КафедраАСОиУ

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**по курсовой работе**

**Факультет**: КТИ

**Выполнили:** Коземерчак Екатерина Ярославовна**Группа:** №1373

Карасева Лада Андреевна **Группа**: №1373

**Преподаватель:** Зорин Кирилл Михайлович

Выполнено 02.12.2014

Дата \_\_\_\_\_\_\_\_\_\_

Оценка \_\_\_\_\_\_\_\_\_\_

Подпись преподавателя \_\_\_\_\_\_\_\_\_\_

Санкт-Петербург

2013

Содержание.

Contents

[Техническое задание. 3](#_Toc406701770)

[Проектирование базы данных. 3](#_Toc406701771)

[Формирование ER-диаграммы. 4](#_Toc406701772)

[Получение отношений из ER-диаграммы. 5](#_Toc406701773)

[Приведение ко 1 НФ 6](#_Toc406701774)

[Приведение ко 2 НФ. 6](#_Toc406701775)

[Приведение к 3 НФ. 7](#_Toc406701776)

[Использование средств БД. 7](#_Toc406701777)

[Вывод. 8](#_Toc406701778)

[Приложение 1. ER-диаграмма. 8](#_Toc406701779)

[Приложение 2. Таблицы базы данных. 9](#_Toc406701780)

# Техническое задание.

Настоящий курсовой проект предназначен для практического освоения проектирования реляционных баз данных (БД). В работе используется трехуровневый подход к проектированию БД: анализ предметной области, логическое проектирование, физическое проектирование. Задачей курсового проекта является выполнение первых двух уровней. Результатом является логическая схема БД в третьей нормальной форме.

Последовательность выполнения курсовой работы:

1. Анализ предметной области и построение концептуальной модели в виде ER-диаграммы.
2. Отображения ER-диаграммы на реляционную схему .
3. Приведение реляционной модели БД к третьей нормальной форме (3 НФ);

Пояснительная записка должна содержать:

1. Задание на курсовую работу.
2. Концептуальная модель (ER-диаграмма) с необходимыми пояснениями.
3. Первоначальный вариант реляционной модели данных.
4. Нормализованная реляционная модель данных.

**Задание:** Создание базы данных, для игрового приложения "Карточная RPG".Такая база данных должна обеспечивать хранения данных о картах, пользователях и результатах партий, для создания рейтинга пользователей.

# Проектирование базы данных.

Распределение данных осуществляется следующим образом: данные с отдельной подсхемой. На различных объектах используются различные структуры данных, объединяемые в интегрированную систему.

Предметные БД, ориентированные на конкретный класс данных, например, предметная БД "Материалы", которая может быть использована для различных приложений. Создаваемая база данных является предметной, так как ориентирована на дальнейшее ее использование программой, т.е. самой карточной RPG.

Проектирование базы происходит по методу «Сущность-Связь» (entity - relationship), в литературе наряду с этим используется термин "ER - модель" или "модель Чена". Базовыми структурами в ER - модели являются "типы сущностей" и "типы связей".

Отличие от типа связи от типа сущности - в установлении зависимости существования реализации одного типа от существования реализации другого.

## Формирование ER-диаграммы.

В версии 1.0 программы, мы имеем только три основные сущности базы данных:

1. Юзер (ID, никнейм, пароль, права (type) )
2. Лог (ID, юзер, дата, результат)
3. Карта (ID, Название, ХП, атака, тип, уровень, способности)

Данная работа рассмотрит более расширенную базу данных, которая будет использована в версии 2.0 программы, чтобы лучше раскрыть тему формирования БД.

