaya.requests.number\_of\_days) from cursovaya.requests where cursovaya.requests.basket\_id = cn;

if (select usr.count\_reqests from cursovaya."user" as usr where usr.id = cn) is null then

curcnt = 0;

else

curcnt := usr.count\_reqests from cursovaya."user" as usr where usr.id = cn;

end if;

cnt = cnt + curcnt;

update cursovaya."user"

set count\_reqests=cnt

where cursovaya."user".id = cn;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

**2.Изменение платформы у игры на «Другая», если данная платформа установлена, при удалении платформы.**

CREATE TRIGGER dp\_platform

before delete or update

ON cursovaya.platforms

FOR EACH ROW

EXECUTE PROCEDURE get\_all\_foo();

CREATE OR REPLACE FUNCTION get\_all\_foo() RETURNS TRIGGER AS $$

DECLARE

r cursovaya.products%rowtype;

BEGIN

FOR r IN

SELECT \* FROM cursovaya.products WHERE id > 0

LOOP

if r.platforms\_id = old.id then

update cursovaya.products set platforms\_id = 9 where id = r.id;

update cursovaya.products set idd = 9 where id = r.id;

end if;

END LOOP;

RETURN old;

END;

$$ LANGUAGE plpgsql;

## 5 ВЫБОР И ОСНОВАНИЕ СРЕДСТВ РАЗРАБОТКИ ПРИЛОЖЕНИЯ

Для реализации задачи по разработке информационной системы бюро проката игр была выбрана СУБД PostgreSQL. PostgreSQL предоставляет множество различных возможностей, достаточно надежна и имеет хорошие характеристики по производительности [3]. Она работает практически на всех UNIX-платформах, включая UNIX-подобные системы, такие как FreeBSD и Linux. Ее можно применять на Windows NT Server и Windows 2000 Server, а для разработки годятся даже такие системы Microsoft для рабочих станций, как ME. Кроме того, PostgreSQL свободно распространяется и имеет открытый исходный код. PostgreSQL выгодно отличается от многих других СУБД. Она обладает практически всеми возможностями, которые есть в других базах данных (коммерческих или Open Source), а также некоторыми дополнительными.

Приведем перечень функциональных возможностей PostgreSQL:

* Транзакции
* Вложенные запросы
* Представления
* Ссылочная целостность - внешние ключи
* Сложные блокировки
* Типы, определяемые пользователем
* Наследственность
* Правила
* Проверка совместимости версий

Для создания приложения используется Intellij IDEA Ultimate 2019.3, а языком написания приложения был выбран – Java. Данные средства служат для написания мощных приложений. Главные достоинства Java:

* Наибольшая среди всех языков программирования степень переносимости программ.
* Мощные стандартные библиотеки.
* Встроенная поддержка работы в сетях (как локальных, так и Internet).

Подключение приложения к базе данных будет осуществляться через драйвер JDBC PostgreSQL Connector. Для тривиальных операций по сохранению Java объектов в БД и наоборот, созданию Java объектов по данным из БД.

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.

**Bootstrap — это открытый и бесплатный HTML, CSS и JS фреймворк, который используется веб-разработчиками для быстрой вёрстки адаптивных дизайнов сайтов и веб-приложений.**

Фреймворк Bootstrap используется по всему миру не только независимыми разработчиками, но иногда и целыми компаниями.

Основная область его применения – это разработка frontend [4].

Основные преимущества Bootstrap:

* Экономия времени — вы экономите усилия потому что используете уже готовые классы и дизайн.
* Адаптивность, высокая скорость и оптимизация, стандартизация интерфейсов — динамичные макеты **Bootstrap** качественно отображаются на самых разных устройствах без необходимости внесения изменений в разметку.
* Простота и открытость — использовать Бутстрап настолько просто, что с ним справляются начинающие веб-разработчики, а открытый исходный код позволяет самому участвовать в разработке, модифицировать под свои нужды или просто пользоваться хорошим бесплатным решением.

