# ВВЕДЕНИЕ

Практическая значимость разработки и использования прототипа информационной системы для обеспечения работы билетных касс состоит в автоматизировании и упрощении процесса покупки билетов и абонементов, регламентировании их цен, удобном администрировании и организованном управлении персоналом, а также в отслеживании информации о посещаемости матчей.

Цель курсового проекта: формирование навыков практической работы, связанной с проектированием, разработкой и реализацией прототипа информационной системы, а также для усовершенствования и повышения эффективности работы сотрудников и стадиона в целом.

Задачи курсового проекта:

* проанализировать предметную область;
* определить функциональные требования к системе;
* построить макеты прототипа системы
* построить диаграммы вариантов использования, диаграммы классов;
* представить схему базы данных в виде ER-модели;
* разработать и представить прототип информационной системы.

# 1 РАЗРАБОТКА ПРОЕКТА ПРОТОТИПА ИНФОРМАЦИОННОЙ СИСТЕМЫ

## 1.1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

Имеется «Билетная касса», прототип информационной системы для которой следует разработать. Данный прототип предназначен для обеспечения просмотра футбольными болельщиками выбранных ими матчей на купленных ими местах. Система должна позволять покупать и сдавать билеты и абонементы на матчи, при условии, что матчи проводятся на одном стадионе, места пронумерованы, и работает одновременно несколько касс.

## 1.2 ГЛОССАРИЙ КУРСОВОГО ПРОЕКТА

Автоинкрементироваться-автоматически увеличиваться на единицу.

База данных-именованная совокупность данных, отражающая состояние объектов и их отношений в рассматриваемой предметной области.[1]

Данные-информация, представленная в определенном виде, позволяющем автоматизировать её сбор, хранение и дальнейшую обработку.[2]

Информационная система-система, предназначенная для хранения, поиска и обработки информации, и соответствующие организационные ресурсы.[3]

Оператор базы данных-работник, занимающийся созданием и ведением базы данных.

Счётчик serial-тип в языке sql, который позволяет атрибуту автоинкрементироваться при добавлении новой строки в таблицу.

Default-значение по умолчанию.

Real-вещественный числовой тип.

Timestamp without time zone-тип в языке sql, позволяющий хранить в себе дату и время без учета часовых зон.

True/false-значение bool, то есть истина или ложь.

## 1.3 ИМЕЮЩИЕСЯ АНАЛОГИ

Похожее решение-информационная система «Tickets Rubin-Kazan». Она дает возможность приобрести билеты и абонементы на матчи «Рубина», а также сдать их.

Преимущества над «Football Tickets»:   
-быстрое и плавно работающее меню  
-возможность перейти на сайт команд, которые играют в данном матче

Недостатки:

-система разработана относительно давно, то есть устарела  
-при выборе места на стадионе тормозит и долго обрабатывает запросы в силу своих технических характеристик  
-при показе мест пользователь не совсем понимают какие занятые, а какие свободны, поэтому приходится спрашивать об этом кассира.

Ещё один аналог-информационная система «Booking Tickets». Эта система в свою очередь дает возможность приобрести билеты и абонементы на любые матчи РПЛ (Российской Премьер Лиги), а также сдать их.

Преимущества над «Football Tickets»:   
-быстро работающий интерфейс  
-возможность посмотреть подробную статистику команд, участвующих в матче, таблицу РПЛ

-относительно новая система

Недостатки:

-невозможность забронировать билеты на месяц и больше вперед

-в некоторых местах незаметно название команды из-за красок меню

## 1.4 РОЛИ СИСТЕМЫ И ФУНКЦИИ

Покупатель-покупает билет/абонемент, сдает билет, просматривает информацию о будущих матчах.

Кассир-совершает процесс оплаты и в случае успешного выполнения выдает билет, удаляет сданный билет из базы данных и возвращает деньги.

Оператор базы данных-создает базу данных, занимается её ведением, добавляет матчи, команды и редактирует их.

Администратор-просматривает общую посещаемость, количество проданных абонементов, посещаемость за выбранный участок времени.

## 1.5 ВАРИАНТЫ ИСПОЛЬЗОВАНИЯ

Диаграмма вариантов использования приведена на рисунках 1.1,1.2.

Рисунок 1.1-«Диаграмма вариантов использования для покупателя и кассира»

Рисунок 1.2-«Диаграмма вариантов использования для администратора и оператора базы данных»

Варианты использования:

* Войти в систему
* Купить билет
* Сдать билет
* Купить абонемент
* Просмотреть общую посещаемость
* Просмотреть посещаемость за выбранный период
* Просмотреть количество купленных абонементов
* Управлять персоналом
* Просмотреть таблицу команд
* Просмотреть таблицу матчей

Теперь разберем каждый вариант использования подробно.

**Вариант использования "Войти в систему"**

*Краткое описание:*

Данный вариант использования описывает вход кассира/администратора/оператора базы данных в систему покупки билетов.

