**Техническое задание по разработке многопользовательской игры «Нарды»**

Варианты использования приложения

**-пользователем Игрок1:**

**User Story 1**. Регистрация

Пользователь хочет иметь возможность регистрироваться в приложении, чтобы получить доступ к игре.

**User Story 2**. Аутентификация

Пользователь хочет иметь возможность аутентифицироваться в приложении, чтобы иметь доступ к игре и просматривать статистику по прошлым играм.

**User Story 3**. Создание игровой сессии

Пользователь хочет иметь возможность получать доступ к новой игровой сессии при входе в систему.

**User Story 4**. Выход из игры

Пользователь хочет иметь возможность выходить из игровой сессии.

**User Story 5**. Просмотр статистики игрока

Пользователь хочет иметь возможность просматривать свою статистику за прошлые игры.

**User Story 6**. Смена языка

Пользователь хочет иметь возможность менять язык пользовательского интерфейса с русского на английский и наоборот.

**User Story 7**. Бросок кубиков для определения очередности ходов

Пользователь хочет иметь возможность бросить кубики, чтобы определить очередность ходов исходя из количества очков.

**User Story 8**. Бросок кубиков для определения хода игрока

Пользователь хочет иметь возможность бросить кубики, чтобы определить свои возможные ходы.

**User Story 9**. Ход игрока

Пользователь хочет иметь возможность совершать ходы исходя из количества выпавших очков.

**User Story 10**. Бросок кубиков соперником

Пользователь хочет иметь возможность видеть количество выпавших очков у соперника.

**User Story 11**. Ход соперника

Пользователь хочет иметь возможность видеть и анализировать ходы противника.

**User Story 12**. Завершение хода соперника

Пользователь хочет иметь возможность знать о завершении хода противника.

**User Story 13**. Поочередное повторение ходов игроков до завершения игры

Пользователь хочет иметь возможность поочередно совершать ходы до окончания игровой сессии.

**User Story 14**. Завершение игры, выявление победителя и проигравшего

Пользователь хочет иметь возможность видеть свой статус(победившего/проигравшего) по окончании игровой сессии

**-пользователем Игрок2:**

**User Story 1**. Регистрация

Пользователь хочет иметь возможность регистрироваться в приложении, чтобы получить доступ к игре.

**User Story 2**. Аутентификация

Пользователь хочет иметь возможность аутентифицироваться в приложении, чтобы иметь доступ к игре и просматривать статистику по прошлым играм.

**User Story 3**. Подключение к игровой сессии

Пользователь хочет иметь возможность подключаться к новой игровой сессии при входе в систему.

**User Story 4**. Выход из игры

Пользователь хочет иметь возможность выходить из игровой сессии.

**User Story 5**. Просмотр статистики игрока

Пользователь хочет иметь возможность просматривать свою статистику за прошлые игры.

**User Story 6**. Смена языка

Пользователь хочет иметь возможность менять язык пользовательского интерфейса с русского на английский и наоборот.

**User Story 7**. Бросок кубиков для определения очередности ходов

Пользователь хочет иметь возможность бросить кубики, чтобы определить очередность ходов исходя из количества очков.

**User Story 8**. Бросок кубиков для определения хода первого игрока

Пользователь хочет иметь возможность бросить кубики, чтобы определить свои возможные ходы.

**User Story 9**. Ход игрока

Пользователь хочет иметь возможность совершать ходы исходя из количества выпавших очков.

**User Story 10**. Бросок кубиков соперником

Пользователь хочет иметь возможность видеть количество выпавших очков у соперника.

**User Story 11**. Ход соперника

Пользователь хочет иметь возможность видеть и анализировать ходы противника.

**User Story 12**. Завершение хода соперника

Пользователь хочет иметь возможность знать о завершении хода противника.

**User Story 13**. Поочередное повторение ходов игроков до завершения игры

Пользователь хочет иметь возможность поочередно совершать ходы до окончания игровой сессии.

