

Содержание

1	Аналитический раздел	4
1.1	Основные определения	4
1.2	Принцип работы букмекерской конторы	4
1.3	Основные роли в букмекерской конторе	6
1.4	Цикл одного события в букмекерской конторе	7
1.5	Требования к базе данных букмекерской конторы	8
1.6	Основные отношения в базе данных	9
2	Конструкторский раздел	12
2.1	Проектирование базы данных	12
2.2	Тип приложения	15
2.3	Проектирование приложения	16
2.4	Алгоритм работы программы при аутентификации пользователя	18
2.5	Алгоритм работы при обновлении состояния матча	20
3	Технологическая часть	22
3.1	Средства реализации программного обеспечения	22
3.1.1	Выбор языка программирования	22
3.1.2	Выбор СУБД	22
3.1.3	Выбор ПО для графического интерфейса	23
3.2	Листинги кода	23
3.2.1	СУБД	23
3.2.2	Приложение	29
3.3	Демонстрация работы программы	29
3.3.1	Вход в систему	30
3.3.2	Регистрация	30
3.3.3	Стартовое меню игрока	32
3.3.4	Пополнение баланса	32
3.3.5	Выполнение ставок	33
3.3.6	Просмотр истории ставок	34
3.3.7	Добавление матчей в линию аналитиком	34
3.3.8	Экран изменения матчей	35

3.3.9	Экран верификации пользователей	36
4	Экспериментальная часть	37
4.1	Технические характеристики	37
4.2	Зависимость времени ответа базы данных от индексации	37
	Список литературы	40

Введение

Букмекерская контора - игорное заведение, в котором организатор азартных игр заключает пари с участниками данного вида азартных игр.

Букмекерские конторы имеют огромное влияние на спорт: они заключают крупные спонсорские контракты с соревнованиями и клубами, вовлекают большое количество болельщиков к просмотру матчей, а также проводят собственные турниры. Согласно статистике, более 6 миллионов человек в России сделало хотя бы одну ставку в течение года.

Актуальность работы состоит в том, что букмекерская контора требует постоянного взаимодействия с базой данных, которую требуется эффективно спроектировать.

Цель курсовой работы: разработать платформу для обеспечения работы букмекерской конторы, в которой пользователи могут заключать пари на виртуальные события.

Для достижения поставленной цели требуется выполнить следующие задачи:

1. Провести формализацию задачи и определить требуемый функционал.
2. Провести анализ СУБД и описать структуру БД, включая объекты, из которых она состоит.
3. Спроектировать приложение для доступа к БД.
4. Создать и заполнить БД.
5. Реализовать понятный интерфейс для клиента.
6. Разработать ПО, позволяющее пользователям совершать интерактивные ставки на различные события в букмекерской конторе.

1 Аналитический раздел

В этом разделе будут рассмотрены определения букмекерской конторы, принципы её функционирования и требования к базе данных, обеспечивающих её работу.

1.1 Основные определения

В статье 4 Федерального Закона от 29.12.2006 N 244-ФЗ (ред. от 02.07.2021) "О государственном регулировании деятельности по организации и проведению азартных игр и о внесении изменений в некоторые законодательные акты Российской Федерации" содержатся следующие определения [1]:

Азартная игра – основанное на риске соглашение о выигрыше, заключенное двумя или несколькими участниками такого соглашения между собой либо с организатором азартной игры по правилам, установленным организатором азартной игры [1].

Пари – азартная игра, при которой исход основанного на риске соглашения о выигрыше, заключаемого двумя или несколькими участниками пари между собой либо с организатором данного вида азартной игры, зависит от события, относительно которого неизвестно, наступит оно или нет [1].

Интерактивная ставка - денежные средства, в том числе электронные денежные средства, передаваемые с использованием электронных средств платежа, в том числе посредством информационно-телекоммуникационных сетей, включая сеть "Интернет"[1].

Букмекерская контора - игорное заведение, в котором организатор азартных игр заключает пари с участниками данного вида азартных игр [1].

1.2 Принцип работы букмекерской конторы

Букмекерская контора предлагает игрокам заключить пари на различные события. Тип событий может быть различным, но почти всегда букмекерская линия представлена спортивными соревнованиями. Спорт доминирует на рынке, так как является непредсказуемым, матчи происходят каждый день, и по

всему миру живёт огромное количество болельщиков.

Букмекерская контора предлагает коэффициенты на возможные исходы события. Например, в случае футбольного матча такими исходами является победа одной из команд или ничья. Игрок имеет возможность заключить с конторой пари на наступление какого-либо исхода, сделав ставку. Если этот исход наступил, то букмекерская контора должна вернуть игроку денежные средства на сумму, умноженную на коэффициент исхода. В противном случае букмекерская контора оставляет сумму ставки у себя [2].

Рассмотрим первое приближение выбора коэффициента. Пусть p - вероятность наступления какого-либо исхода. Тогда коэффициент рассчитывается по формуле 1.1:

$$k = \frac{1}{p} \quad (1.1)$$

Пусть N – количество ставок игрока, а S – сумма одной ставки. Математическое ожидание заработка игрока можно посчитать по формуле 1.2:

$$M = k_1 * S * p * N - S * N = \frac{1}{p} * S * p * N - S * N = 0 \quad (1.2)$$

Так как выигрыш букмекера формируется из проигрыша игрока, из 1.2 следует, что и математическое ожидание заработка букмекера тоже равняется нулю.

Но букмекерская контора рассчитывает зарабатывать и в краткосрочной, и в долгосрочной перспективе. Для получения прибыли в каждом матче в коэффициенты закладывается маржа. Пусть p_{win} - процент маржи, которые контора хочет иметь, а ω – количество исходов события. Тогда коэффициент рассчитывается по формуле 1.3:

$$k = \frac{1}{p + \frac{p_{win}}{\omega}} \quad (1.3)$$

В таком случае математическое ожидание заработка игрока будет отрицательным, так как коэффициент стал меньше, чем в 1.1. Исходя из этого, букмекерская контора остаётся в выигрыша при наступлении любого исхода.

Для того, чтобы постоянно зарабатывать, букмекерам требуется безошибочно рассчитывать вероятности наступления всех событий. Этим обычно занимаются профессиональные аналитики и статистики.

Теперь рассмотрим событие, в котором есть два противоположных исхода

– выигрыш одной из команд. Пусть k_1 и k_2 – коэффициенты на эти исходы, а S_1 и S_2 – денежная сумма, поставленная на эти исходы. Если реализуется первый исход, то заработок БК составит $S_1 + S_2 - k_1 * S_1$, а если второй исход – $S_1 + S_2 - k_2 * S_2$.

Денежные потоки могут быть распределены таким образом, что их распределение не будет соответствовать рассчитанным вероятностям наступления событий. В таком случае при наступлении одного из событий БК потеряет деньги. Следовательно, требуется постоянно обновлять коэффициенты, опираясь на объём средств, поставленных игроками на каждый исход.

При этом всё на букмекерском рынке большая конкуренция, следовательно, требуется предлагать самые выгодные коэффициенты на рынке.

Подводя итог, расчёт коэффициентов на матчи – сложная математическая, экономическая и маркетинговая задача, которая и позволяет букмекерам зарабатывать деньги [2].

1.3 Основные роли в букмекерской конторе

Букмекерская контора заключает пари с игроками, поэтому первая роль, которая требуется – **игрок**. Для игроков требуется добавить возможность просмотра линии событий, заключения пари, пополнение баланса, изменения личных данных. При этом нужно добавить регистрацию, чтобы новый игрок мог воспользоваться аккаунтом, только лишь загрузив приложение.

Согласно закону № 115-ФЗ «О противодействии легализации (отмыванию) доходов, полученных преступным путем, и финансированию терроризма» об азартных играх, букмекерская контора должна обязательно проводить процедуру идентификации клиентов [3]. Обычно в букмекерских конторах есть администраторы, которые проверяют указанные игроком данные, и верифицируют аккаунты. Лишь после этого у клиента есть возможность совершать ставки на какие-либо события. Поэтому следующая роль – **администратор**.

В его функционал входит верификация новых пользователей или отклонение заявки игрока. Подразумевается, что администратор имеет доступ к какой-то сторонней базе данных, где может проверить паспортные данные, однако это выходит за пределы курсовой работы. Поэтому администратор просто имеет возможность изменить статус игрока.

Также в букмекерской конторе кто-то должен заниматься линией матчей. Требуется добавлять новые матчи в линию, изменять их статус (событие началось, закончилось), менять текущий счёт и изменять коэффициенты. Этим занимается **аналитик**, который имеет возможность добавлять события в линию и менять о них информацию.