Для проектирования базы предметной области будет достаточно семи сущностей:

1. Юзер (ID, никнейм, пароль, права (type) )
2. Лог (ID, юзер, дата, результат)
3. Карта (ID, Название, ХП, атака, тип, уровень, IDспособности, функция воздействия)
4. Способности (ID, функция воздействия, ограничения)
5. Права (IDюзера, Тип пользователя, доступ к редактированию карт, доступ к редактированию лога, доступ к редактированию пользователя)
6. Колоды (ID, ID юзера, карты колоды)
7. Звания (ID, звание, описание)

**Связи между сущностями:**

1. *Рейтинг игроков:* Игрок (ID, никнеим) - Лог (результат) - У игрока может быть несколько результатов и результат персонален для каждого игрока. Следовательно, имеет место связь «**один ко многим».** Игрок может не иметь результата, но результат не может не иметь игрока
2. *Права пользователей:* Юзеры (права) - Права (тип пользователя) - У игрока может быть один вид прав и один тип пользователя может принадлежать разным игрокам. Следовательно, имеет место связь **«один ко многим».** Игрок не может не иметь типа. Тип может быть не присвоен ни одному игроку.
3. *Колоды пользователей:* Юзер(ID)- Колода (ID юзера) - У игрока может быть только одна колода и одна колода может быть у нескольких игроков. Следовательно, имеет место связь **«один ко многим».** Игрок не может не иметь колоды. Колода не может не иметь игрока.
4. *Звания пользователя:* Юзер(ID) - звание (ID юзера) - У игрока может быть только одно звание, одно звание может быть у нескольких игроков. Следовательно, имеет место связь **«один ко многим»**. Игрок не может не иметь звания. Звание может быть не присвоен ни одному игроку.
5. *Использование способности карты:* Карта (способности) - Способности (ID) - каждой карты может быть одна способность, одна способности может быть только у одной карты. Следовательно, имеет место связь **«один к одному».** У карты может не быть способности. У способности не может не быть карты.
6. *Карта в колоде* - Колода (ID) - Карта (ID) - В каждой колоде может быть много карт. Одна карта может быть в нескольких колодах. Следовательно, имеет место связь **«многие ко многим».** Карта может существовать без колоды, в колоде может не быть карт.

## 

## Получение отношений из ER-диаграммы.

Связь [*Рейтинг игроков*] - связь 1-*М* с обязательным классом принадлежности у *М*-связной сущности, раскрывается по правилу №4 – в отношение, отводимое *М*-связной сущности переходит, как атрибут, ключ из 1-связной сущности:

Лог (ID, *юзер*, дата, результат)*(уже присутствует)*

Юзер (ID, никнейм, пароль, права (type) )

Связь [*Колоды пользователей*] - связь 1-*М* с обязательным классом принадлежности у *М*-связной сущности, раскрывается по правилу №4 – в отношение, отводимое *М*-связной сущности переходит, как атрибут, ключ из 1-связной сущности:

Юзер (ID, никнейм, пароль, права (type), *IDколоды*)

Колоды (ID, карты колоды)

Связь [*Звания пользователя*] - связь 1-*М* с обязательным классом принадлежности у *М*-связной сущности, раскрывается по правилу №4 – в отношение, отводимое *М*-связной сущности переходит, как атрибут, ключ из 1-связной сущности:

Звания (ID, звание, описание*,*)

Юзер (ID, никнейм, пароль, права (type),*ID звания*)

Связь [*Права пользователей*] - связь 1-*М* с обязательным классом принадлежности у *М*-связной сущности, раскрывается по правилу №4 – в отношение, отводимое *М*-связной сущности переходит, как атрибут, ключ из 1-связной сущности:

Юзер (ID, никнейм, пароль, права (type) )

Права (*ID юзера*, Тип пользователя, доступ к редактированию карт, доступ к редактированию лога, доступ к редактированию пользователя)*(уже присутствует)*

Связь [*Карта в колоде*] - связь M-N, независимо от класса принадлежности сущностей по правилу №6 формируем три отношения. Два отношения соответствуют связываемым сущностям и их ключи являются первичными ключами этих отношений. Третье отношение является связным между первыми двумя, а его ключ объединяет ключевые атрибуты связываемых отношений.