*Основной поток событий:*

Данный вариант использования начинает выполняться, когда кассир/администратор/оператор базы данных хочет войти в систему покупки билетов.

1.Система запрашивает имя пользователя и пароль.

2.Кассир/администратор/оператор базы данных вводит имя и пароль.

3.Система подтверждает имя и пароль, после чего открывается доступ в систему.

*Альтернативные потоки:*

*Неправильное имя/пароль*

Если во время выполнения основного потока обнаружится, что кассир/администратор/оператор базы данных ввел неправильное имя и/или пароль, система выводит сообщение об ошибке. Он может вернуться к началу основного потока или отказаться от входа в систему, при этом выполнение варианта использования завершается.

**Вариант использования "Купить билет"**

*Краткое описание:*

Данный вариант использования описывает процесс покупки покупателем билета на выбранный матч.

*Основной поток событий:*

Данный вариант использования начинает выполняться, когда покупатель начинает выбирать матч для покупки билета.

1.Система отображает список ближайших матчей.

2.Покупатель выбирает матч из каталога.

3.Система отображает каталог секторов.

4.Покупатель выбирает нужный сектор.

5.Система отображает каталог мест. Около начала ряда отображается номер ряда, на сидении указано место: если оно черного цвета-занято.

6.Покупатель выбирает свободное место.

7.Система рассчитывает стоимость выбранных билетов.

8.Покупатель оплачивает картой или наличными кассиру.

9.После успешного совершения оплаты кассир выдает покупателю билет(ы).

10.Система отмечает только что купленные места занятыми

11.Система спрашивает покупателя хочет ли он купить еще билет

*Покупатель хочет купить еще билет*

1.Система отображает каталог мест сектора, который покупатель до этого выбрал.

2.Пользователь может выбрать место на этом же секторе, а может перейти к выбору секторов или матча сначала, начиная основной поток событий заново

*Покупатель не хочет больше покупать билет*

1.Система завершает свою работу и приходит в исходное состояние.

*Альтернативные потоки:*

*Ближайших матчей нет*

В первый месяц после последней игры сезона матчей не проводится и расписания будущих матчей нет. В таком случае обычно у кассиров выходной и покупатель не может купить билет, так как система не работает.

*Не осталось мест*

Покупатель выбирает матч и действие *Купить билет*, но все билеты куплены. В таком случае система выдает сообщение «Все билеты куплены» и возвращается к началу потока, где предлагает покупателю выбрать другой матч.

*Покупатель выбрал купленное место*

Покупатель не может выбрать купленное место, так как для его выбора доступны только свободные места.

*Недостаточно средств на карте или введен неверный пин-код*

Система позволяет ввести пин-код еще раз или оплатить наличными или другой картой. Если покупатель так и не смог оплатить, кассир проходит авторизацию и сдает билет, возвращая место свободным.

**Вариант использования** **"Сдать билет"**

*Краткое описание:*

Данный вариант использования описывает процесс сдачи покупателем билета.

*Основной поток событий:*

Данный вариант использования начинается с потока войти в систему

1.Кассир входит в систему.

2.Покупатель говорит кассиру о его намерении сдать билет.

3.Кассир выбирает действие сдать билет.

4.Кассир вносит данные о билете и система удаляет из базы данных этот билет и делает место свободным, при условии что дата матча больше нынешней и билет был действительным

4.Кассир возвращает деньги за билет.

*Альтернативные потоки:*

*Выбранное место свободно*

У покупателя либо недействительный билет, либо матч уже прошел. В таком случае система отклоняет запрос на сдачу билета.

**Вариант использования "Купить абонемент"**

*Краткое описание:*

Данный вариант использования описывает процесс покупки покупателем абонемента на сезон.

*Основной поток событий:*

Данный вариант использования позволяет покупателю купить абонемент

1.Меню системы отображает несколько функций, в том числе приобретения абонемента.

2.Покупатель выбирает приобрести абонемент.

3.Система отображает каталог секторов.

4.Покупатель выбирает нужный сектор.

5.Система отображает каталог мест. Около начала ряда отображается номер ряда, на сидении указано место: если оно черного цвета-занято.

6.Покупатель выбирает свободное место.

7.Система рассчитывает стоимость абонемента.

8.Покупатель оплачивает картой или наличными кассиру.

9.После успешного совершения оплаты кассир выдает покупателю абонемент.

10.Система отмечает только что купленные места занятыми

11.Система отмечает, что на данную трибуну куплен абонемент.

*Альтернативные потоки:*

*Недостаточно средств на карте или введен неверный пин-код*

Система позволяет ввести пин-код еще раз или оплатить наличными или другой картой. Если покупатель так и не смог оплатить, кассир проходит авторизацию и сдает абонемент, возвращая место свободным.