Use-Case диаграмма ролей проекта представлена на рисунке 1.1:

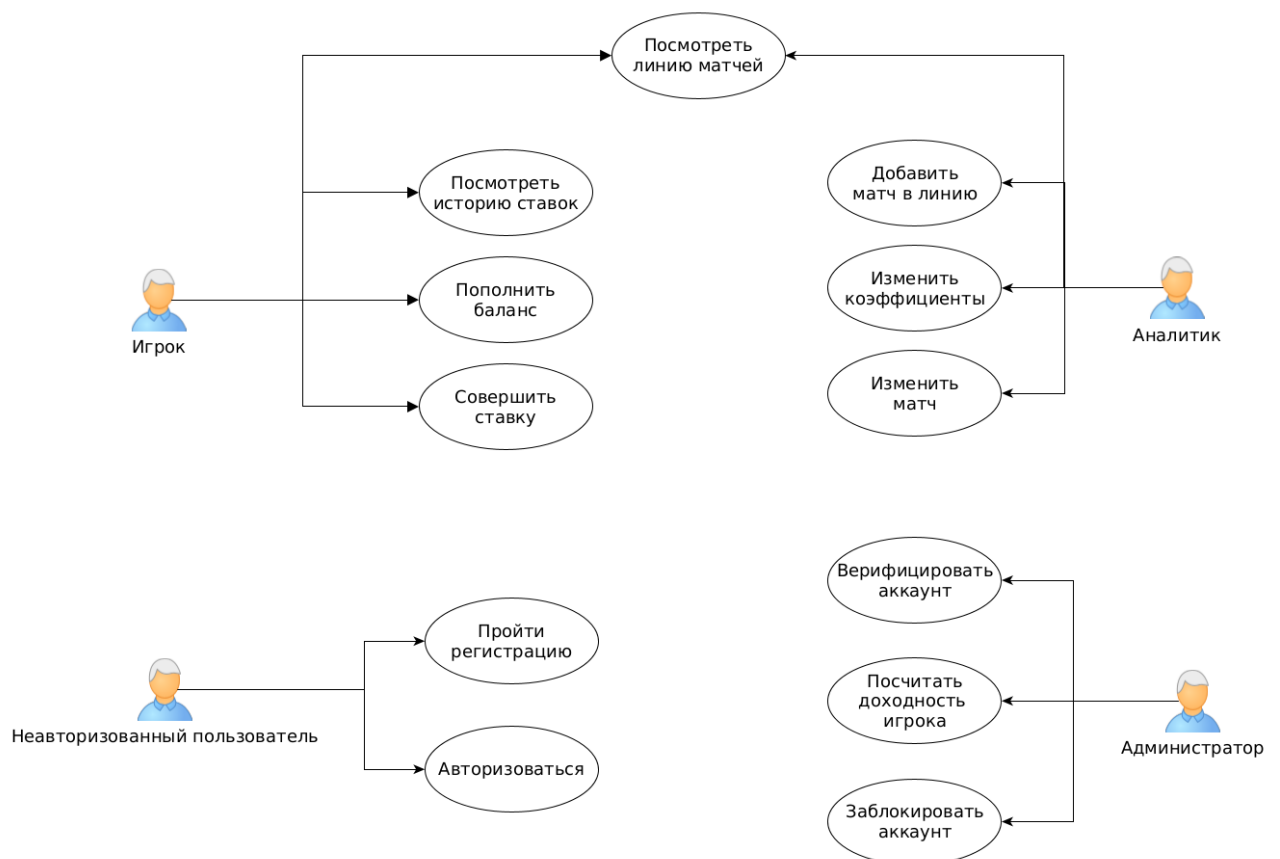


Рисунок 1.1: Use-Case диаграмма

1.4 Цикл одного события в букмекерской конторе

Для обеспечения работы букмекерской платформы требуется, чтобы игроку был предоставлен выбор из событий. Требуется, чтобы аналитик добавил событие в букмекерской конторе. Событие происходит в абстрактном виде спорта, в котором возможно три исхода: победа первой команды, ничья, и победа второй

команды. У каждого события есть дата и время начала, две команды, которые встречаются в матче, а также коэффициенты на каждый из исходов. Аналитик должен указать все данные события. Для упрощения работы, список команд должен быть постоянным, и храниться в базе данных. При этом в одном событии не может быть двух одинаковых команд, а коэффициенты на исход должны соответствовать теоретическим изложениям из п.1.2 настоящего РПЗ.

У каждого события есть атрибут статуса. Всего есть три варианта:

1. Событие запланировано, но ещё не началось.
2. Событие идёт в текущий момент.
3. Событие закончилось.

Аналитику требуется своевременно вносить правки в статус матча: изменять статус события, изменять счёт встречи, а также менять коэффициенты в зависимости от того, как складывается встреча.

Как только аналитик пометил событие законченным, у матча меняется статус на "закончен и букмекерская контора должна рассчитать тех, кто выиграл пари, и заплатить им.

1.5 Требования к базе данных букмекерской конторы

Платформа для обеспечения работы букмекерской конторы должна использовать базу данных для нескольких целей:

1. Хранения информации о пользователях, их балансе и личной информации.
2. Хранения информации о событиях в линии, причём как запланированных и текущих, так и о законченных.
3. Хранения информации о совершённых игроком ставок для прозрачности работы конторы.

4. Автоматического обновления баланса игроков при заключении пари и перерасчёта в случае выигрыша.
5. Хранения информации об аккаунтах и ролях, связанных с ними.
6. Различий в возможностях ролей при работе с базой данных.

Также требуется обратить внимание на следующие детали при разработке базы данных:

- данные в букмекерской конторе не должны быть избыточны, следовательно, требуется привести отношения к нормальной форме;
- букмекерская контора хранит персональные данные пользователей, следовательно, требуется обеспечить безопасность работы с паспортными данными, паролями, чтобы их было невозможно получить нежелательным лицам;
- так как состояние матча также хранится в БД, автоматическое обновление баланса игроков можно реализовать при помощи триггеров на определённые события;
- ставка может считаться сделанной, если информация о ней записана в БД и с баланса игрока списаны средства, равные размеру ставки. При этом коэффициент может измениться во время работы системы. Поэтому требуется использовать транзакции для поддержки функционала заключения пари.

1.6 Основные отношения в базе данных

В соответствии с вышеперечисленными аспектами базы данных для платформы, в БД должны быть представлены следующие отношения с атрибутами:

1. **Аккаунты.** В качестве информации должен храниться логин и пароль для входа в систему, а также роль, соответствующая аккаунту. Это отношение может быть изменено напрямую только администратором. Также оно автоматически обновляется при регистрации нового игрока.

2. **Пользователи.** В этом отношении должна храниться основная личная информация о пользователе: номер телефона, электронная почта, номер паспорта. Также оно содержит информацию о балансе игрока и максимальной ставке. Предполагается, что букмекерская контора будет иметь возможность порезать счёт успешного игрока, существенно уменьшив собственные убытки.
3. **Команды.** В этом отношении должны храниться команды, которые могут быть в событии. В качестве атрибутов рассматривается название, логотип и город.
4. **События.** В этом отношении хранятся все события: дата создания, их статус, итоговый счёт, команды, принимающие участия.
5. **Ставки.** В этом отношении хранятся все совершённые ставки и события.

Концептуально отношения в базе данных можно представить с помощью ER-модели нотации Чена. На рисунке 1.2 представлена ER-модель в нотации Чена:



Рисунок 1.2: ER-модель в нотации Чена

2 Конструкторский раздел

В этом разделе будет проведено проектирование базы данных и приложения, а также приведены схемы разрабатываемых алгоритмов.

2.1 Проектирование базы данных

База данных приложения будет реализована с помощью следующих страниц:

1. таблица пользователей User;
2. таблица игроков Account;
3. таблица команд Teams;
4. таблица матчей Games;
5. таблица ставок Bet;

Таблица пользователей **User** содержит информацию о логинах, паролях аккаунтов, а также о ролях, соответствующих аккаунту. Содержит следующие поля:

- id – первичный ключ, идентификатор аккаунта;
- userLogin – логин пользователя;
- password – пароль пользователя;
- creationDate – дата регистрации аккаунта;
- userRole – роль, соответствующая аккаунту.

Таблица пользователей **Account** содержит информацию о личности игрока, который совершает ставки в букмекерской конторе. Содержит следующие поля:

- accID – первичный ключ, идентификатор аккаунта;

- **userID** – внешний ключ, соответствует идентификаторам интернет аккаунта;
- **userName** – имя пользователя, указанное при регистрации;
- **userSurname** – фамилия пользователя, указанное при регистрации;
- **dateOfBirth** – дата рождения пользователя;
- **phoneNumber** – номер телефона пользователя;
- **email** – электронная почта пользователя;
- **passportNumber** – номер паспорта пользователя;
- **userStatus** – статус пользователя: на верификации, активный и заблокированный аккаунты;
- **balance** – текущий баланс пользователя;
- **maxBet** – размер максимальной ставки, доступной игроку.