Также, в Bootstrap используется динамический язык стилей LESS, которые расширяет возможности CSS: разработчики могут управлять цветами, создавать вложенные колонки и переменные.

[5]

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

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

Главная страница представляет собой HTML страницу с использованием шаблонизатора Thymeleaf. Все повторяющиеся элементы были вынесены в фрагменты, которые встраиваются в нужное место на странице. Для стилизации страницы использована библиотека Bootstrap. Она позволяет выбрать стили элементов в одном виде, и они не будут сильно отличать друг от друга. Все элементы были размещены в «нужных» местах, где пользователь ожидал бы их найти.

Основной упор вёрстки был сделан на удобство использования, а потом уже на красоту. Даже при условии использования библиотеке стилей нужно было использовать свои стили для выравнивания элементов. Так же для некоторых действий был использован JavaScript. Он позволяет обновлять данные на странице полностью её не перерисовывая. Он в основном используется для изменения суммы заказа продукта.

Скриншоты главной и некоторых диалоговых окон представлены в приложении C.

6.2 **Построение главного меню и кнопок панели инструментов**

Верхнее навигационное меню представлено ссылками на жанры, список платформ, информацию о сайте, информацию о доставке. Левое меню представлено ссылками на доступные консоли, с которых можно перейти на нужную и прочитать о ней информацию. В зависимости от прав пользователя доступные разные пункты в меню. Для администратора добавлены дополнительные ссылки на добавление игры, жанры и платформы. Так же изменяются ссылки при входе и выходе из аккаунта. Если пользователь вошёл, то ему предоставлены ссылки на выход из аккаунта и личных кабинет, а если пользователь не вошёл, то ему предлагается войти.

Главное навигационное меню сайта представлено ссылками на категории продаваемых товаров,

6.3 **Выполнение программного кода на Spring Boot**

Для построения приложения использовался паттерн Model View Controller. Реализация данного паттерна представлена фреймворком Spring Boot. Это значит что пользователь не пишет каждый раз запрос к базе данных «руками», а обращается к entity-сущностям для изъятия данных из базы данных. Вся логика обрабатывается в классах-сервисах. В классах-контроллерах мы соединяем данные полученные из сервисов с нужным мапингом ссылок. Главный класс приложения запускающий конфигурацию:

1. import org.slf4j.Logger;
2. import org.slf4j.LoggerFactory;
3. import org.springframework.boot.SpringApplication;
4. import org.springframework.boot.autoconfigure.SpringBootApplication;
5. import org.springframework.context.annotation.Configuration;
6. @Configuration
7. @SpringBootApplication
8. public class DemoApplication {
9. private static Logger logger = LoggerFactory.getLogger(DemoApplication.class);
10. public static void main(String[] args) {
11. SpringApplication.run(DemoApplication.class, args);
12. }}

Подключение к серверу базы данных организуется посредством Java Persistence API с использованием Hibernate:

1. server.port=8080
2. spring.jpa.database=POSTGRESQL
3. spring.datasource.platform=postgres
4. spring.datasource.url=jdbc:postgresql://localhost:5432/kurss
5. spring.datasource.username=postgres
6. spring.datasource.password= postgres

Все запросы к базе данных обрабатывает реализация интерфейса JpaRepository. Он позволяет совершать выборку данных из таблиц без написания запроса «руками». Пример интерфеса с необычными запросами к базе данных:

1. import com.psu.kurs.demo.entity.User;
2. import org.springframework.data.jpa.repository.JpaRepository;
3. import org.springframework.stereotype.Repository;
4. @Repository
5. public interface UserRepository extends JpaRepository<User, Long> {
6. User findByUsername(String username);
7. }

Результат выполнения данного запроса возвращает данных с выбранными условиями.

## ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы была реализована база данных информационной системы бюро проката ретро игр. Позволяющая администратору интерактивно взаимодействовать с некоторыми данными через пользовательский интерфейс.