*Не осталось абонементов*

Все абонементы на трибуну выкуплены. В таком случае система выдает сообщение об этом.

*Покупатель выбрал купленное место*

Покупатель не может выбрать купленное место, так как для его выбора доступны только свободные места.

**Вариант использования "Просмотреть общую посещаемость"**

*Краткое описание:*

Данный вариант использования позволяет администратору просмотреть общую посещаемость

*Основной поток событий:*

Данный вариант использования начинается с потока войти в систему.

1.Администратор входит в систему.

2.Система показывает меню администратора.

3.Администратор выбирает показать общую посещаемость.

4.Система показывает общее количество купленных билетов.

*Альтернативные потоки:*

*Количество билетов равно нулю*

Если билетов еще не было куплено в этом сезоне и в базе данных нет данных о билетах, то система выдает значение количества билетов равное нулю.

**Вариант использования " Просмотреть посещаемость за выбранный период"**

*Краткое описание:*

Данный вариант использования позволяет администратору просмотреть посещаемость матчей за выбранный им период.

*Основной поток событий:*

Данный вариант использования начинается с потока войти в систему.

1.Администратор входит в систему.

2.Система показывает меню администратора.

3.Администратор выбирает показать посещаемость за период.

4.Администратор выбирает границы дат, посещаемость в которые хочет посмотреть.

5.Система показывает посещаемость за выбранный период.

*Альтернативные потоки:*

*Матчей в выбранном периоде не было*

Если матчей не было, система показывает количество равное нулю

*Дата левой границы больше даты правой границы*

Администратор выбрал первую дату периода большую, чем вторую дату. Тогда система выдает сообщение о том, что вторая дата должна быть не меньше первой. Администратор вновь может изменить даты.

**Вариант использования "Просмотреть количество купленных абонементов"**

*Краткое описание:*

Данный вариант использования позволяет администратору просмотреть количество купленных абонементов.

*Основной поток событий:*

Данный вариант использования начинается с потока войти в систему.

1.Администратор входит в систему.

2.Система показывает меню администратора.

3.На меню администратора появляется количество купленных абонементов.

*Альтернативные потоки:*

*Абонементов нет*

Если абонементов еще не было куплено, то количество купленных абонементов выводится как равное нулю.

**Вариант использования "Управлять персоналом"**

*Краткое описание:*

Данный вариант использования позволяет администратору управлять его подчиненными.

*Основной поток событий:*

1.Администратор входит в систему.

2.Система показывает меню администратора.

3.Администратор выбирает функцию управления персоналом.

4.Система открывает таблицу персонала и возможные операции.

*Добавить сотрудника*

1.Администратор выбирает добавить сотрудника.

2.Система показывает поля для добавления сотрудника и дает администратору их заполнить.

3.Администратор заполняет все поля.

4.Система добавляет сотрудника.

*Редактировать данные сотрудника*

1.Администратор выбирает редактировать данные сотрудника.

2.Система показывает поля и значения, которые принадлежат выбранному сотруднику.

3.Администратор изменяет значения.

4.Система вносит изменения в базу данных.

*Уволить сотрудника*

1.Администратор выбирает уволить сотрудника.

2.Система запрашивает действие пользователя, правда ли он хочет уволить сотрудника с выбранным ФИО и должностью.

*Да*

Система оставляет данные о сотруднике, но удаляет данные о его логине, пароле и роли в системе, лишая его доступа к своим обязанностям.

*Нет*

Система показывает таблицу сотрудников.

**Вариант использования "Просмотреть таблицу команд"**

*Краткое описание:*

Данный вариант использования позволяет оператору базы данных управлять данными о командах.

*Основной поток событий:*

1.Оператор базы данных входит в систему.

2.Система показывает меню оператора.

3.Оператор базы данных выбирает функцию просмотреть таблицу команд.

4.Система показывает таблицу команд и возможные операции.

*Добавить команду*

1.Оператор выбирает добавить команду.

2.Система показывает поля для добавления команды и дает оператору их заполнить.

3.Оператор заполняет все поля.

4.Система добавляет команду.

*Редактировать данные команды*

1.Оператор выбирает редактировать данные команды.

2.Система показывает поля и значения, которые принадлежат выбранной команде.

3.Оператор изменяет значения.

4.Система вносит изменения в базу данных.

**Вариант использования "Просмотреть таблицу матчей"**

*Краткое описание:*

Данный вариант использования позволяет оператору базы данных управлять данными о матчах.

*Основной поток событий:*

1.Оператор базы данных входит в систему.

2.Система показывает меню оператора.

3.Оператор базы данных выбирает функцию просмотреть таблицу матчей.

4.Система показывает таблицу матчей и возможные операции.

*Добавить матч*

1.Оператор выбирает добавить матч.

2.Система показывает поля для добавления матча и дает оператору их заполнить.

3.Оператор заполняет все поля.

4.Система добавляет команду.