Таблица команд **Teams** содержит информацию о командах, которые могут принимать участие в матче. Содержит следующие поля:

- **teamID** – первичный ключ, идентификатор команды;
- **teamName** – название команды;
- **teamcity** – город, из которого команда;
- **logo** – локальный путь к картинке эмблемы команды.

Таблица матчей **Games** содержит информацию о матчах, на которые принимаются ставки в букмекерской конторе. Содержит следующие поля:

- **gameID** – первичный ключ, идентификатор матча;
- **gameStatus** – соответствует текущему состоянию матча: запланирован, идёт или закончился;
- **team1ID** – внешний ключ, соответствует id первой команды;

- team2ID – внешний ключ, соответствует id второй команды;
- w1Coef – текущий коэффициент на победу первой команды;
- drawCoef – текущий коэффициент на ничью;
- w2Coef – текущий коэффициент на победу второй команды;
- gameResult – текущий счёт в матче;
- gameDate – дата проведения матча;
- gameTime – время начала матча;

Таблица ставок **Bet** содержит информацию о совершённых пользователями ставках. Содержит следующие поля:

- betID – первичный ключ, идентификатор ставки;
- gameId – внешний ключ, соответствует идентификатору матча;
- accID – внешний ключ, соответствует id пользователя, который совершил ставку;
- choosedResult – выбранный результат (0 – ничья, 1 – победа первой команды, 2 – победа второй команды);
- betDate – время совершения ставки;
- betSize – размер совершенной ставки;
- koef – коэффициент, по которому игрок совершил ставку;
- betStatus – текущее состояние ставки: принята, выиграна или проиграна;
- payoutAmount – выплата по результатам совершения события;

Схема базы данных представлена на рисунке 2.1:

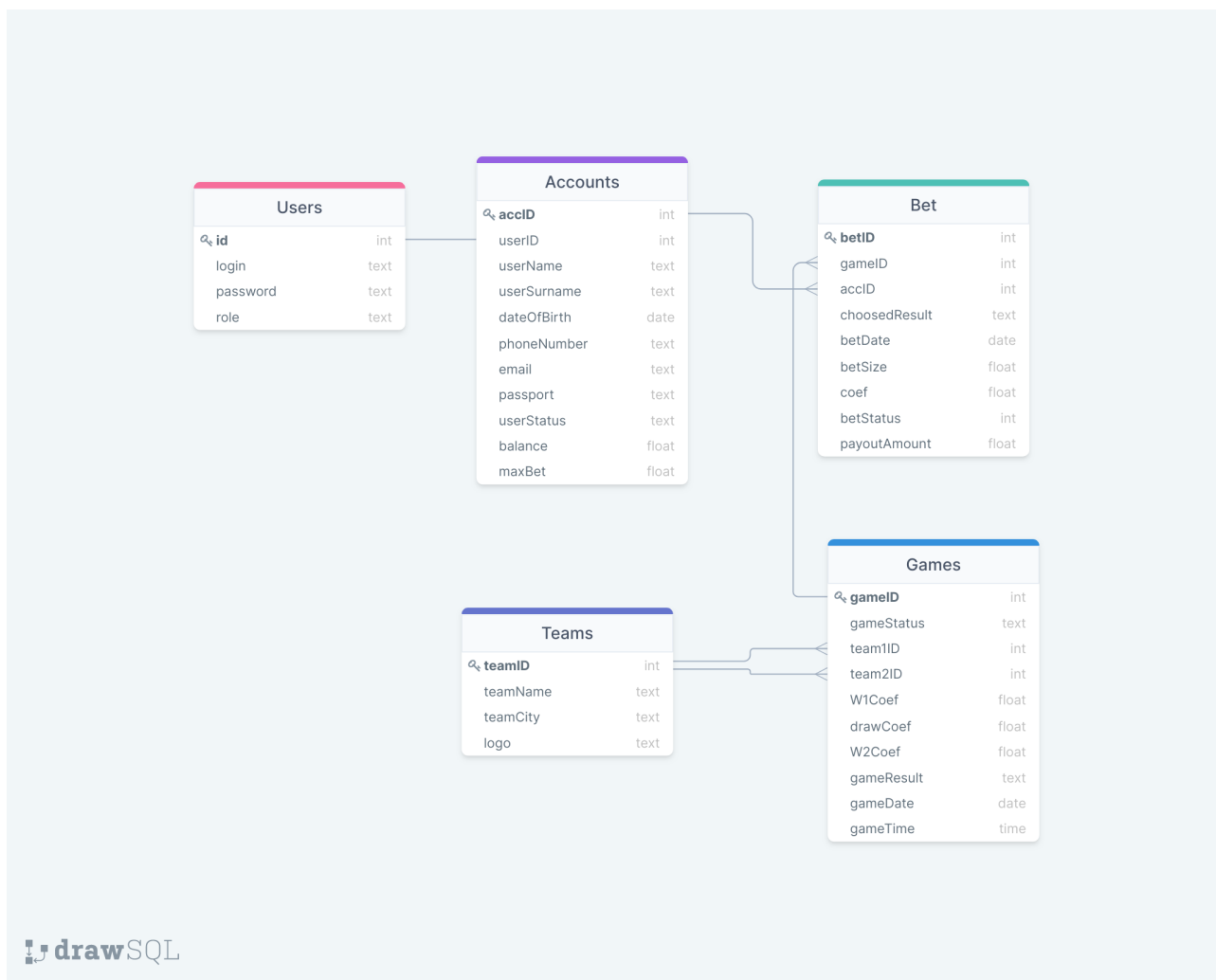


Рисунок 2.1: Схема базы данных

2.2 Тип приложения

Тип приложения был выбран Desktop по двум следующим причинам:

1. Реальные букмекерские конторы разрабатывают версии программного обеспечения для ПК, так как это позволяет пользователям удобнее и быстрее взаимодействовать с системой.
2. Desktop-приложение не предоставляет дополнительных требований к быстродействию, как, например, высоконагруженное Web-приложение.

2.3 Проектирование приложения

Структура приложения основана на парадигмах ООП. Для реализации работы приложения были реализованы следующие классы:

- class UI – класс содержит функции для загрузки новой страницы формата .ui и создания контроллера под неё;
- class Controllers – набор классов, каждый из которых контролирует связь элементов страниц друг с другом;
- class Facade – класс, является реализацией паттерна ”Фасад”;
- class Command – набор классов, являются реализацией паттерна ”Команда”;
- class AuthManager – класс, являющийся сущностью между действий в UI и базой данных для неавторизованного пользователя;
- class PlayerManager – класс, являющийся сущностью между действий в UI и базой данных для вошедшего в систему игрока;
- class AdminManager – класс, являющийся сущностью между действий в UI и базой данных для администратора БК;
- class AnalyzerManager – класс, являющийся сущностью между действий в UI и базой данных для аналитика;
- class AuthRepo – класс, непосредственно взаимодействующий с базой данных для неавторизованного пользователя;
- class PlayerRepo – класс, непосредственно взаимодействующий с базой данных для авторизованного игрока;
- class AdminRepo – класс, непосредственно взаимодействующий с базой данных для администратора БК;
- class AnalyzerRepo – класс, непосредственно взаимодействующий с базой данных для аналитика;

- class SessionHolder – класс, реализующий паттерн ”Прокси выполняющий хэширование данных текущего игрока. Позволяет не взаимодействовать с БД для получения данных пользователя;
- class ValidateManager – набор классов, валидующих данные: при регистрации, при добавлении нового матча или корректировке коэффициентов.

На рисунке 2.2 представлена диаграмма классов в виде UML-диаграммы.