База данных была разработана спроектирована и разработана с помощью программных встроенных утилит в InelliJ IDEA. Программный продукт реализован при помощи языков разметки HTML, CSS с использованием фреймворка Bootstrap, Java Script и с использованием платформы фреймворка Spring Boot.

Приложение позволяет:

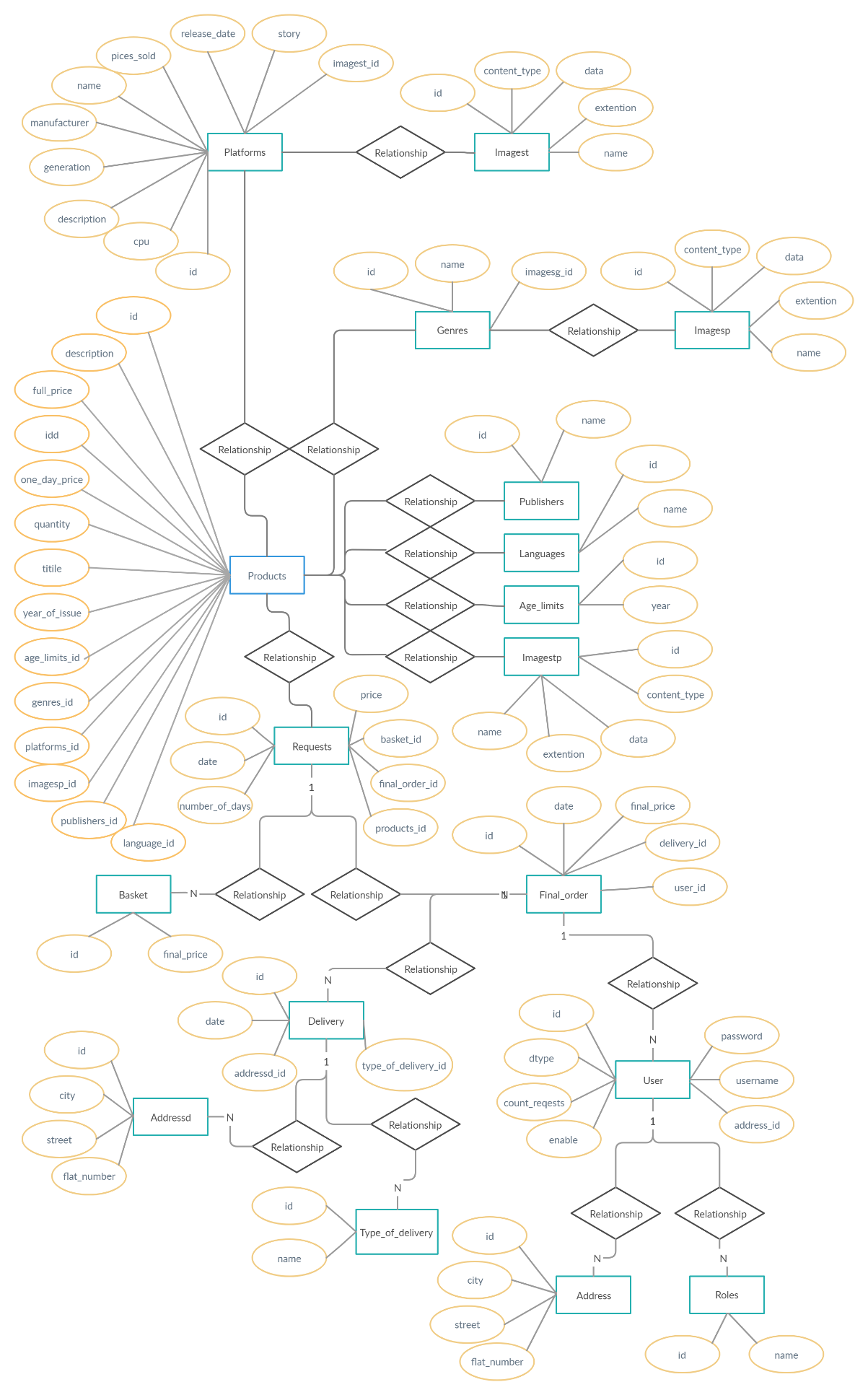
* Просматривать список игр;
* Просматривать список жанров;
* Просматривать список платформ;
* Выбирать игры по жанрам;
* Выбирать игры по платформам;
* Добавление игр;
* Добавление платформ;
* Добавление жанров;
* Добавление товара в корзину;
* Оформление заказа;
* Просмотр сделанных заказов;