**User Story 14**. Завершение игры, выявление победителя и проигравшего

Пользователь хочет иметь возможность видеть свой статус(победившего/проигравшего) по окончании игровой сессии

Функциональные требования

**User Case 1**. Регистрация

1. *UserStory1*: Пользователь хочет иметь возможность регистрироваться в приложении, чтобы получить доступ к игре
2. *Пользователи:* Игрок1, Игрок2
3. *Места в программе:* контекст базы данных, класс пользователя, сервис аккаунта, форма регистрации
4. *Задачи:*

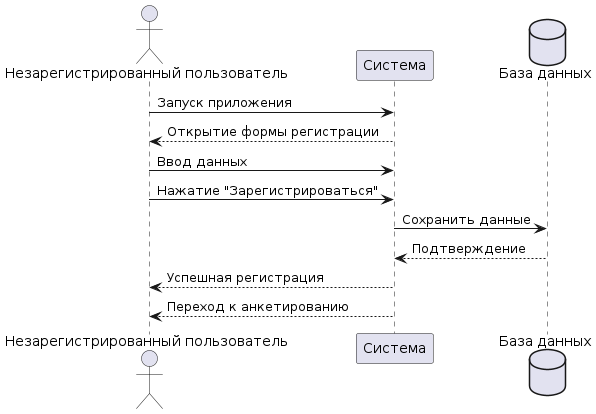
1. Сверстать вкладку регистрации формы входа/регистрации

2. Реализовать сохранение данных пользователя в базе данных

3. Создать сервис аккаунта

4. Реализовать функционал вкладки регистрации

1. *Диаграмма последовательности:*



6) *Техническое задание:*

1. Добавить следующие элементы управления:

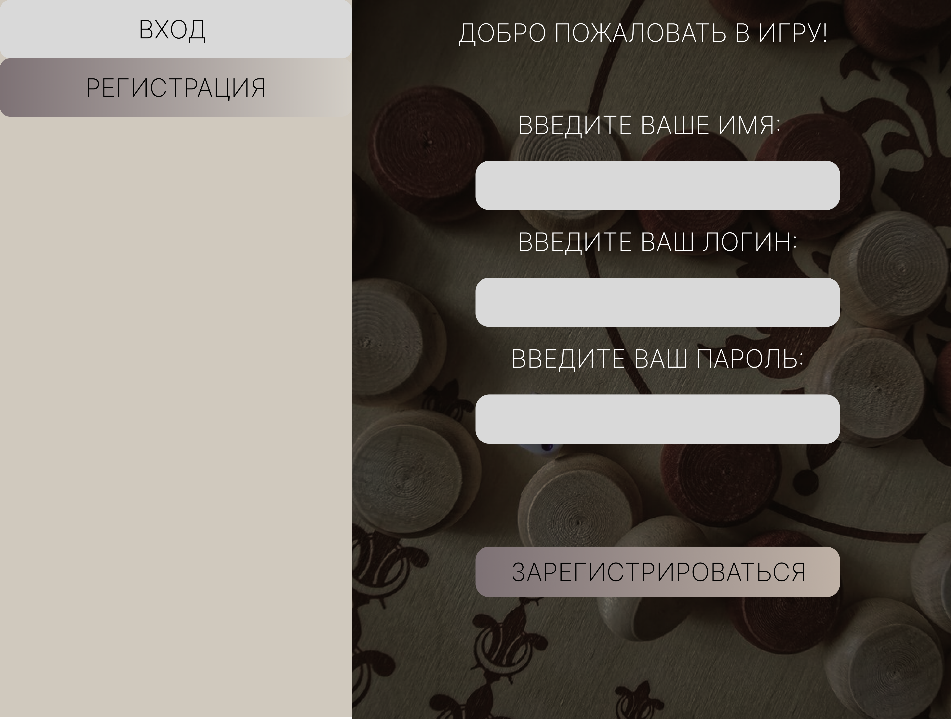
- TextBox-ы для ввода имени, логина и пароля

- Label - ы для наименования полей для ввода

- Label «Добро пожаловать в игру!»

- Button «Зарегистрироваться»

Для каждого TextBox настроить ограничение на количество вводимых символов(до 20)



Вкладка регистрации формы регистрации/входа

На вкладке регистрации 3 поля для ввода имени, логина и пароля, которые нужно заполнить для регистрации. Кнопка «Зарегистрироваться» при корректном вводе значений в полях сохраняет введенные данные в БД и открывает главное окно игровой сессии.