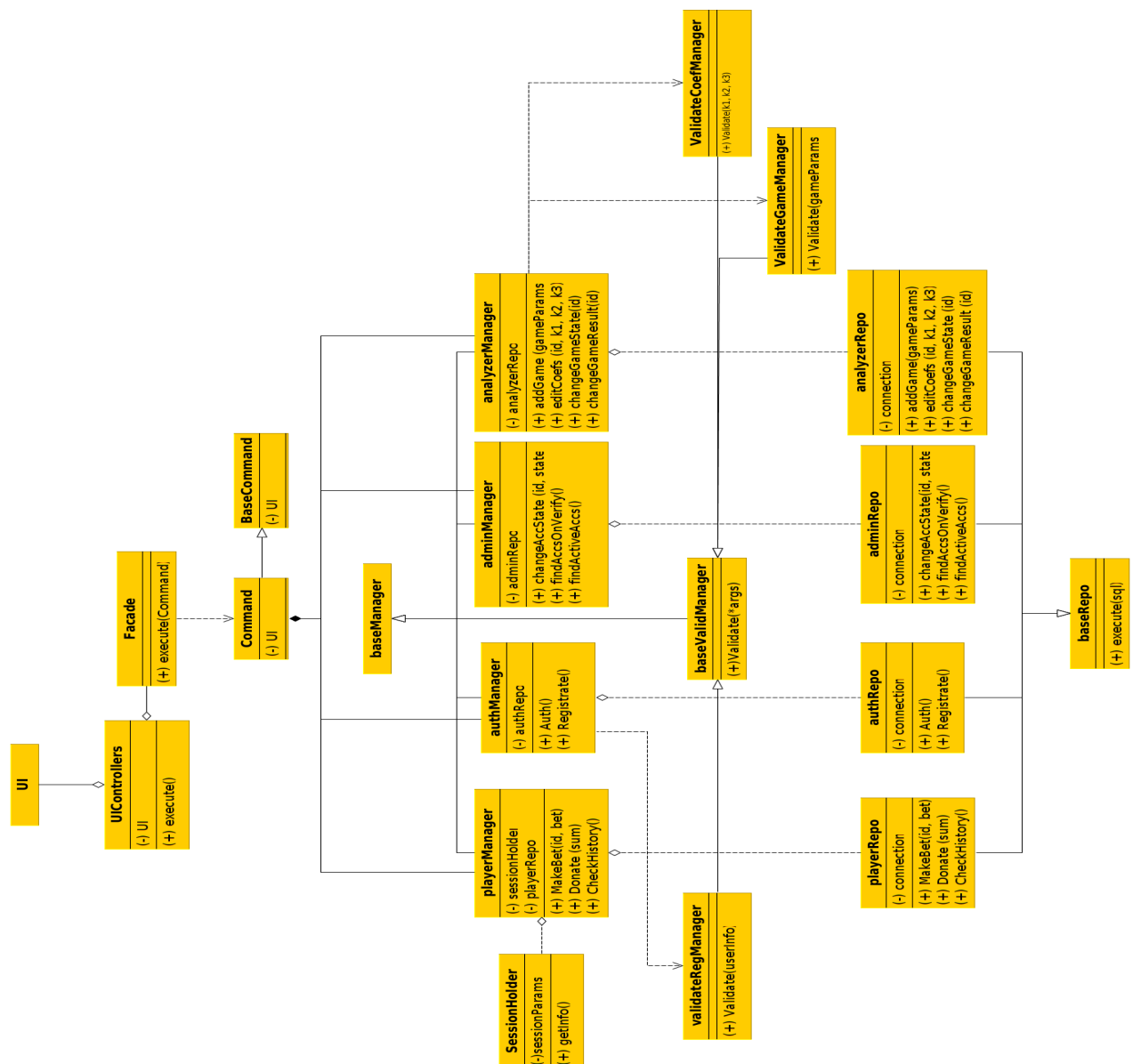


Рисунок 2.2: UML-диаграмма классов

2.4 Алгоритм работы программы при аутентификации пользователя

Аутентификация пользователя происходит при активном взаимодействии с базой данных.

В первую очередь требуется проверить, что логин существует в базе данных. Для этого достаточно найти одно значение логина в базе, при этом гарантируется, что поле логина уникальное, и логин присутствует в единственном экземпляре.

После соответствия требуется проверить соответствие введенного пароля и пароля в системе. Если они совпадают, то можно выполнить вход в систему, иначе вывести сообщение об ошибке.

При входе системы требуется также получить из БД роль, соответствующий аккаунту, в который входит пользователь. Для аналитика и администратора в дальнейшем загружается UI, но для игрока требуется провести несколько дополнительных действий.

Во-первых, требуется выгрузить из БД всю основную информацию о пользователе, и кэшировать её. Это нужно для того, чтобы в дальнейшем не обращаться к БД для проверки параметров (например, хватает ли у пользователя средств для совершения ставки), что снижает время работы с самой базой данных. Во-вторых, требуется проверить, какой статус у аккаунта. Если у пользователя заблокированный аккаунт, то он не может выполнять никаких действий, поэтому требуется в UI заблокировать все соответствующие элементы.

На рисунке 2.3 представлена схема алгоритма работы программы при аутентификации:

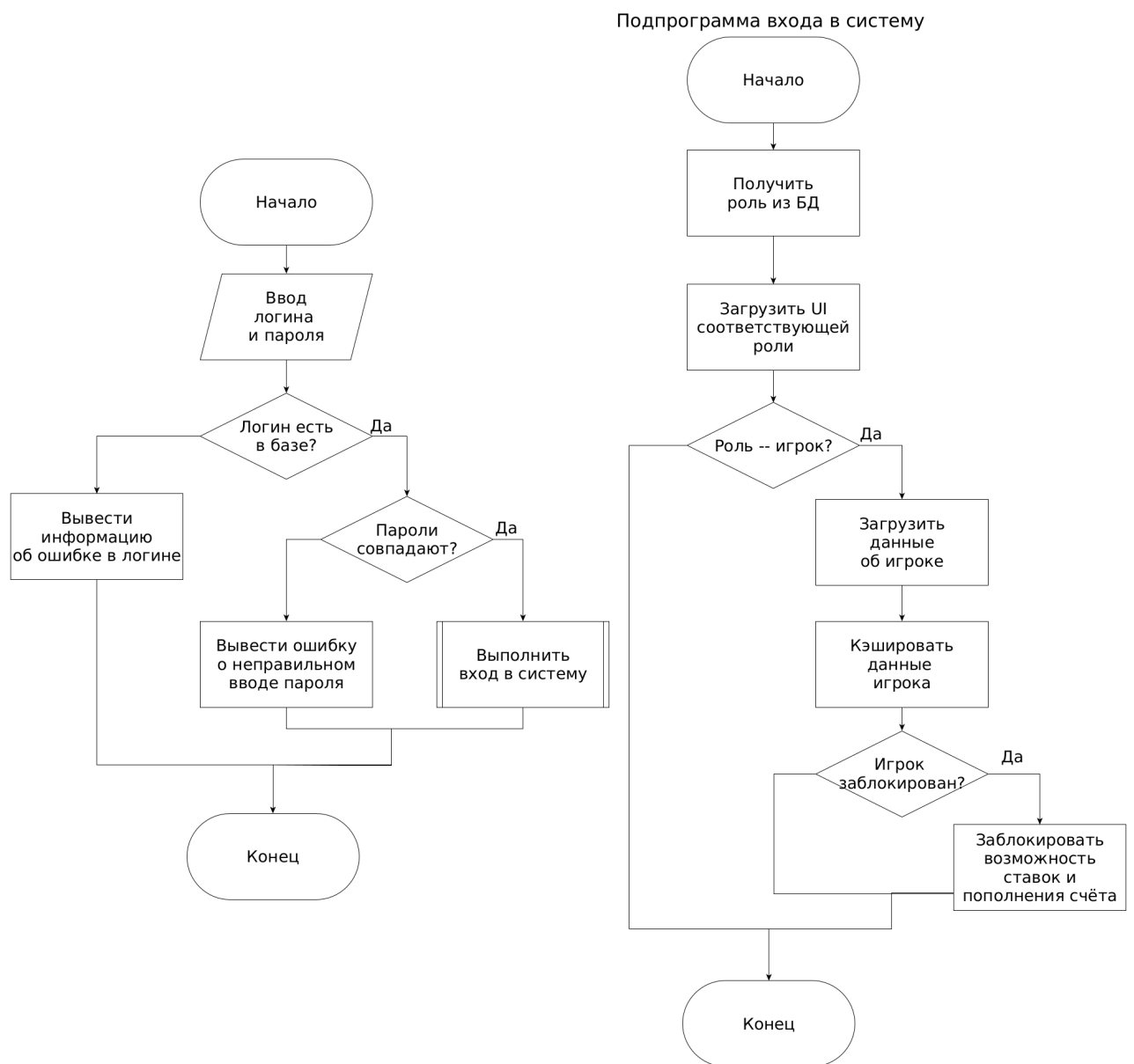


Рисунок 2.3: Схема алгоритма работы программы при аутентификации

2.5 Алгоритм работы при обновлении состояния матча

Алгоритм работы при обновлении состояния матча является важнейшим для работы системы, так как расчёт выплат по ставкам производится по окончанию матча.

Для автоматического обновления баланса пользователей требуется разработать триггер. Он будет навешан на таблицу Bet, но сложность заключается в том, что баланс находится в другом отношении – Account. Следовательно, требуется вызывать подпрограмму для обновления баланса пользователей.

Также для определения результата матча требуется реализовать собственную функцию, так как в таблице хранится текущий счёт встречи: в таблице нельзя отдельно хранить результат из-за того, что тогда возникнет избыточность в базе, а счёт требуется для корректной оценки пользователем действий в матче.

Для обновления статуса ставок требуется сравнить результат, выбранный игроком, и реальный. Статус ставки хранится в виде целочисленного флага: 1 соответствует выигрышу, 0 – проигрышу.

Для последовательного прохода по id требуется реализовать курсор, так как требуется изменять баланс пользователей построчно, последовательно итерируясь по всем пользователям, делавших ставки на конкретное событие.