*Редактировать данные матча*

1.Оператор выбирает редактировать данные матча.

2.Система показывает поля и значения, которые принадлежат выбранному матчу.

3.Оператор изменяет значения.

4.Система вносит изменения в базу данных.

*Альтернативные потоки*

*Оператор пытается изменить данные о матче, который уже прошел*

Оператор пытается изменить данные о прошедшим матче, тогда система выдает сообщение о том, что нельзя изменить данные об этом матче.

*Оператор пытается внести данные о новом матче, дата которого меньше нынешней на момент внесения*

Система выдает сообщение о том, что дата должна быть больше нынешней.

## 1.6 ПРОЕКТИРОВНИЕ БАЗЫ ДАННЫХ

Следующим этапом проектирования прототипа информационной системы является проектирование базы данных. Этот этап является одним из самых значимых и важных в проектировании, так как ошибки, допущенные на нем стоят очень дорого как в плане денежном, так и в плане сил и времени, потраченных на переделывание системы исходя из внесенных изменений в хранилище данных. На рисунке 1.3 показана ER-модель проектируемой базы.

\Рисунок 1.3-«Модель базы данных»

Разберем подробно каждую таблицу. Таблица сотрудники хранит в себе информацию о работниках билетной кассы. Поле код сотрудника является уникальным кодом и первичным ключом для данной таблицы, имеющим тип serial-счётчик. Поля фамилия, имя, отчество хранят соответствующие данные о сотруднике в текстовом типе. Поля логин, пароль и роль являются очень важными в работе билетной кассы, так как именно по логину и паролю осуществляется вход в систему, а роль определяет, какие функции и обязанности несет данный сотрудник. Данные поля также являются типом text.

Следующая таблица-Команда. В ней хранятся данные о командах, с которой придется сыграть команде-владелице стадиона, в нашем случае это казанский Рубин. Код команды это первичный ключ и уникальный код, тип которого также serial,то есть счётчик. Название команды хранит в виде text название о команде соперника, а текстовое поле город-родной город команды, с которой предстоит сыграть. Таблица Команда связывается с таблицей Матч через внешний ключ код команды. Связь 1 ко многим, так как одна команда может играть несколько матчей, но один матч хранит в себе только одну команду гостей. Для уникальности каждого поля в таблице Матч введен первичный ключ код матча, также являющийся счётчиком. Поле вида text тип соревнования несет информацию о виде соревнования, в котором проводится матч. Это может быть чемпионат России-Российская Премьер-Лига, или Кубок России и т.д. Дата и время говорит о том, когда состоится матч. Тип timestamp without time zone означает, что в базе хранится не только дата матча, но и время.

Таблица матч связана с таблицей бронь места через внешний ключ код матча. Данная связь будет являться связью вида 1 ко многим, так как на один матч огромное количество мест, но на одну строку с уникальным кодом места только один матч. Данный первичный ключ также является типом serial. Следующие три поля-сектор, ряд, место являются числовыми типами и хранят соответствующую информацию. Класс места может быть обычным или vip. Бронь абонемента хранит информацию о том, куплен этот билет как отдельный или же это место забронировано абонементом. Тип данного поля является bool-true если абонементное, false если билетное. По умолчанию данному полю присваивается значение false. Последнее поле в таблице – стоимость. Оно хранит вещественное представление стоимости места real и имеет свое ограничение default , чтобы стоимость билета начиналась от 50 рублей и выше. Эта таблица связана с таблицей Билет через внешний ключ Код места и связь между ними 1 к 1, то есть одна бронь места соответствует одному билету. Также в таблице билет есть поля Код билета, которое является счётчиком и код сотрудника, которое связывается с таблицей сотрудников также связью один к одному.

## 1.7 МАКЕТЫ ПРОТОТИПА СИСТЕМЫ

После всех вышеперечисленных этапов разработки прототипа информационной системы переходим к самому последнему-макетам. Макеты отображают то, как будет выглядеть система после ее создания, каков будет интерфейс и возможности. Обратимся к меню, которое видит покупатель при запуске системы. Оно изображено на рисунке 1.4. Он может выбрать матч для покупки билета, купить абонемент или завершить работу.

Рисунок 1.4-«Макет 1»

Нажимая на матч, выходит картина стадиона с подписанными секторами. Покупатель может выбрать сектор, который ему нравится. Это иллюстрировано на рисунке 1.5.

Рисунок 1.5-«Макет 2»

Далее покупатель выбирает место. Черные места-занятые. Данные кнопки не доступны для нажатия, поэтому не происходит варианта нажатия на занятые места. Это продемонстрировано на рисунке 1.6.

Рисунок 1.6-«Макет 3»

Далее выходит сообщение о цене и о том, чтобы покупатель оплатил билет кассиру. Затем спрашивается, хочет ли покупатель купить билет еще. Если да, то он выбирает место, или открывает другой сектор или матч, а в противном случае работа программы завершается. Данный макет представлен на рисунке 1.7.