Логин создается в формате электронной почты, а пароль должен иметь минимум 8 символов, заглавную и строчную буквы, цифру и специальный символ. Ограничения на максимальный ввод во всех полях – 20 символов

1. - создать модель(класс) для хранения данных пользователя(с

обязательными свойствами ID, имени, логина, пароля, количества всех игр, побед и поражений пользователя)

- создать контекст базы данных

- создать таблицу в базе данных для хранения данных всех зарегистрировавшихся пользователей

- реализовать хеширование пароля

- создать статическую модель для хранения информации о текущем пользователе

1. - создать метод для регистрации

- создать метод для проверки корректности введенного пароля

- создать метод для проверки корректности введенного логина

- создать метод для проверки отсуствия пустых полей при регистрации

1. Реализовать следующее:

- при нажатии на кнопку «Зарегистрироваться» сохранять введенные данные в базе данных

- при успешном сохранении данных, должен осуществлятся переход на главную форму игры

**User Case 2**. Аутентификация

1. *UserStory2*: Пользователь хочет иметь возможность

аутентифицироваться в приложении, чтобы получить доступ к игре и просматривать статистику по прошлым играм.

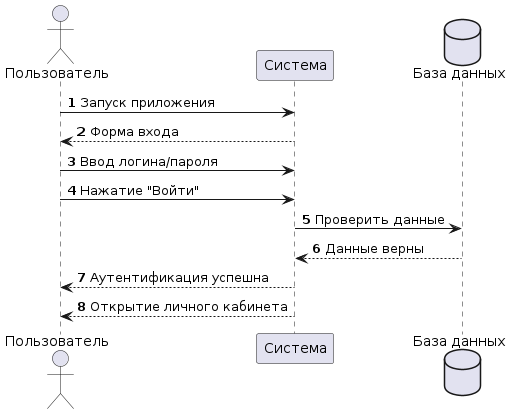
1. *Пользователи*: Игрок1, Игрок2

3) *Места в программе*: сервис аккаунта, форма аутентификации

4) *Задачи*:

1. Сверстать вкладку входа в систему
2. Реализовать проверку данных и вход в систему
3. Настроить переход к игровой форме при успешной аутентификации

5) *Диаграмма последовательности*:



6) *Техническое задание*:

1. Добавить элементы управления на форму входа:

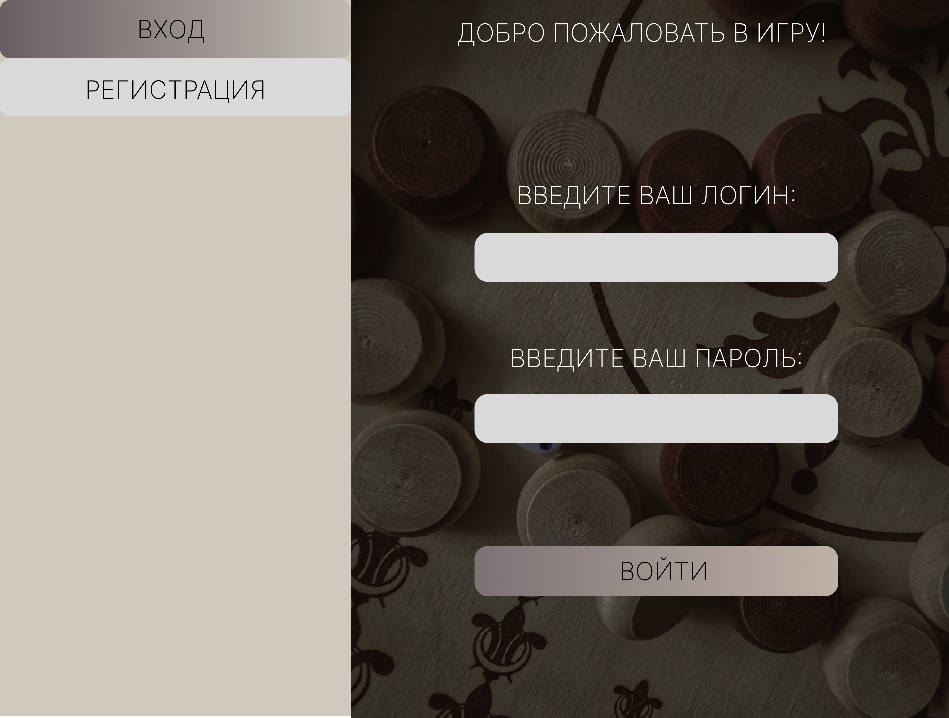
-TextBox для логина

-TextBox для пароля

-Label’ы для подписи к полям

-Label «Добро пожаловать в игру!»

-Button «Войти»  
Ограничение длины ввода в TextBox — до 20 символов



Вкладка аутентификации формы регистрации/входа

На вкладке аутентификации 2 поля для ввода логина и пароля, которые были введены при регистрации. Эта панель нужна, чтобы зарегистрированный пользователь мог зайти в приложение повторно. В поле для ввода пароля добавлено свойство сокрытия пароля. Кнопка «Войти» при корректном вводе значений в полях открывает главное окно игровой сессии

1. В сервисе аккаунта реализовать:

-метод для аутентификации

-метод проверки пустых полей

1. В форме входа/аутентификации:

- При успешной аутентификации (корректный логин/пароль) должен осуществляется переход на главную форму игры

**User Case 3**.**1** Подключение к игровой сессии

1) *User Story*: пользователь хочет получить доступ к новой игровой сессии при входе в систему.

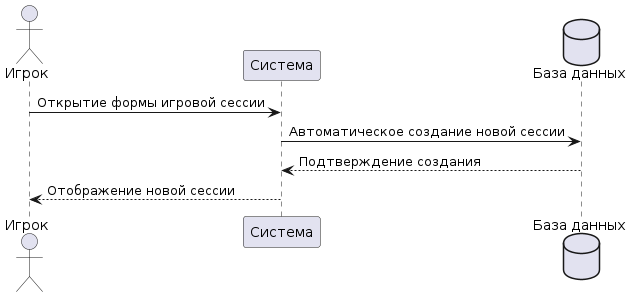
2) *Пользователь*: Игрок1

3) *Места в программе*: контекст базы данных, форма игровой сессии, модель игровой сессии, сервис игровой сессии

4) *Задачи*:

1. Разработать интерфейс формы игровой сессии
2. Создать модель игровой сессии
3. Создать сервис игровой сессии