На рисунке 2.4 представлена схема алгоритма обновления состояния матча:

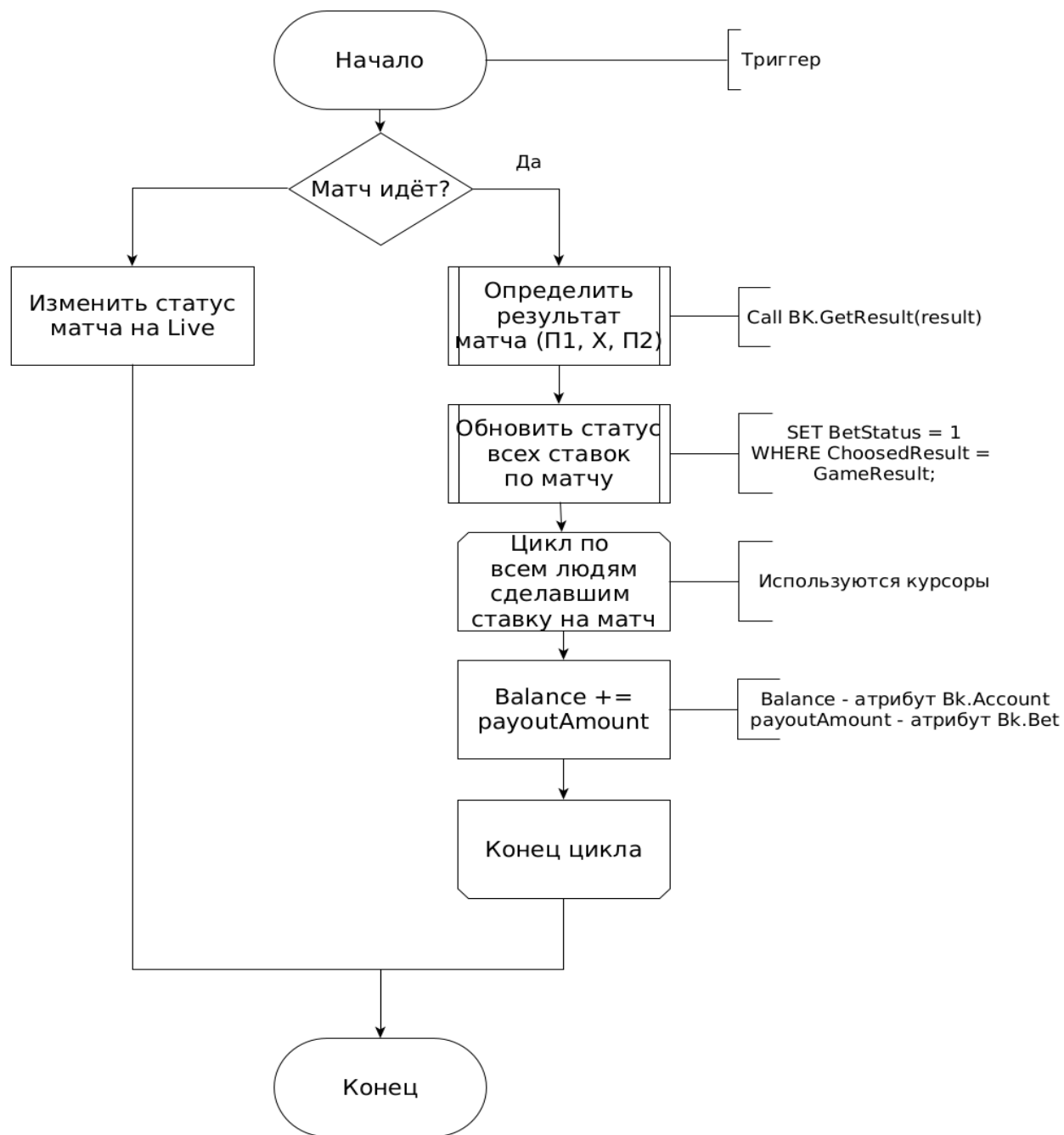


Рисунок 2.4: Схема алгоритма обновления состояния матча

3 Технологическая часть

В этом разделе будет проведен выбор средств реализации ПО, будут приведены листинги кода и демонстрация работы программы.

3.1 Средства реализации программного обеспечения

3.1.1 Выбор языка программирования

В качестве языка программирования для реализации платформы был выбран Python 3.9, так как имеется опыт разработки проектов на этом языке. Также данный ЯП поддерживает парадигму ООП, соответственно, даёт возможность реализовать структуру приложения, приведённую в конструкторской части.

3.1.2 Выбор СУБД

В качестве СУБД был выбран PostgreSQL.

Postgres [5] – это свободно распространяемая объектно-реляционная система управления базами данных, наиболее развитая из открытых СУБД в мире и являющаяся реальной альтернативой коммерческим базам данных [?]. Данная СУБД предназначена для обработки ряда рабочих нагрузок, от отдельных компьютеров до хранилищ данных или веб-сервисов с множеством одновременных пользователей.

Выбор обусловлен тем, что имеется опыт разработки проектов с данной СУБД, а также тем, что Python и PostgreSQL являются совместимыми технологиями.

3.1.3 Выбор ПО для графического интерфейса

В качестве ПО для реализации UI был выбран среда QT по следующим причинам:

1. Qt предоставляет широкую поддержку для взаимодействия с Python-приложениями.
2. Qt является программной средой для реализации Desktop-приложений. Пользователь может воспользоваться ПО QtDesigner для разработки UI.
3. Структура приложений QT позволяет внедрить их в ООП-архитектуру.
4. У автора есть опыт разработки проектов с использованием QT.

3.2 Листинги кода

3.2.1 СУБД

На листинге 3.1 представлен код создания таблиц в базе данных:

Листинг 3.1: Создание таблиц в БД

```
DROP SCHEMA BK cascade;

CREATE SCHEMA BK;

CREATE TABLE BK.Users
(
    UserID SERIAL PRIMARY KEY,
    UserLogin TEXT UNIQUE,
    UserPassword TEXT,
    UserRole TEXT
);

CREATE TABLE Bk.Account
(
    AccountId SERIAL PRIMARY KEY,
    UserID INT,
    UserName TEXT,
    UserSurname TEXT,
    DateOfBirth DATE,
    PhoneNumber TEXT,
```

```

    PassportNumber TEXT,
    Email TEXT,
    CreationDate DATE,
    UserStatus TEXT,
    Balance         FLOAT,
    MaxBet          INT,
    FOREIGN KEY (UserID) references BK.Users(UserID)
);

CREATE TABLE BK.Teams
(
    TeamId SERIAL PRIMARY KEY,
    TeamName TEXT,
    TeamCity TEXT,
    Logo TEXT --
);

CREATE TABLE BK.Games
(
    GameID SERIAL PRIMARY KEY,
    GameStatus TEXT,
    Team1ID INT,
    Team2ID INT,
    W1Coef FLOAT,
    DrawCoef FLOAT,
    W2Coef FLOAT,
    GameResult TEXT,
    GameDate DATE,
    GameTime TIME,
    FOREIGN KEY (Team1ID) references BK.Teams(TeamID) on DELETE CASCADE,
    FOREIGN KEY (Team2ID) references BK.Teams(TeamID) on DELETE CASCADE
);

CREATE TABLE BK.Bet
(
    BetID SERIAL PRIMARY KEY,
    GameID INT,
    ChoosedResult INT,
    AccountID INT,
    BetDate TIMESTAMP,
    BetSize FLOAT,
    Koef FLOAT,
    BetStatus INT,
    PayoutAmount FLOAT,
    FOREIGN KEY (GameID) references BK.Games(GameID),
    FOREIGN KEY (AccountID) references BK.Account(AccountId)
);

```


На листинге 3.2 представлен код создания ограничений на таблицы:

Листинг 3.2: Создание ограничений на таблицы

```
ALTER TABLE Bk.Account
    ADD CONSTRAINT correctBalance CHECK (Balance >= 0);

ALTER TABLE BK.Account
    ADD CONSTRAINT correctDateBirth CHECK (DateOfBirth <= '2003-12-31'::date);

ALTER TABLE BK.Account
    ADD CONSTRAINT correctPassport CHECK (LENGTH(passportNumber) = 10);

ALTER TABLE BK.Account
    ADD CONSTRAINT correctStatus CHECK (UserStatus = 'Active' OR UserStatus = '
    On Verify' OR UserStatus = 'Blocked');

ALTER TABLE BK.Games
    ADD CONSTRAINT correctCoefs CHECK (W1Coef > 1 AND W2Coef > 1 AND DrawCoef >
    1);

ALTER TABLE BK.Games
    ADD CONSTRAINT correctSumCoef CHECK ((1 / W1Coef + 1 / W2Coef + 1 / DrawCoef
    ) > 1);

ALTER TABLE BK.Games
    ADD CONSTRAINT correctStatus CHECK (GameStatus = 'Plain' OR GameStatus = '
    Live' OR GameStatus = 'Finished');

ALTER TABLE BK.Bet
    ADD CONSTRAINT correctBetResult CHECK (ChoosedResult = 0 OR ChoosedResult =
    1 OR ChoosedResult = 2);

ALTER TABLE BK.Bet
    ADD CONSTRAINT correctBetSize CHECK (BetSize >= 10 AND BetSize <= 1000);

ALTER TABLE BK.Bet
    ADD CONSTRAINT correctCoefBet CHECK (Kcoef > 1);

ALTER TABLE BK.Bet
    ADD CONSTRAINT correctBetStatus CHECK (BetStatus = 0 OR BetStatus = 1 OR
    BetStatus = -1);

ALTER TABLE BK.Users
    ADD CONSTRAINT correctLogin CHECK (LENGTH(UserLogin) > 4);

ALTER TABLE BK.Users
    ADD CONSTRAINT correctPassword CHECK (LENGTH(UserPassword) > 4);
```