Рисунок 1.7-«Макет 4»

Для покупателей авторизации нет, так как незачем хранить данные о всех посетителях матча, скольких каждый раз свыше десяти тысяч человек. Авторизация предназначена для сотрудников. Как выглядит эта форма показано на рисунке 1.8.

Рисунок 1.8-«Макет 5»

Пройдем авторизацию как кассир. Его меню изображено на рисунке 1.9. Он может сдать билет, выбрав матч, сектор, ряд, место. Если это место действительно занято и матч еще не прошел, то строка об этом месте удалится из базы данных. Также ему доступны функции выйти-то есть пройти авторизацию заново и завершить, что завершит выполнение программы.

Рисунок 1.9 –«Макет 6»

Пройдем авторизацию как оператор базы данных. Его меню изображено на рисунке 1.10. Он может просмотреть таблицу матчей, где ему доступно добавление и редактирование, а также просмотреть таблицу команд, где ему доступны идентичные таблице матчей функции. Выход означает новую авторизацию, а завершение-выход из программы.

Рисунок 1.10-«Макет 7»

Если в авторизации ввести логин и пароль администратора, то будет меню как на рисунке 1.11. Администратору доступно просмотреть количество купленных абонементов и количество купленных абонементов в совокупности и за определенный период. Также он может управлять персоналом, куда входить добавление сотрудника, редактирование его данных и увольнение.

Рисунок 1.11-«Макет 8»

# 2. РАЗРАБОТКА ПРОТОТИПА ИНФОРМАЦИОННОЙ СИСТЕМЫ

## 2.1 РЕАЛИЗАЦИЯ ФУНКЦИЙ ПОКУПАТЕЛЯ

Для покупателя следует создать прототип информационной системы, который бы выполнял все нужные для него функции. Начнём с написания кода для форм, которыми будет пользоваться покупатель. На рисунке 2.1 изображена стартовая форма с выбранным вторым матчем.

При загрузке формы загружаются матчи, которые еще не прошли и билеты на них можно купить. Также есть кнопка Следующая страница, которая позволяет открыть следующие после этих 6 матчей по дате. Для реализации был написан следующий код:

private void Form1\_Load(object sender, EventArgs e)

{

Options op = new Options();

if (op.SoonMatches(0)[0] == "Матч недоступен")

{

kods[0] = 0;

label2.Text = "Матч недоступен";

}

else

{

kods[0] = int.Parse(op.SoonMatches(0)[0]);

label2.Text = op.SoonMatches(0)[1] + "\n" + op.SoonMatches(0)[2] + "\n" + op.SoonMatches(0)[3];

}

Здесь показано как матч отображается на экране. Идет обращение к классу Options, в котором есть метод SoonMatches, принимающий на вход числовое значение и возвращающий массив строк. Код для него выглядит следующим образом:

public string[] SoonMatches(int i)

{

var query = (from m in db.match

join t in db.team on m.team\_id equals t.team\_id

where m.date\_match>DateTime.Now

orderby m.date\_match

select new { m.match\_id,m.type,m.date\_match,t.name,t.city}).ToList();

int max = query.Count();

string[] str= new string[4];

if (i >= max)

{

str[0]= "Матч недоступен";

}

else

{

str[0] = query[i].match\_id.ToString();

str[1] = query[i].type;

str[2] = query[i].date\_match.ToLongDateString() + " " + query[i].date\_match.ToShortTimeString();

str[3]=" Рубин (Казань) - " + query[i].name + " (" + query[i].city + ")";

}

return str;

}

Данные из базы данных также берутся из класса Options через строку-

public static ftEntities1 db = new ftEntities1();

При нажатии на любой из матчей срабатывает следующий код:

private void label3\_Click(object sender, EventArgs e)

{

this.Hide();

if (label3.Text == "Матч недоступен") MessageBox.Show("На данный матч купить билет нельзя");

else if (label11.Text == "Перейти на следующую страницу матчей")

{

kod\_match = kods[1];

Stadium st = new Stadium(kod\_match);

st.Show();

}

else if (label11.Text == "Перейти на предыдущую страницу матчей")

{

kod\_match = kods[7];

Stadium st = new Stadium(kod\_match);

st.Show();

}

}

Если матч недоступен, то выйдет сообщение об этом. В противном случае открывается форма со стадионом для выбора сектора. У данной формы есть входной параметр-код матча, который нужен в дальнейшем для добавления билета в базу данных. Форма стадиона выглядит так, как показано на рисунке 2.2.