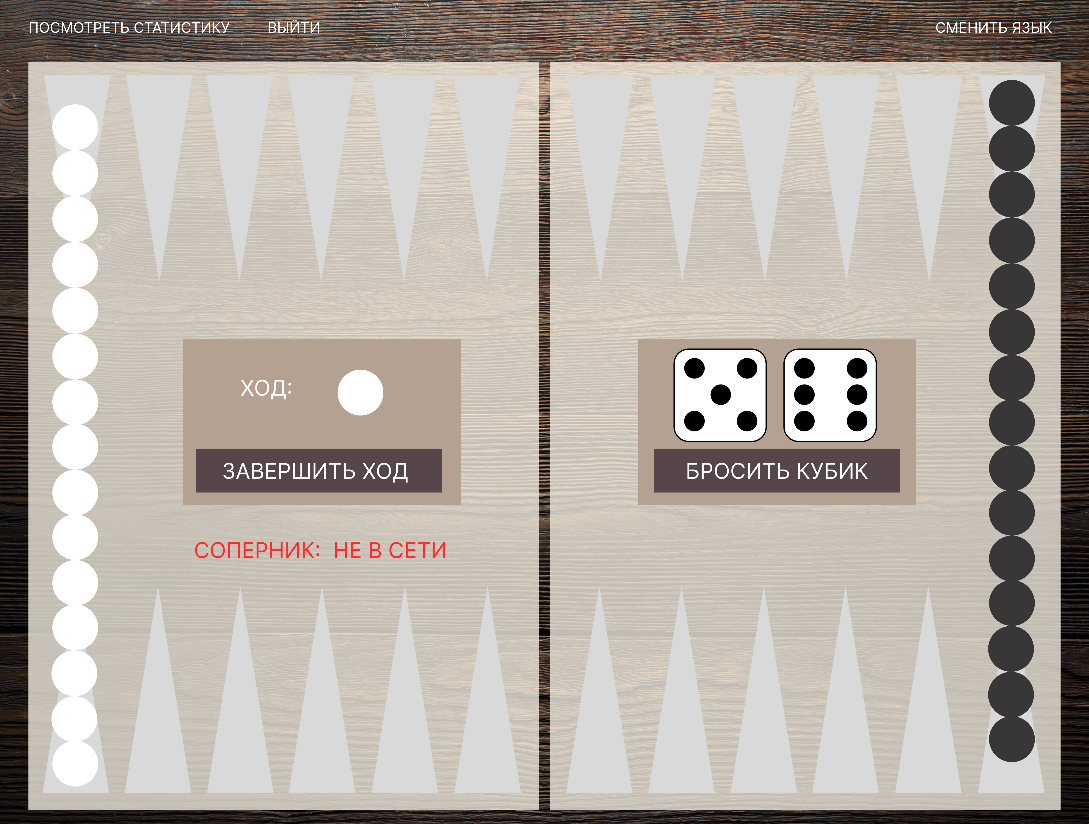
5) *Диаграмма последовательности*



6) *Техническое задание*:

1. Реализовать графический интерфейс:

-Программная отрисовка доски и фишек по макету



Главная форма игровой сессии

Окно игровой сессии – окно, которое открывается после успешного входа/регистрации. Оно представляет собой игровую доску, предназначенную для игры в классические нарды. Окно включает в себя следующие элементы:

- стандартное игровое поле, включающее в себя:  
1) 12 ячеек для ходов с каждой стороны доски

2) По 15 фишек двух цветов (белые и черные, у каждого игрока свой цвет) с каждой стороны доски

3) Панель с обозначением текущего хода в виде цвета фишки и с кнопкой «Завершить ход» для перехода хода сопернику

4) Панель с графическим отображением двух кубиков и с кнопкой «Бросить кубик», при нажатии на которую на кубиках выпадают рандомные цифры от 1го до 6

5) Label для отображения статуса соперника(в сети, не в сети)

- верхнее меню, включающее в себя:

1) Кнопка «Посмотреть статистику» для открытия окна статистики игрока.

2) Кнопка «Выйти» для завершения текущей сессии и возврата на окно входа/регистрации.

3) Кнопка «Сменить язык» для переключенияа языка интерфейса между русским и английским.

Игровая сессия создаётся автоматически при открытии формы, то есть без запроса на создание или подключение. После создания игровой сессии, оба игрока должны выполнить первичный выброс кубиков для определения очередности ходов(у кого сумма очков больше, тот первый будет ходить).

После чего игрок текущего хода повторно бросает кубики для определения своих ходов в соответствии с правилами классических нардов, совершает ход и нажимает на кнопку «Заверишить ход». Ход переходит к другому игроку. Он выполняет те же действия. Ход переходит к другому игроку. Данный цикл повторяется до тех пор, пока кто-то из игроков не выведет свои фишки за пределы доски. После чего объявляются победитель и проигравший, результаты игры сохраняются в базе данных и игровая сессия обновляется

1. Создать:

-Модель для игровой сессии и таблицу в базе данных для нее

1. В сервисе игровой сессии реализовать методы:

-Создания новой игровой сессии

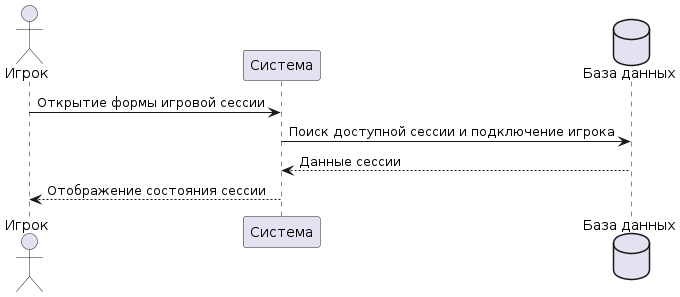
-Подключения второго игрока

**User Case 3.2.** Подключение к игровой сессии

1. *UserStory3.2*: Пользователь хочет иметь возможность подключаться к существующей игровой сессии
2. *Пользователи*: Игрок2
3. *Места в программе*: форма игровой сессии, сервис игровой сессии
4. *Задачи*:

1. Реализовать синхронизацию форм игроков

2. Реализовать отображение статусов игроков

5) *Диаграмма последовательности*:  


1. *Техническое задание*:

1. В сервисе реализовать методы синхронизации двух форм

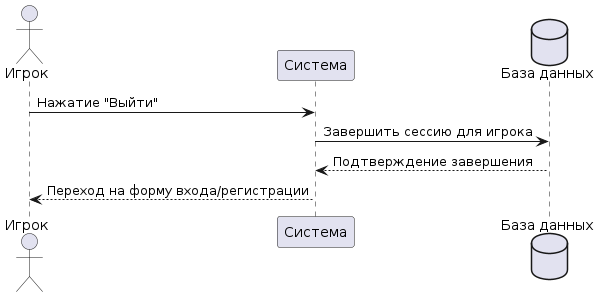
2. В форме игровой сессии добавить Label для отображения статуса соперника

**User Case 4.** Выход из игры

1. *UserStory4*: Пользователь хочет иметь возможность выходить из игровой сессии
2. *Пользователи*: Игрок1, Игрок2
3. *Места в программе*: форма игровой сессии, сервис игровой сессии
4. *Задачи*:

1. Добавить управление выходом из игры

2. Реализовать соответствующие методы сервиса игровой сессии

5) *Диаграмма последовательности*:  


6) *Техническое задание*:

1. В форме игровой сессии добавить LinkLabel для выхода, ведущий на форму входа

2. В сервисе игровой сессии реализовать метод выхода из игровой сессии

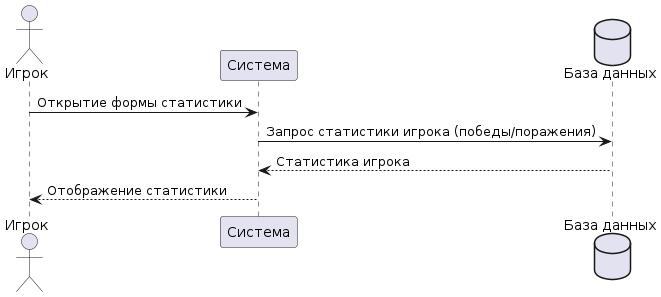
**User Case 5**. Просмотр статистики игрока

1. *UserStory5*: Пользователь хочет иметь возможность просматривать свою статистику за прошлые игры
2. *Пользователи*: Игрок1, Игрок2
3. *Места в программе*: форма игровой сессии, форма статистики игрока, сервис аккаунта
4. *Задачи*:

1. Реализовать получение данных статистики игрока

2. Сделать верстку формы статистики игрока

3. Реализовать открытие формы статистики игрока

5) *Диаграмма последовательности*:  


6) *Техническое задание*:

1. В сервис аккаунта добавить:  
 -метод для получения количества побед и поражений игрока из базы данных

2. -Создать форму статистики игрока

-Сделать верстку формы в соответствии с макетом(добавить соотвествующие Label’ы для отображения информации о количестве всех игр, побед и поражений пользователя)



Форма статистики игрока

Окностатистики игрока — окно, которое открывается при нажатии на кнопку «Посмотреть статистику». Оно извлекает из базы данных информацию о количестве игр, побед и поражений игрока и отображает их под соответствующими заголовками.