На листинге 3.3 представлен код получения таблицы матчей в текущей линии:

Листинг 3.3: Получение таблицы матчей в текущей линии

```
CREATE OR REPLACE FUNCTION BK.viewGamesAnalyze(name TEXT)
RETURNS TABLE (
    id int,
    matchdate DATE,
    matchtime TIME,
    t1name TEXT,
    t2name TEXT,
    status TEXT,
    result TEXT,
    w1cf FLOAT,
    xcf FLOAT,
    w2cf FLOAT
)
AS $$
BEGIN
    RETURN QUERY
    SELECT gameid, gamedate, gametime, tmp.teamname, T2.teamname, gamestatus
        , gameresult, w1coef, drawcoef, w2coef
        FROM (BK.Games as G JOIN BK.Teams T on (t.teamid = G.team1id)) as
            tmp
            JOIN BK.Teams as T2 on (tmp.team2id = T2.teamid)
    WHERE (tmp.teamname like name OR T2.teamname like name)
        AND (gamestatus = 'Live' OR gamestatus = 'Plain')
    ORDER BY gamedate, gametime;
END;
$$ language plpgsql;
```

На листинге 3.4 представлен код расчёта итога события:

Листинг 3.4: Расчёт итога события

```
CREATE OR REPLACE FUNCTION BK.GetResult(gameResult TEXT)
RETURNS integer
AS
$$
    DECLARE firstGoal TEXT;
           secondGoal TEXT;
           middlePos INT;
           len INT;
    BEGIN
        len = length(gameResult);
        middlePos = position(': ' in gameResult);
        firstGoal = substring(gameResult from 0 for middlePos);
        secondGoal = substring(gameResult from middlePos + 1 for len);
```

```

        if firstGoal > secondGoal
            then return 1;
        end if;
        if firstGoal < secondGoal
            then return 2;
        end if;
        if firstGoal = secondGoal
            then return 0;
        end if;
    END;
$$ language plpgsql;

```

На листинге 3.5 представлен код триггера на обновления таблицы Games, связанным с изменением состояния матча:

Листинг 3.5: Триггер на обновление таблицы Games

```

CREATE OR REPLACE FUNCTION BK.UpdateBet()
returns trigger
AS
    $$
    BEGIN
        --          ID
        if (new.gamestatus = 'Finished') then
            UPDATE Bk.Bet as B
            SET BetStatus = 1
            WHERE B.GameID = new.GameID AND B.ChoosedResult = Bk.getresult(new.
                GameResult);

            UPDATE Bk.Bet as B
            SET PayoutAmount = B.BetSize * B.koef
            WHERE B.gameid = new.gameid AND betstatus = 1;

            UPDATE BK.Bet as B
            SET betstatus = -1
            WHERE B.GameID = new.GameID AND B.ChoosedResult != Bk.getresult(new.
                GameResult);

            CALL BK.UpdateBalance(new.gameid);
        end if;
        return new;

    END;
    $$ language plpgsql;
DROP trigger if exists UpdateBetTrigger on BK.Games;
CREATE TRIGGER UpdateBetTrigger AFTER UPDATE ON BK.Games
FOR ROW EXECUTE PROCEDURE BK.UpdateBet();

```

На листинге 3.6 представлен код процедуры обновления баланса пользователей:

Листинг 3.6: Обновление балансов пользователей

```
CREATE OR REPLACE PROCEDURE BK.UpdateBalance(id int)
AS
    $$
    DECLARE
        tmpAcc RECORD;
        allAccs CURSOR for
        SELECT Acc.accountid, B.payoutamount
        FROM BK.bet as B JOIN BK.Account as Acc on Acc.accountid = B.
            accountid
        WHERE B.GameID = id and B.betstatus = 1;
    BEGIN
        OPEN allAccs;
        LOOP
            fetch allAccs into tmpAcc;
            UPDATE Bk.account
            SET balance = balance + tmpAcc.payoutamount
            WHERE accountid = tmpAcc.accountid;
            EXIT When not found;
        end loop;
        close allAccs;
    END;
    $$ language plpgsql;
```

На листинге 3.7 представлен код выдачи прав по ролям:

Листинг 3.7: Выдача прав ролям

```
CREATE ROLE Client NOCREATEDB NOSUPERUSER ;
CREATE ROLE Administrator NOSUPERUSER NOCREATEDB;
CREATE ROLE Analyzist NOSUPERUSER NOCREATEDB;

GRANT ALL PRIVILEGES ON BK.games TO Client;
GRANT ALL PRIVILEGES ON BK.bet TO Client;
GRANT USAGE, SELECT ON ALL SEQUENCES IN SCHEMA BK TO Client;

GRANT ALL PRIVILEGES ON BK.games TO Analyzist;
GRANT USAGE, SELECT ON ALL SEQUENCES IN SCHEMA BK TO Analyzist;
GRANT ALL PRIVILEGES ON BK.bet TO Analyzist;
GRANT ALL PRIVILEGES ON BK.teams TO Analyzist;

GRANT ALL PRIVILEGES ON BK.account TO Administrator;
GRANT ALL PRIVILEGES ON BK.users TO Administrator;
GRANT ALL PRIVILEGES ON BK.bet TO Administrator;
```

3.2.2 Приложение

Все действия, которые приложение может выполнить, обернуты в SQL-функции, которые затем вызываются классами, ответственными за взаимодействие с базой данных.

На листинге 3.8 представлен код выполнения SQL-запроса в Python:

Листинг 3.8: Выполнение SQL-запроса

```
def execute(self, executeString):
    errorCode = OK
    result = []
    try:
        self.cur.execute(executeString)
    except Exception as e:
        print(e)
        errorCode = DATABASE_ERROR
        self.connection.rollback()
    else:
        self.connection.commit()

    try:
        result = self.cur.fetchall()
    except:
        pass

    return errorCode, result
```

На листинге 3.9 представлен код пополнения баланса пользователя:

Листинг 3.9: Пополнение баланса пользователя

```
def Donate(self, id, value):
    executeString = f"CALL BK.Donate({id}, {value});"

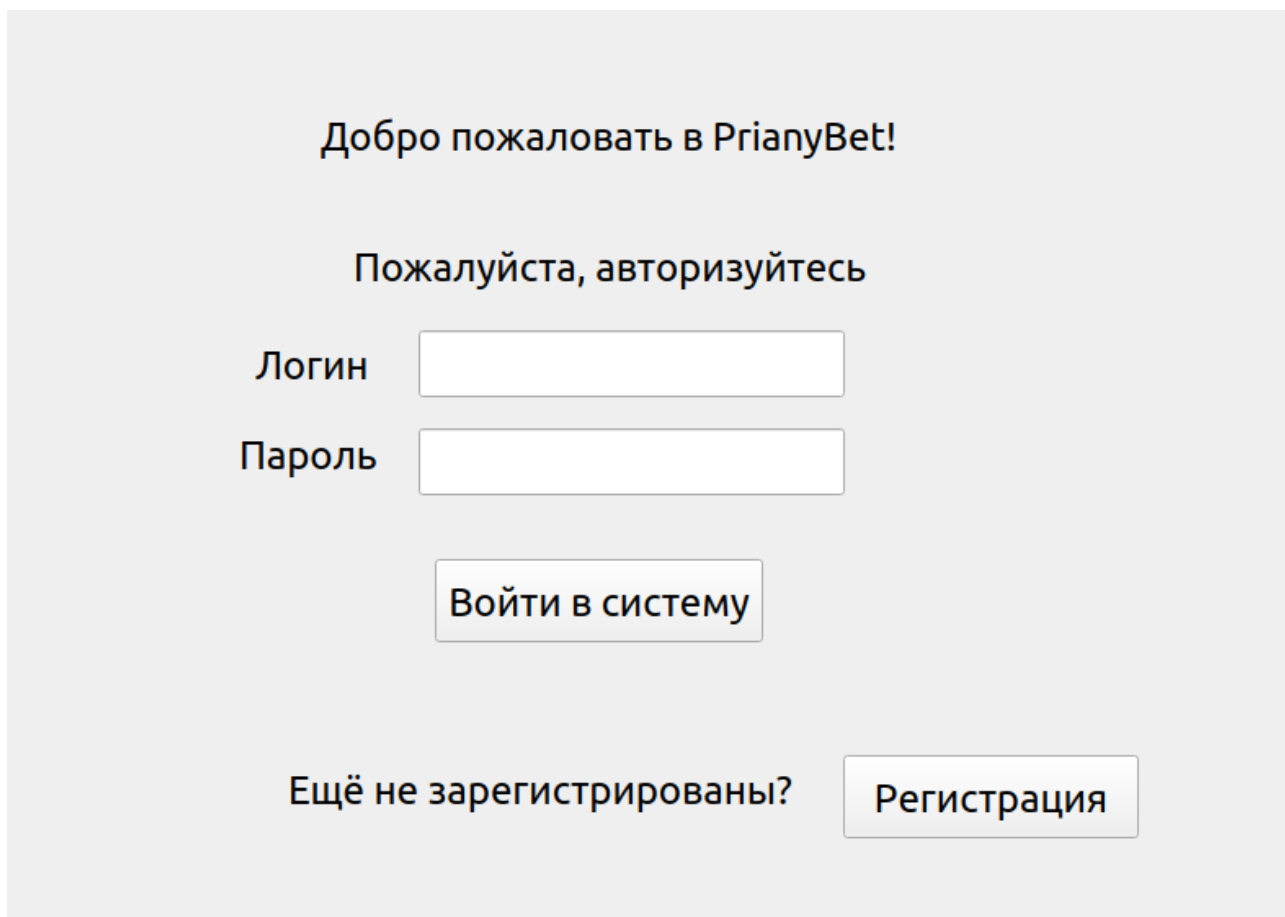
    return self.execute(executeString)
```