Карта (ID, Название, ХП, атака, тип, уровень, ID способности, функция воздействия)

Колоды (ID,название,ID юзера)

Карта в колоде (IDкарты, IDколоды)

Связь [*Использование способности карты*] – связь 1-1 с обязательным классом принадлежности обеих сущностей, раскрывается по правилу №1: формируется одно отношение. Первичным ключом этого отношения может быть ключ любой из двух сущностей.

Карта (ID, Название, ХП, атака, тип, уровень, ID способности, функция воздействия, ограничения)

**После добавления неключевых атрибутов схемы отношений примут следующий вид:**

1. Юзер (ID, никнейм, пароль, права (type), *IDколоды,ID звания,*)
2. Лог (ID, *юзер*, дата, результат)
3. Карта (ID, Название, ХП, атака, тип, уровень, ID способности, функция воздействия)
4. Колоды (ID, название, ID юзера)
5. Карта в колоде (IDкарты, IDколоды)
6. Права (*ID юзера*, Тип пользователя, доступ к редактированию карт, доступ к редактированию лога, доступ к редактированию пользователя)
7. Звания (ID, звание, описание*,*)

Получили первоначальный вариант реляционной модели данных.

### Приведение к 1 НФ

Когда поле в данной записи содержит более одного значения для каждого вхождения первичного ключа, такие группы данных называются *повторяющимися группами*. 1НФ не допускает наличия таких многозначных полей.

Таблица находится в первой нормальной форме, если каждый её атрибут атомарен. Под выражением «атрибут атомарен» понимается, что атрибут может содержать только одно значение. Таким образом, не существует 1NF таблицы, в полях которых могут храниться списки значений. Для приведения таблицы к 1NF обычно требуется разбить таблицу на несколько отдельных таблиц.

Атрибуты всех отношений являются простыми (имеют единственное значение), т.е. **отношения находятся в 1НФ.**

### Приведение ко 2НФ.

Таблица находится во второй нормальной форме, если она находится в первой нормальной форме, и при этом любой её атрибут, не входящий в состав первичного ключа, функционально полно зависит от первичного ключа. Функционально полная зависимость означает, что атрибут функционально зависит от всего первичного составного ключа, но при этом не находится в функциональной зависимости от какой-либо из входящих в него атрибутов(частей). Или другими словами: в 2НФ нет неключевых атрибутов, зависящих от части составного ключа (+ выполняются условия 1НФ).

Все отношения находятся во второй нормальной форме.

### Приведение к 3 НФ.

Таблица находится в третьей нормальной форме, если она находится во второй нормальной форме, и при этом любой её неключевой атрибут функционально зависит только от первичного ключа

Все отношения находятся в третьей нормальной форме.

### Использование средств БД.

1. Использование индексов

Для ускорения работы программы в базе используется индекс по полю Юзер.Никнейм:

create index [gamer\_name\_index] on [gamer]([gamer\_name]);

1. Использование триггеров

Для сохранения целостности данных используется триггер в таблице Юзер: при создании пользователя с новым типом права, создается соответсвующая строчка в таблице Права:

IF OBJECT\_ID ('create\_right','TR') IS NOT NULL

DROP TRIGGER create\_right;

GO

CREATE TRIGGER create\_right

ON [gamer]

FOR INSERT

AS

if (select count(\*) from inserted join [right] on right\_user\_type = gamer\_type)= 0