Рисунок 2.2-«Форма стадиона»

На этой форме 35 кнопок, при нажатии на которые открывается нужный сектор. Также рядом с подписями о типе трибун (северной, западной и т.д.) написана стоимость минимального билета на ней. При нажатии на любой их секторов срабатывает следующий код:

public Stadium(int kom)

{

InitializeComponent();

this.kod\_m = kom;

}

public int ChSector;

public int kod\_m;

private void button23\_Click(object sender, EventArgs e)

{

ChSector = 4;

Mesta ms = new Mesta(kod\_m, ChSector);

ms.Show();

}

При инициализации формы происходит приравнивание переданного первой форма кода матча к публичной целочисленной переменной kod\_m. Переменная ChSector служит для передачи на форму мест значения сектора. При нажатии на сектор открывается форма мест, входные параметры для которой-код матча и номер сектора. Форма мест выглядит следующим образом-рисунок 2.3.

Черными цветами отмечены занятые места. Около ряда подписано, какой это ряд, над местами повышенного комфорта подписано vip-места. Загрузка данной формы реализована следующим образом:

private void Mesta\_Load(object sender, EventArgs e)

{

Options op = new Options();

Form1 f = new Form1();

if (op.CheckPlace(kom, sk, 1, 1) == true)

{

button1.BackColor = Color.Black;

button1.Enabled = false;

}

Метод CheckPlace проверяет занято ли место и находится в классе Options. Входные параметры для него-переданные с формы стадиона код матча и номер сектора, а также значения ряда и места, которые для каждой кнопки на форме свои. Данный метод имеет следующий код:

public bool CheckPlace(int match\_num, int num\_sec, int num\_row, int num\_pl)

{

bool flag=false;

var query = (from p in db.places

select p).ToList();

if (match\_num==0)

{

var query2 = (from m in db.match

where m.date\_match > DateTime.Now

orderby m.date\_match

select m.match\_id).ToList();

match\_num = query2[0];

}

foreach (places p in query)

{

if ((num\_sec ==p.sector) && (num\_row == p.row) && (num\_pl == p.num\_place) && (match\_num == p.match\_id)) flag = true;

}

return flag;

}

Он возвращает значение true, если место занято и false, если свободно. При нажатии на место выходит сообщение с ценой так как показано на рисунке 2.4.

Рисунок 2.4-«Завершение покупки билета»

При совершении покупки билета срабатывает следующий код.

private void button16\_Click(object sender, EventArgs e)

{

Options op = new Options();

Form1 f = new Form1();

MessageBox.Show(op.BuyTicket(kom,sk,2, 8, "обычный", false, 1));

button16.Enabled = false;

ChooseD cd = new ChooseD();

cd.Show();

button16.BackColor = Color.Black;

}

Место становится занятым, и вызывается метод BuyTicket из класса. Код метода следующий:

public string BuyTicket(int id\_match,int num\_sec,int num\_row,int num\_pl,string type\_tick,bool abon,int emp\_code)

{

if (id\_match == 0) abon = true;

if ((num\_sec < 6) || (num\_sec > 31)) cost += 150;

if (((num\_sec > 5)&&(num\_sec < 15))||((num\_sec>22)&&(num\_sec<32))) cost += 50;

if ((num\_sec >14) && (num\_sec < 23)) cost += 100;

if (num\_row == 3) cost += 50;

if (num\_row == 6) cost += 1500;

if (abon == true) cost = 10000;

int number\_pl;

if (db.places.Count() == 0) number\_pl = 1;

else number\_pl = db.places.Max(n => n.place\_id) + 1;

if (abon == false)

{

places new\_pl = new places { place\_id = number\_pl, match\_id = id\_match, sector = num\_sec, row = num\_row, num\_place = num\_pl, type = type\_tick, season\_ticket = abon, price = cost };

db.places.Add(new\_pl);

db.SaveChanges();

ps = db.places.OrderBy(o => o.place\_id).ToList();

int number\_t;

if (db.ticket.Count() == 0) number\_t = 1;

else number\_t = db.ticket.Max(n => n.ticket\_id) + 1;

ticket new\_t = new ticket { ticket\_id = number\_t, place\_id = number\_pl, emp\_id = emp\_code };

db.ticket.Add(new\_t);

db.SaveChanges();

ts = db.ticket.OrderBy(o => o.ticket\_id).ToList();

return "Вы приобрели билет на матч. С вас " + cost + " рублей. Оплатите в кассе";

}

else

{

var query2 = (from m in db.match

where m.date\_match > DateTime.Now

orderby m.date\_match

select m.match\_id).ToList();

int number\_t;

for (int i=0;i<query2.Count;i++)

{

places new\_pl = new places { place\_id = number\_pl, match\_id = query2[i], sector = num\_sec, row = num\_row, num\_place = num\_pl, type = type\_tick, season\_ticket = abon, price = cost };

db.places.Add(new\_pl);

db.SaveChanges();

ps = db.places.OrderBy(o => o.place\_id).ToList();

if (db.ticket.Count() == 0) number\_t = 1;

else number\_t = db.ticket.Max(n => n.ticket\_id) + 1;

ticket new\_t = new ticket { ticket\_id = number\_t, place\_id = number\_pl, emp\_id = emp\_code };

db.ticket.Add(new\_t);

db.SaveChanges();

ts = db.ticket.OrderBy(o => o.ticket\_id).ToList();

number\_pl++;

number\_t++;

}

return "Вы приобрели абонемент на сезон. С вас " + cost + " рублей. Оплатите в кассе";

}

Сначала происходит расчёт стоимости, исходя из места, которое было куплено. Затем идет добавление места в таблицу Бронь мест, далее-добавление билета в таблицу Билет. Метод возвращает строку, в которой хранится стоимость купленного билета. Покупка билета завершена.