3.3 Демонстрация работы программы

В этом разделе будут показана демонстрация работы программы в различных ситуациях.

3.3.1 Вход в систему

Стартовый экран приложения представлен на рисунке 3.1. Пользователь может войти в систему, введя логин и пароль в специальном поле. Также отсюда можно перейти к регистрации



Добро пожаловать в PrianuBet!

Пожалуйста, авторизуйтесь

Логин

Пароль

Ещё не зарегистрированы?

Рисунок 3.1: Стартовый экран приложения

3.3.2 Регистрация

Экран регистрации представлен на рисунке 3.2. Пользователь вводит все данные, которые после ввода проверяются программой. В случае успеха пользователь регистрируется в системе, либо же система выведет сообщение об ошибке.

Введите данные для регистрации:

Фамилия	Имя	Дата рождения
<input type="text" value="Прянишников"/>	<input type="text" value="Александр"/>	<input type="text" value="28.05.2001"/>
Электронная почта	Номер паспорта	Номер телефона
<input type="text" value="mrpriany@mail.ru"/>	<input type="text" value="0415786283"/>	<input type="text" value="88005553535"/>
Логин	<input type="text" value="prianechka"/>	
Пароль	<input type="text" value="kaifarik"/>	
<input type="button" value="Зарегистрироваться"/>		<input type="button" value="Выход"/>

Рисунок 3.2: Экран регистрации

3.3.3 Стартовое меню игрока

Стартовый экран игрока представлен на рисунке 3.3. Игрок может посмотреть сверху свои данные: баланс, логин, а также статус аккаунта. С этого экрана можно перейти к просмотру истории ставок, пополнению баланса или непосредственно к выполнению ставок.

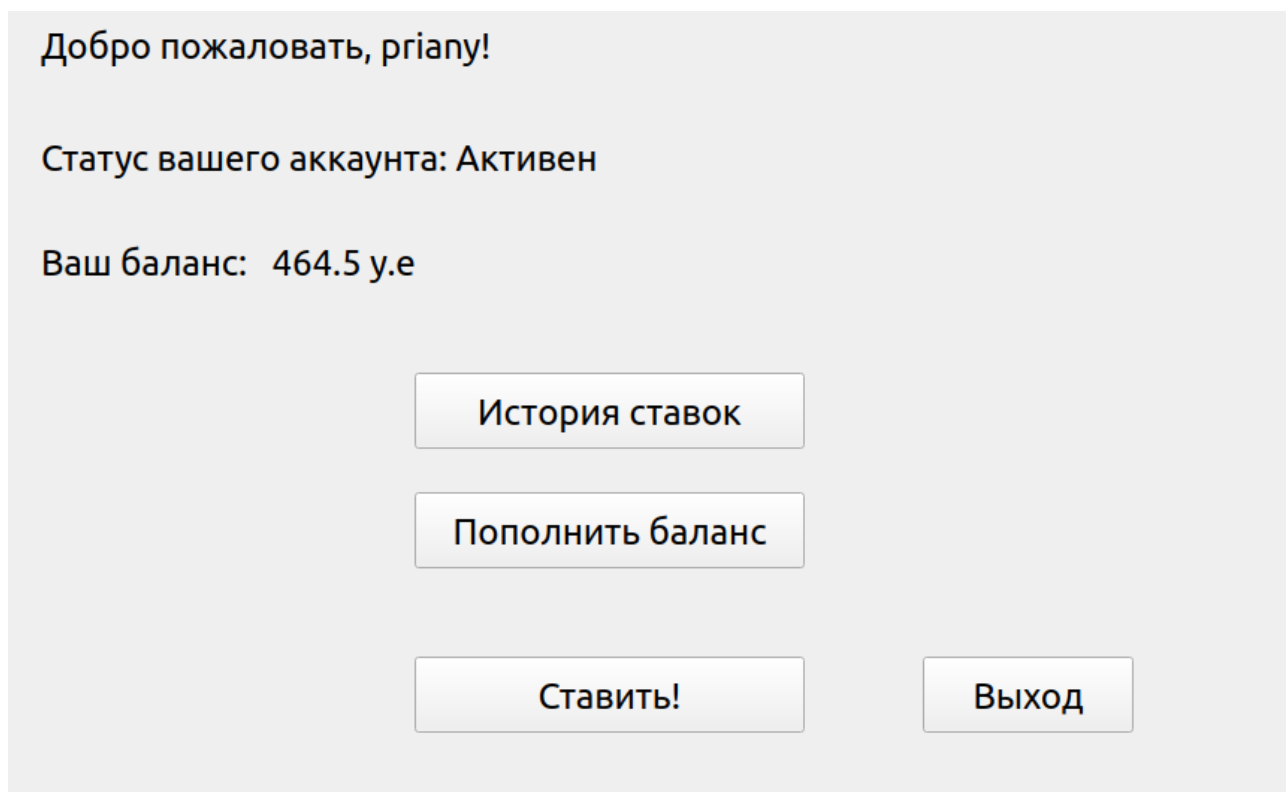


Рисунок 3.3: Стартовый экран игрока

3.3.4 Пополнение баланса

Экран пополнения баланса представлен на рисунке 3.4. Игрок выбирает сумму пополнения, после чего при нажатии кнопки пополнения баланс успешно пополняется.

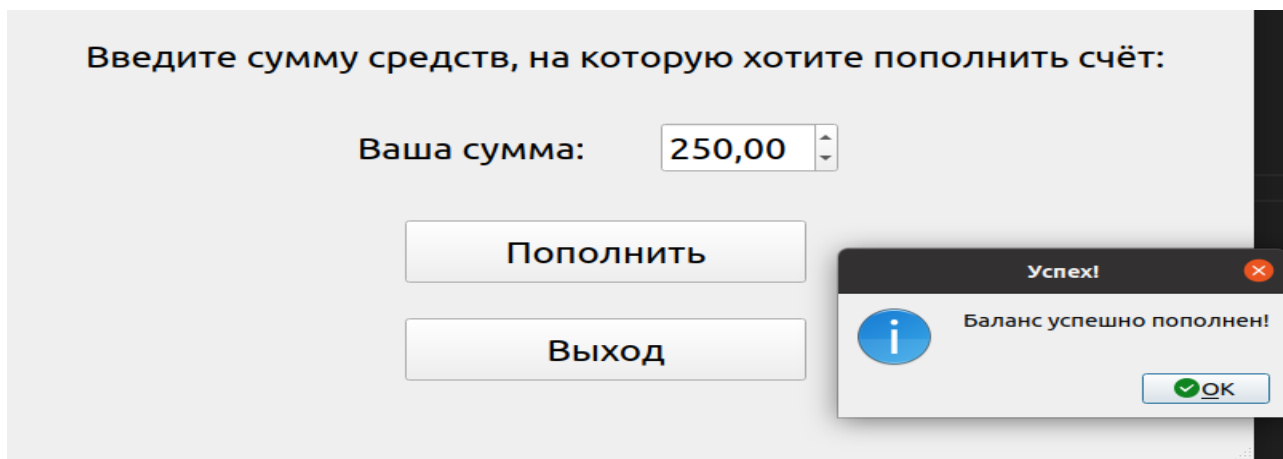


Рисунок 3.4: Экран пополнения баланса

3.3.5 Выполнение ставок

Экран выполнения ставок представлен на рисунке 3.5.