begin

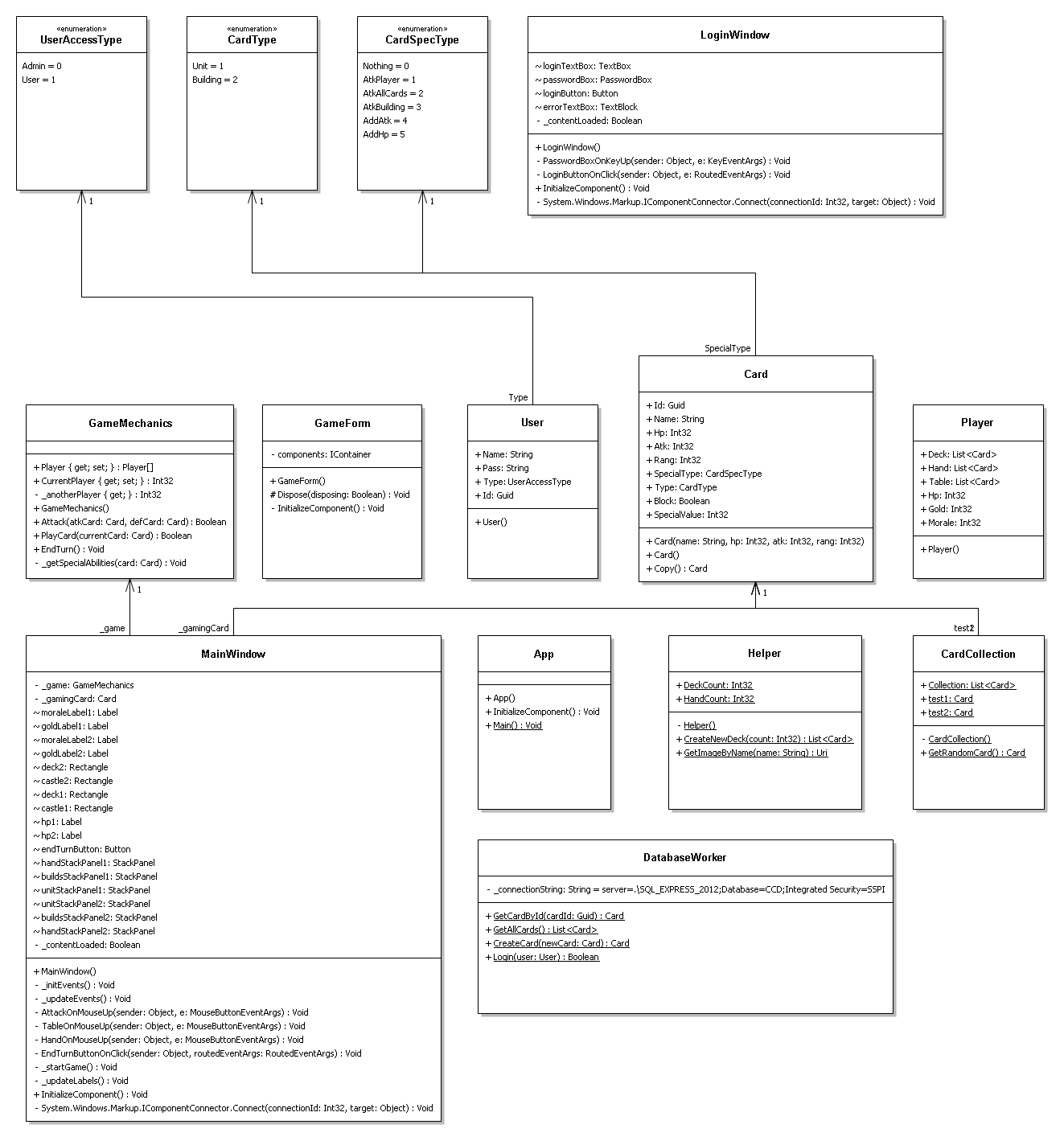
insert into [right] (right\_id, right\_user\_type, right\_edit\_card, right\_edit\_log, right\_edit\_user)