Покупка абонемента состоит из всех вышеперечисленных этапов, однако добавление в базу данных происходит иначе. Метод BuyTicket идет по другой ветви if и добавляет в таблицу места со значением season\_ticket true.

Сдача билета покупателем происходит благодаря работе авторизированного кассира, поэтому данный вариант использования будет рассмотрен позже.

## 2.2 РЕАЛИЗАЦИЯ ФУНКЦИЙ КАССИРА

Работа кассира начинается с авторизации. Форма авторизации выглядит как показано на рисунке 2.5.

Рисунок 2.5-«Авторизация»

Для реализации авторизации был написан следующий код.

Options op = new Options();

string role=op.CheckUser(textBox1.Text,textBox2.Text);

if (role=="кассир")

{

this.Hide();

Ticketer t = new Ticketer();

t.Show();

}

else if (role=="оператор базы данных")

{

this.Hide();

DBOperator dbop = new DBOperator();

dbop.Show();

}

else if (role=="администратор")

{

this.Hide();

; Admin a = new Admin();

a.Show();

}

else if (role=="ошибка")

{

MessageBox.Show("Неверный логин или пароль");

textBox1.Text = "";

textBox2.Text = "";

}

В классе Options есть метод, который проверяет правильность логина и пароля и в успешном варианте возвращает роль сотрудника. Код приведен ниже:

public string CheckUser(string l, string p)

{

workers = (from emps in db.employer select emps).ToList();

int count = 0;

foreach (var emps in workers)

{

if ((emps.login == l) && (emps.password == p))

{

count++;

role = emps.role;

name = emps.name;

patronymic = emps.patronymic;

}

}

if (count == 0)

{

return "ошибка";

}

else return role;

}

После того, как кассир вошел в систему он видит меню, показанное на рисунке 2.6. Рисунок 2.6-«Меню кассира»

Здесь сотрудник вносит данные для сдачи билета, которые преподносит ему покупатель. При успешной проверки возможности сдачи билета, выводится сообщение об этом. В противном случае выводится сообщение об ошибке. Также кассир может выйти из своего аккаунта или вовсе завершить работу, нажав соответствующие кнопки. Код для сдачи билета, находящийся в классе Options, выглядит следующим образом:

public void DeletePlace(int match\_num, int num\_sec, int num\_row, int num\_pl)

{

var delete\_id =(from p in db.places

where (p.match\_id == match\_num)&&(p.sector==num\_sec)&&(p.row==num\_row)&&(p.num\_place==num\_pl)

select p.place\_id).ToList();

var dlt = delete\_id[0];

var delete\_ticket=(from t in db.ticket

where t.place\_id==dlt

select t).First();

var delete\_pl = (from t in db.places

where t.place\_id == dlt

select t).First();

db.ticket.Remove(delete\_ticket);

db.places.Remove(delete\_pl);

db.SaveChanges();

}

## 2.3 РЕАЛИЗАЦИЯ ФУНКЦИЙ ОПЕРАТОРА БАЗЫ ДАННЫХ

После авторизации оператора базы данных он видит картину, изображенную на рисунке 2.7. Рисунок 2.7-«Меню оператора базы данных»

Оператор может открыть таблицу матчей, открыть таблицу команд, выйти или завершить работу. При открытии таблицы команд выходит следующая форма, которая показана на рисунке 2.8.

Рисунок 2.8-«Просмотр таблицы команд»

Для того, чтобы просмотреть эту таблицу реализован следующий код в классе Options:

public void ShowMatches(DataGridView dgv)

{

matches = (from m in db.match select m).ToList();

var query = (from m in matches

join t in db.team on m.team\_id equals t.team\_id

orderby m.match\_id

select new { m.match\_id, t.name, m.type, m.date\_match }).ToList();

dgv.DataSource = query;

dgv.ReadOnly = true;

dgv.Columns[0].HeaderText = "Код матча";

dgv.Columns[1].HeaderText = "Команда соперника";

dgv.Columns[2].HeaderText = "Тип соревнования";

dgv.Columns[3].HeaderText = "Дата матча";

}

Попробуем добавить новую команду Амкар из города Пермь. Добавление представлено на рисунках 2.9, 2.10.