Игрок может увидеть в таблице все события, которые присутствуют в линии в данный момент. Для выбора события, на которое нужно сделать ставку, пользователь должен выделить строку с нужным событием. Слева внизу игрок может выбрать событие для ставки, а также сумму, на которую он хочет заключить пари.

После нажатия кнопки выполнения ставки программа оценит валидность введённых данных, и, в случае успеха, выдаст информационное сообщение.

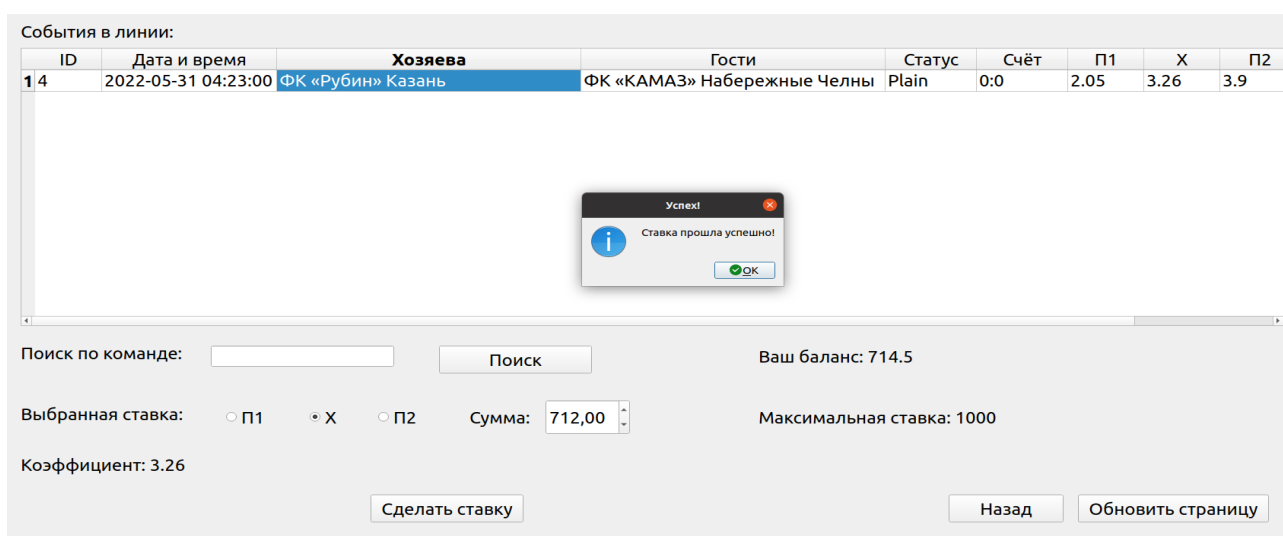


Рисунок 3.5: Экран выполнения ставок

3.3.6 Просмотр истории ставок

Экран просмотра истории ставок представлен на рисунке 3.6.

Игрок может увидеть в таблице все события, которые присутствуют в линии в данный момент. Для выбора события, на которое нужно сделать ставку, пользователь должен выделить строку с нужным событием. Слева внизу игрок может выбрать событие для ставки, а также сумму ставки.

После нажатия кнопки выполнения ставки программа оценит валидность введённых данных, и, в случае успеха, выдаст информационное сообщение.

В таблице представлена история ваших ставок:

ID	Дата ставки	Команда 1	Команда 2	Ставка	Коэффициент	Сумма	Результат	Выплата
1 1	2022-05-31 00:55	РФК «Ахмат» Грозный	ФК «Металлург-Кузбасс»	П1	1.34	25.0	Выиграна	33.5
2 2	2022-05-31 00:55	ФК «Анжи» Махачкала	ФК «Чита» Чита	X	1.8	44.0	Проиграна	0.0
3 3	2022-05-31 01:25	ФК «Рубин» Казань	ФК «КАМАЗ» Набережные Челны	X	3.26	712.0	Принята	0.0

Обновить Назад

Рисунок 3.6: Экран просмотра истории ставок

3.3.7 Добавление матчей в линию аналитиком

Экран добавления матчей в линию представлен на рисунке 3.7.

Экран доступен только в случае входа аналитика в приложение из стартового окна. В таблице представлен список клубов, доступных для добавления в событие. Можно воспользоваться кнопкой поиска для нахождения клубов. Выбор команды осуществляется нажатием на название мышкой.

Также внизу аналитик должен выбрать вероятности наступления событий. Сумма вероятностей должна быть равна 100%.

Выберите команды из списка для добавления матча:

ID	Название	Город
1 1	ПФК ЦСКА Москва	Москва
2 29	ФК «КАМАЗ» Набережные Челны	Набережные Челны
3 31	ФК «СКА-Хабаровск»	Хабаровск

Поиск по команде:

ID первой команды:

ID второй команды:

Вероятность П1:

Вероятность Х:

Вероятность П2:

Дата матча:

Время начала матча:

Рисунок 3.7: Экран добавления матчей в линию

3.3.8 Экран изменения матчей

Экран изменения матчей в линии представлен на рисунке 3.8.

Экран доступен только в случае входа аналитика в приложение из стартового окна. В таблице представлены матчи, добавленные аналитиком в предыдущем окне. Для изменения состояния матча или изменения счёта нужно нажать на строку с событием, а затем нажать на требуемую кнопку внизу таблицы.

Также внизу аналитик может изменить вероятности наступления событий.

События в линии:

ID	Дата и время	Хозяева	Гости	Статус	Счёт	П1	Х	П2
1 4	2022-05-31 04:23:00	ФК «Рубин» Казань	ФК «КАМАЗ» Набережные Челны	Plain	0:0	2.05	3.26	3.9

Изменить состояние матча:

Изменить вероятности:

Рисунок 3.8: Экран изменения матчей в линию

3.3.9 Экран верификации пользователей

Экран верификации пользователей в линии представлен на рисунке 3.9.

Экран доступен только в случае входа администратора в приложение из стартового окна. Аналитик в таблице видит список аккаунтов, которые требуют верификации. В зависимости от введенных данных аналитик может либо верифицировать аккаунт, либо заблокировать.

Поиск по фамилии:

Список аккаунтов, требующих верификации:

ID	ФИО	Паспорт	Номер телефона	Эл.почта
1 2	Александр Прянишников	0415786283	88005553535	mrpriany@mail.ru

Рисунок 3.9: Экран верификации пользователей

4 Экспериментальная часть

В этом разделе будут приведены результаты тестирования алгоритмов.

4.1 Технические характеристики

Технические характеристики устройства, на котором выполнялось тестирование, следующие:

- операционная система: Ubuntu 20.04.1 LTS;
- память: 8 GB;
- процессор: Intel Core i5-1135G7 @ 2.40GHz [?].
- количество ядер процессора: 8

Во время тестирования ноутбук был нагружен только встроенными приложениями окружения, а также непосредственно системой тестирования.

4.2 Зависимость времени ответа базы данных от индексации

Требуется оценить зависимость времени ответа базы данных от индексации и размера таблицы.

Оценка будет производиться для таблицы Games, так как она является самой востребованной – её просматривают все игроки, и её обновление происходит в реальном времени. Измеряться будет время ответа от функции, которая возвращает актуальную линию. Индексироваться будет поле team1ID, связанное с ID одной из команд.

В таблице 4.1 представлены результаты тестов. Время - в миллисекундах.

Таблица 4.1: Результаты тестов

Кол-во записей в БД	Без индексации	С индексацией
10	34	30
30	41	33
50	43	37
100	55	39
200	70	44
400	81	55
600	91	69
800	98	79
1000	112	91

На рисунке представлен график зависимости:

Как видно, индексация позволила уменьшить время ответа. При $N = 600$ время сократилось на 30%, при $N = 800$ – на 20%. При этом индексированная таблица возвращала результат быстрее при всех измерениях.

Заключение

В рамках курсового проекта было создано ПО для обеспечения работы букмекерской конторы.

Была спроектирована предметная база данных. Было написано 10 функций, 11 процедур и два триггера, включающие внутри себя сложные аналитические запросы. Была выделена ролевая модель как на уровне БД, так и на уровне приложения. Была исследована зависимость быстродействия БД от индексации таблицы.

Также в работе были выполнены следующие задачи:

- проведена формализация задачи и определен требуемый функционал;
- проведён анализ СУБД и описана структуру БД, включая объекты, из которых она состоит;
- создана и заполнена БД;
- создан программный продукт для решения задачи, реализованы выбранные алгоритмы;
- реализован понятный интерфейс для клиента.