select NEWID(), gamer\_type, 0, 0, 0 from inserted group by gamer\_type

end

GO

**Диаграмма классов UML**



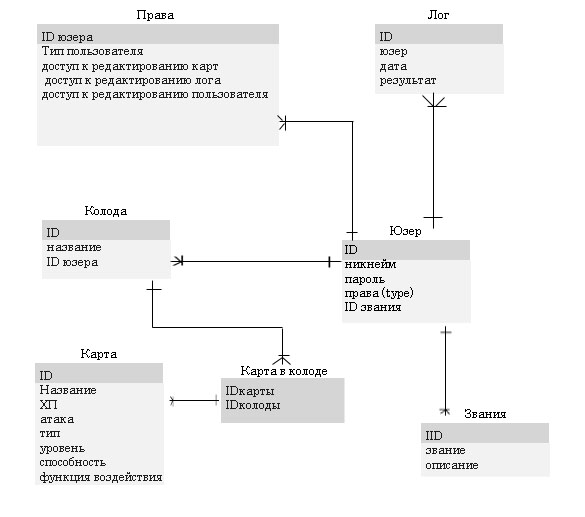
# Вывод.

В процессе выполнения курсового проекта «Концептуальное и логическое проектирование баз данных» была создана реляционная база данных. В качестве области применения была выбрана база данных «Карточная RPG». В ходе выполнения курсового проекта была построена концептуальная модель, а затем и первичный вид реляционной базы данных. Методов последовательного приведения к видам нормальной формы база данных была приведена к третьей нормальной форме, также была построена ER-диаграмма раскрывающая связи и объекты.

# ER-диаграмма.

**Логическая ER-диаграмма**

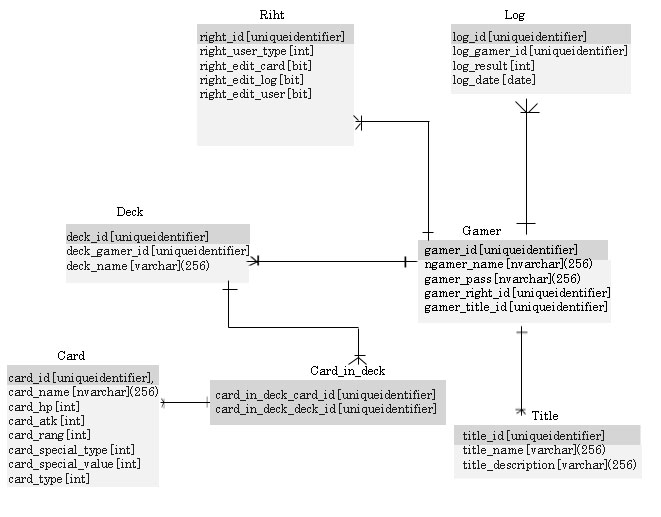
(на русском языке, без типов данных)