В процессе выполнения данной курсовой работы были закреплены навыки в программировании на Java с использованием фреймворка Spring Boot, проектировании баз данных и реализации их в СУБД PostgreSQL.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. IDEF1X - Wikipedia, the free encyclopedia / Многоязычная общедоступная свободно распространяемая энциклопедия, публикуемая в Интернете Википедия. Режим доступа: https://en.wikipedia.org/wiki/IDEF1X
2. Нормальная\_форма - Wikipedia, the free encyclopedia / Многоязычная общедоступная свободно распространяемая энциклопедия, публикуемая в Интернете Википедия. Режим доступа: https://ru.wikipedia.org/wiki/Нормальная\_форма
3. Что такое PostgreSQL? Плюсы и минусы бесплатной базы данных. Режим доступа: [https://oracle-patches.com/common/3214 -что-такое-postgresql]3
4. Что такое Bootstrap и зачем он нужен? Режим доступа: https://itchief.ru/bootstrap/introduction
5. Что такое Bootstrap? Режим доступа: https://blogwork.ru/chto-takoe-bootstrap/

## ПРИЛОЖЕНИЕ А: КОНЦЕПТУАЛЬНАЯ СХЕМА БД



**Рисунок A.1** – ER-диаграмма проектируемой базы данных

## ПРИЛОЖЕНИЕ Б: СХЕМА РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ

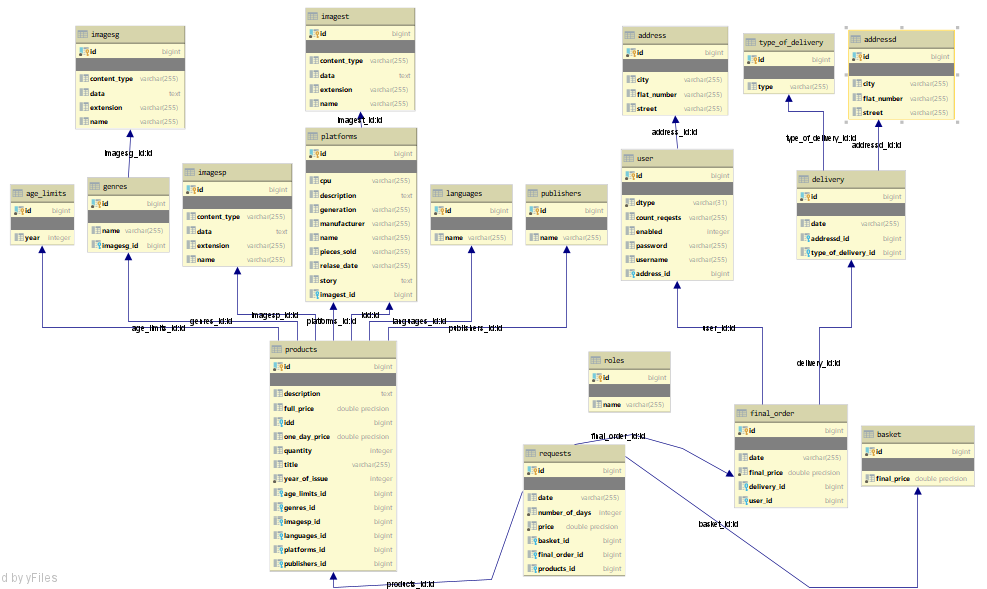