Рисунок 2.9-

«Добавление команды»

Рисунок 2.10-«Добавление команды 2»

Для добавления команды в классе Options есть метод, код которого представлен ниже:

public void AddTeam(string new\_name, string new\_city)

{

int number\_team = db.team.Max(n => n.team\_id) + 1;

team new\_team = new team { team\_id = number\_team, name = new\_name, city = new\_city };

db.team.Add(new\_team);

db.SaveChanges();

}

Теперь попробуем редактировать только что добавленную команду и изменить имя на Сибирь и город на Новосибирск. Результаты на рисунках 2.11, 2.12.

Рисунок 2.11- Рисунок 2.12-«Редактирование матча 2»

«Редактирование матча»

Редактирование осуществляется также с помощью метода из класса Options. Код метода представлен ниже:

public void EditTeam(string t1, string t2)

{

var result = db.team.SingleOrDefault(w => w.team\_id == item.team\_id);

result.name = t1;

result.city = t2;

db.SaveChanges();

}

При просмотре таблицы матчей открываются похожие на то, что было выше формы, идентичные функции и похожий код для методов в классе Options, поэтому опустим демонстрацию данных функций.

## 2.4 РЕАЛИЗАЦИЯ ФУНКЦИЙ АДМИНИСТРАТОРА

При авторизации администратора он видит меню как на рисунке 2.13. Рисунок 2.13-«Меню администратора»

Он может просмотреть количество купленных абонементов, общее количество билетов и количество билетов за определенный период. Также может управлять персоналом, выйти из аккаунта и завершить работу.

Попробуем посмотреть общее количество билетов и проданных на майские игры. Результаты на рисунках 2.14, 2.15.

Рисунок 2.14-«Количество билетов» Рисунок 2.15-«Количество билетов за период»

Для реализации вывода количества купленных билетов реализована перегрузка методов в классе Options. Два метода возвращают по своей сути одинаковые значения, но принимают разные входные параметры. Код представлен ниже:

public int TicketCount()

{

return (from t in db.ticket select t).Count();

}

public int TicketCount(DateTime date1,DateTime date2)

{

return (from t in db.ticket

join p in db.places on t.place\_id equals p.place\_id

join m in db.match on p.match\_id equals m.match\_id

where (m.date\_match >= date1) && (m.date\_match <= date2)

select t).Count();

}

Управление персоналом включает в себя добавление сотрудника, редактирование его данных, а также увольнение. Меню управления персоналом выглядит так, как показано на рисунке 2.16.

Рисунок 2.16-«Меню управление персоналом»

На рисунке мы можем видеть информацию о всех сотрудниках. В последней строке мы видим информацию о Дроздове Алексее Михайловиче, но у него нет логина, пароля и роли. Это связано с тем, что данный сотрудник работал раньше, заносил свой код в таблицу билет, но затем был уволен. При увольнении данные остаются, однако логин, пароль и роль стираются и доступа к работе у сотрудника больше нет.

Кнопки Добавить нового сотрудника, редактировать и уволить частично появлялись в курсовом проекте выше, поэтому приведем только методы, которые вызывают эти кнопки.

public void AddWorker(string s,string n,string p,string l,string pw,string r)

{

int number\_w;

if (db.employer.Count() == 0) number\_w = 1;

else number\_w = db.employer.Max(we => we.emp\_id) + 1;

employer new\_emp = new employer {surname=s,name=n,patronymic=p,login=l,password=pw,role=r };

db.employer.Add(new\_emp);

db.SaveChanges();

}

public void EditWorker(string s, string n, string p, string l, string pw, string r)

{

var result = db.employer.SingleOrDefault(w => w.emp\_id == item3.emp\_id);

result.surname = s;

result.name = n;

result.patronymic = p;

result.login = l;

result.password = pw;

result.role = r;

db.SaveChanges();

}

public void DeleteW()

{

var result = db.employer.SingleOrDefault(d => d.emp\_id == item4.emp\_id);

result.login = "";

result.password = "";

result.role = "";

db.SaveChanges();

}

# ЗАКЛЮЧЕНИЕ

В результате выполнения курсового проекта был создан прототип информационный системы для билетной кассы. Ее разработка позволила автоматизировать и упростить процесс покупки билетов и абонементов, регламентировать их цены, удобном администрировать и организованно управлять персоналом, а также отслеживать информацию о посещаемости матчей.

Задачи проекта были выполнены-была проанализирована предметная область, определены функциональные требования к системе, построены макеты прототипа системы, построены диаграммы вариантов использования, диаграммы классов, представлена схема базы данных в виде ER-модели, разработан и представлен прототип информационной системы.

По итогу всего вышесказанного можно сделать несколько предложений по применению. Добавить к проекту возможность оплаты картой, печать билета, расширить базу данных и хранить эмблемы команд и фотографии сотрудников, а также одновременно покупать несколько билетов.