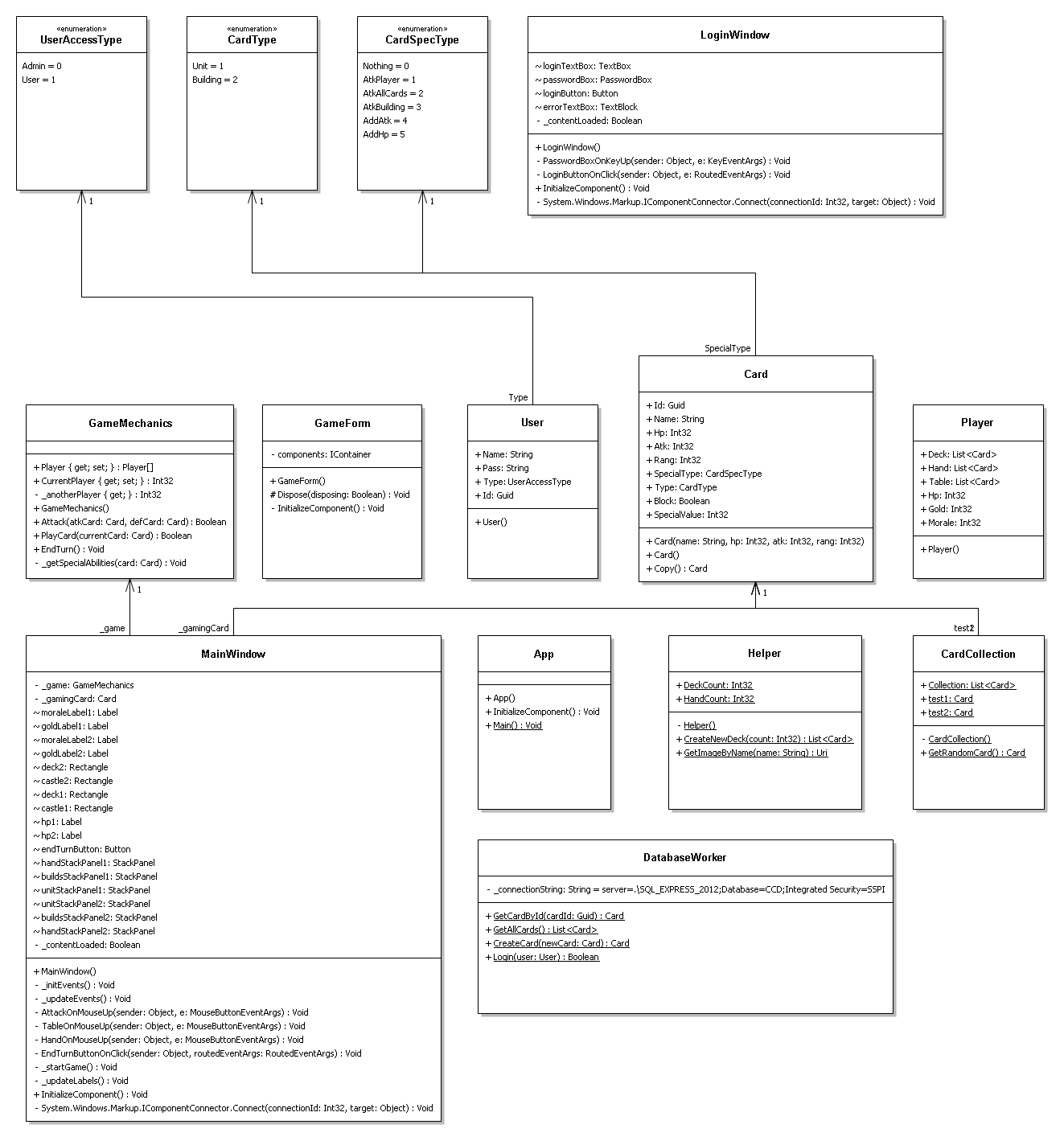
# 

**Физическая ER-диаграмма**

(на аглийском языке, с типами данных)



# Диаграмма классов UML



# Таблицы базы данных.

1. Юзер (ID, никнейм, пароль, права (type), *ID звания,*)
2. Лог (ID, *юзер*, дата, результат)
3. Карта (ID, Название, ХП, атака, тип, уровень, способность, функция воздействия)
4. Колоды (ID, название, ID юзера)
5. Карта в колоде (IDкарты, IDколоды)
6. Права (*ID юзера*, Тип пользователя, доступ к редактированию карт, доступ к редактированию лога, доступ к редактированию пользователя)
7. Звания (ID, звание, описание*,*)

*Gamer*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Name | Password | Type | Title\_ID |
|  |  |  |  |  |

*Log*

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Gamer\_ID | Date | Result |
|  |  |  |  |

*Deck*

|  |  |  |
| --- | --- | --- |
| ID | Name | User\_ID |
|  |  |  |

*User\_type*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID\_user | User\_type | Access\_to\_edit\_card | Access\_to\_edit\_log | Access\_to\_edit\_user |
|  |  |  |  |  |

*Card\_in\_Deck*

|  |  |
| --- | --- |
| ID\_card | ID\_Deck |
|  |  |

*Title*

|  |  |  |
| --- | --- | --- |
| ID | name | description |
|  |  |  |

*Card*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID\_card | HP | Attack | Card\_type | Level | Special\_type | Special\_value |
|  |  |  |  |  |  |  |

**Разработка программного обеспечения**

Так же в данной курсовой работе был использован веб-сервис GitHub, использующийся для совместной разработке ПО.