****

Рисунок Б.1 – Схема реляционной базы данных

## ПРИЛОЖЕНИЕ В: ГЛАВНАЯ И РАБОЧИЕ ФОРМЫ ПРИЛОЖЕНИЯ

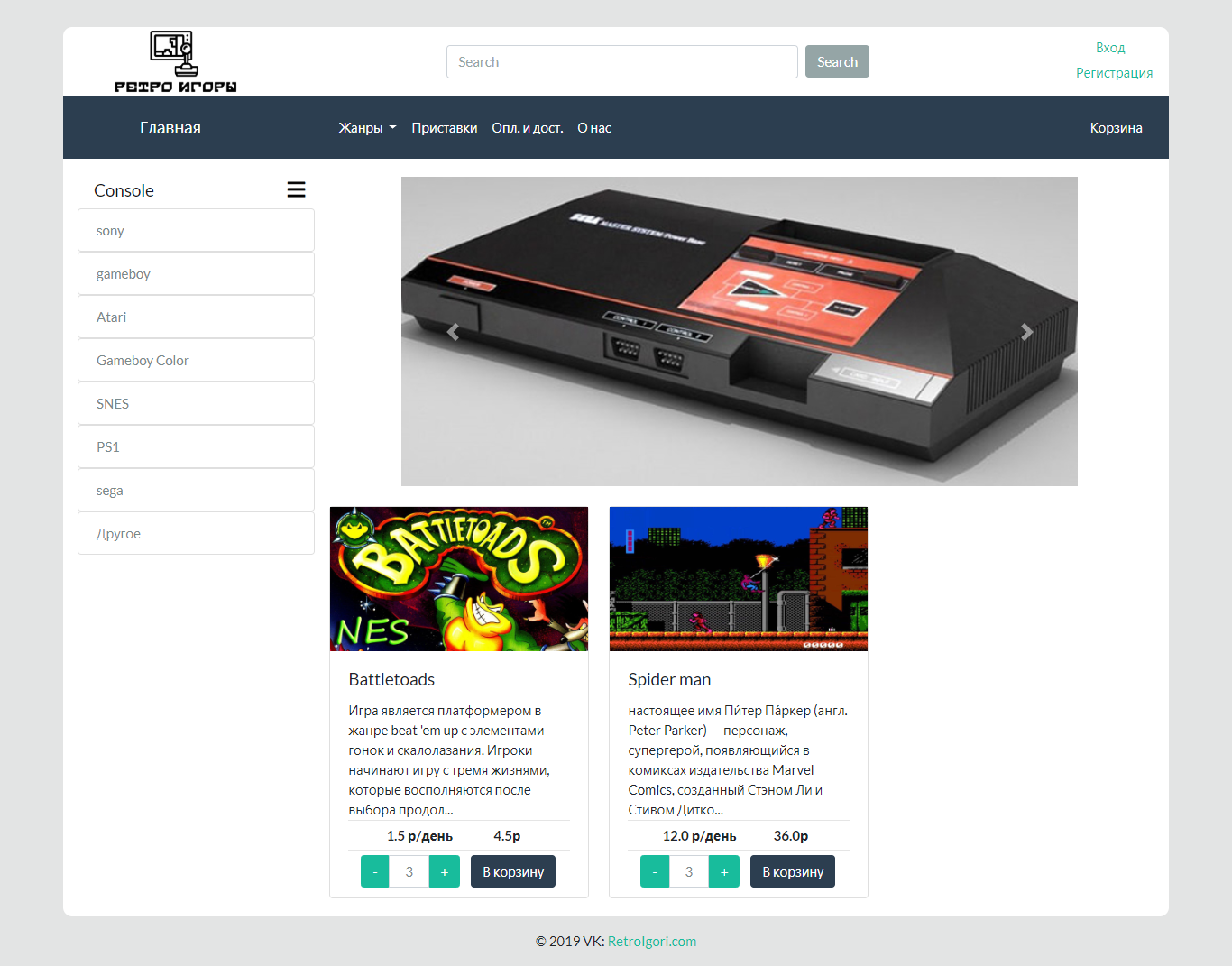


Рисунок В.1 – Главная страница

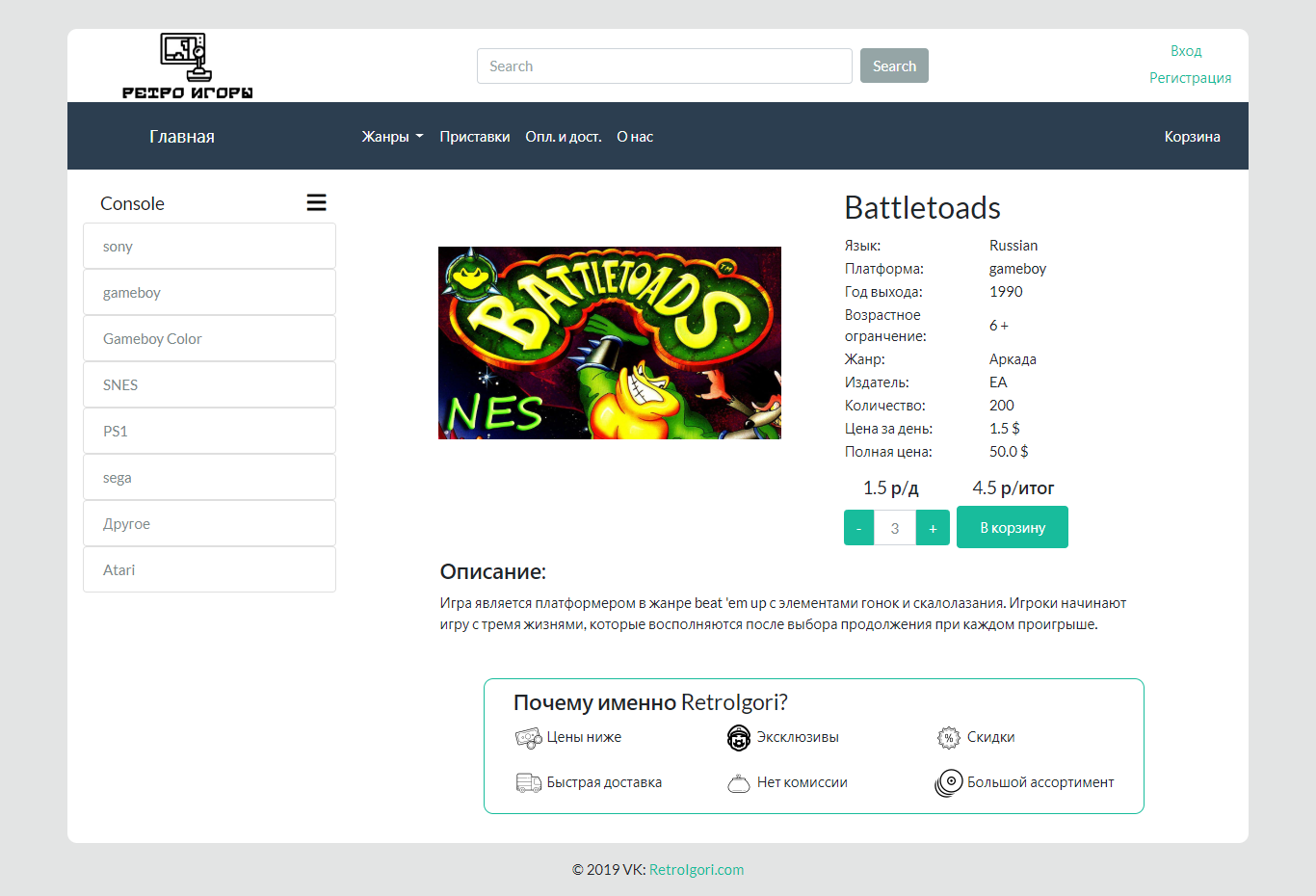


Рисунок В.2 – Информация об игре

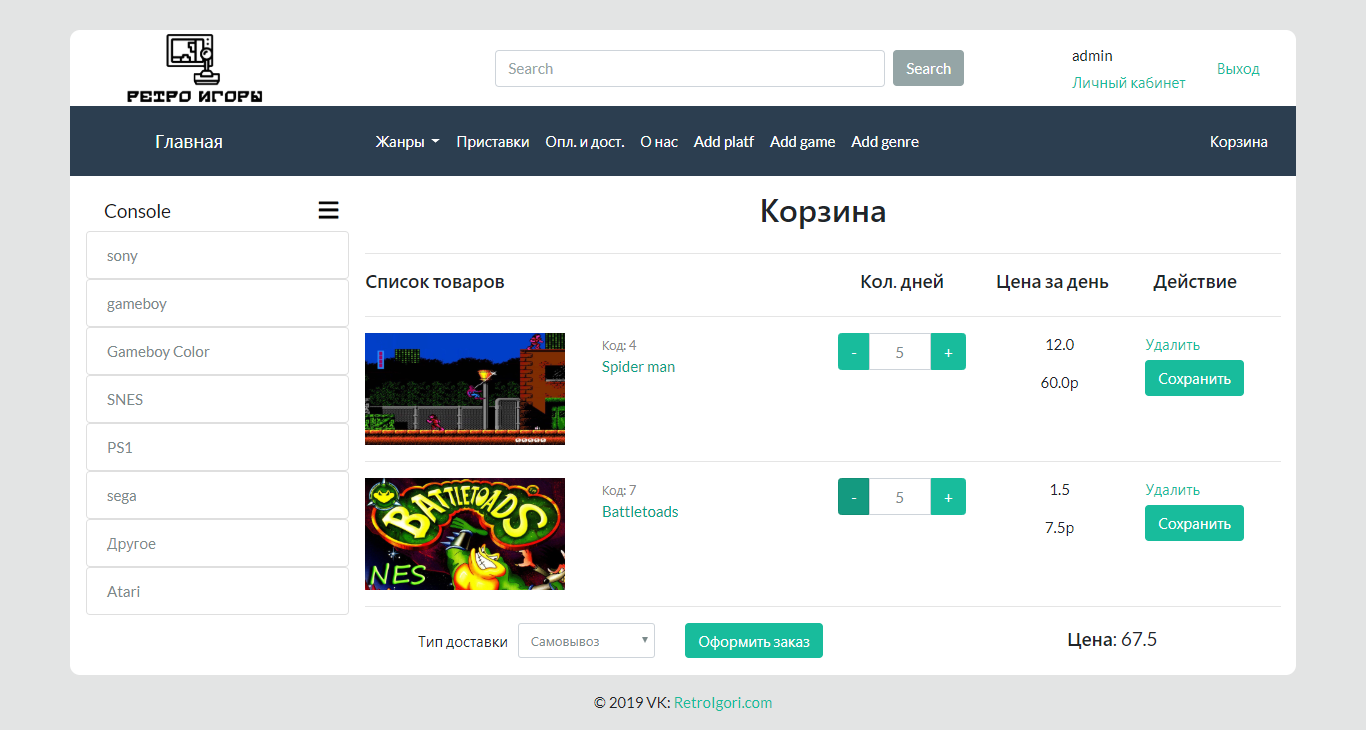


Рисунок В.3 – Корзина

## ПРИЛОЖЕНИЕ Г: ЛИСТИНГ ПРОГРАММЫ