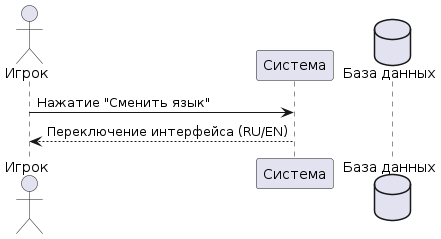
3. В форме игровой сессии добавить LinkLabel, при нажатии на который будет открываться форма статистики текущего пользователя

**User Case 6**. Смена языка

1. *UserStory6*: Пользователь хочет иметь возможность менять язык пользовательского интерфейса с русского на английский и наоборот
2. *Пользователи*: Игрок1, Игрок2
3. *Места в программе*: форма игровой сессии
4. *Задачи*:

1. Реализовать систему локализации

2. Добавить возможность смены языка

5) *Диаграмма последовательности*:  


6) *Техническое задание*:

1. -настроить свойства локализации текстовых элементов интерфейса

-создать файл ресурсов

-установить значения свойств текстовых элементов

2. В форму игровой сессии добавить LinkLabel для осуществления смены языка, при нажатии на него все текстовые элементы интерфейса приложения должны поменяться с русского на английский и наоборот

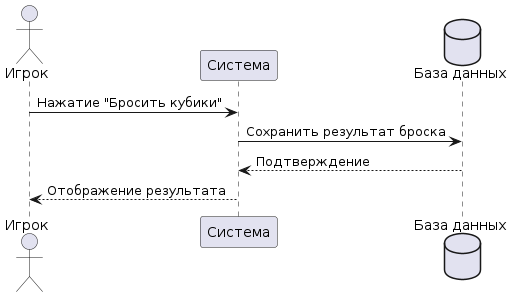
**User Case 7**. Бросок кубиков для определения очередности ходов

1. *UserStory7*: Пользователь хочет иметь возможность бросить кубики, чтобы определить очередность ходов исходя из количества очков
2. *Пользователи*: Игрок1, Игрок2
3. *Места в программе*: форма игровой сессии, сервис игровой сессии
4. *Задачи*:

1. Реализовать графическое отображение брошенных кубиков на форме игровой сессии

2. Реализовать методы для взаимодействия с кубиками

3. Реализовать функционал формы для взаимодействия с кубиками

5) *Диаграмма последовательности*:  


6) *Техническое задание*:

1. На форме игровой сессии реализовать программную отрисовку грани кубика

2. В форме игровой сессии добавить:

- метод для осуществления броска кубиков

- метод для сравнения значений, выпавших на кубиках соперников

3. В форму игровой сессии добавить:

- Panel(с Button «Бросить кубики» внутри) в соответствии с макетом для осуществления бросков кубиков и отображения выпавших кубиков игрока текущего хода

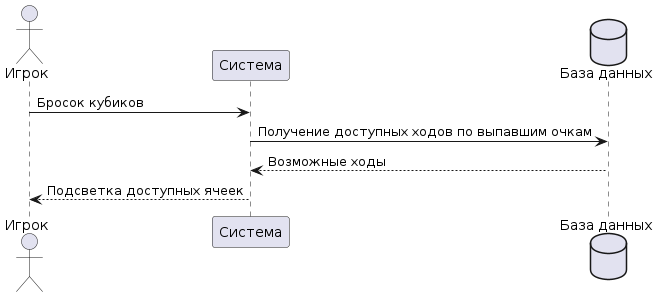
- Panel(с Label внутри) в соответствии с макетом для отображения текущего хода исходя из количества выпавших очков игроков

**User Case 8**. Бросок кубиков для определения хода игрока

1. *UserStory8*: Пользователь хочет иметь возможность менять язык пользовательского интерфейса с русского на английский и наоборот
2. *Пользователи*: Игрок1, Игрок2
3. *Места в программе*: форма игровой сессии, сервис игровой сессии
4. *Задачи*:

1. Реализовать определение всевозможных ходов игрока

2. Реализовать выделение ячеек, доступных для текущего хода

5) *Диаграмма последовательности*:  


6) *Техническое задание*:

1. В сервис игровой сессии добавить метод для определения всевозможных ходов игрока исходя из количества выпавших очков на кубиках

2. В форме реализовать выделение ячеек, на которые может ходить игрок

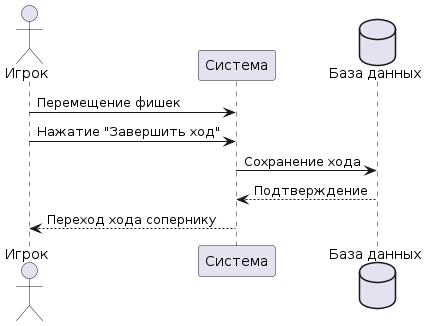
**User Case 9**. Ход игрока

1. *UserStory9*: Пользователь хочет иметь возможность совершать ходы исходя из количества выпавших очков
2. *Пользователи*: Игрок1, Игрок2
3. *Места в программе*: форма игровой сессии, сервис игровой сессии
4. *Задачи*:

1. Реализовать перемещение фишек по доступным ячейкам

2. Реализовать сохранение ходов игрока в базе данных

3. Реализовать переход хода сопернику

5) *Диаграмма последовательности*:  
 

6) *Техническое задание*:

1. В форме игровой сессии реализовать перемещение фишек по доступным для игрока ячейкам исходя из выпавших кубиков

2. Реализовать сохранение хода игрока в базе данных после нажатия на Button «Завершить ход»(добавить Button на Panel с отображением текущего хода), в сервсис игровой сессии добавить:

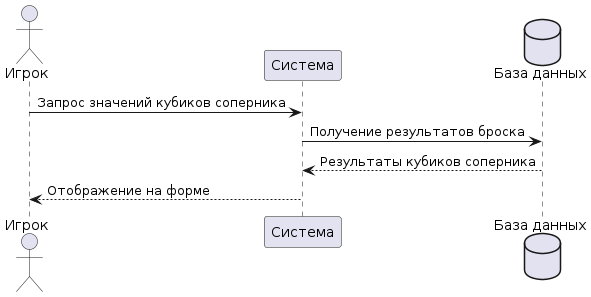
-метод для сохранения хода игрока в базе данных

3. Реализовать изменение Label’a на панели для отображения текущего хода после перехода хода сопернику

**User Case 10**. Бросок кубиков соперником

1. *UserStory10*: Пользователь хочет иметь возможность видеть количество выпавших очков у соперника
2. *Пользователи*: Игрок1, Игрок2
3. *Места в программе*: форма игровой сессии, сервис игровой сессии
4. *Задачи*:

1. Реализовать отображение у игрока выпавших кубиков соперника

5) *Диаграмма последовательности*:  
 

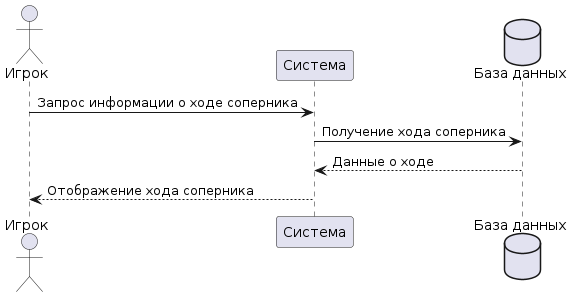
6) *Техническое задание*:

1. В сервисе игровой сессии реализовать метод для отображения у игрока выпавших кубиков соперника

**User Case 11**. Ход соперника

1. *UserStory11*: Пользователь хочет иметь возможность видеть и анализировать ходы противника
2. *Пользователи*: Игрок1, Игрок2
3. *Места в программе*: форма игровой сессии, сервис игровой сессии
4. *Задачи*:

1. Реализовать отображение у игрока хода соперника

5) *Диаграмма последовательности*:  
 

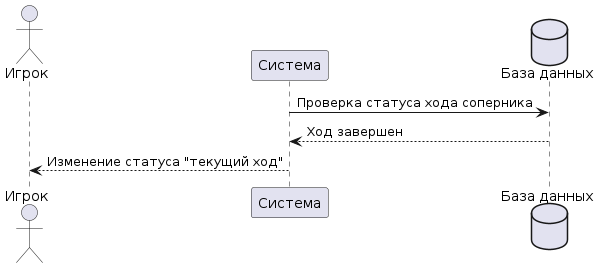
6) *Техническое задание*:

1. В сервисе игровой сессии реализовать метод для отображения у игрока хода противника

**User Case 12**. Завершение хода соперника

1. *UserStory12*: Пользователь хочет иметь возможность знать о завершении хода противника
2. *Пользователи*: Игрок1, Игрок2
3. *Места в программе*: форма игровой сессии, сервис игровой сессии
4. *Задачи*:

1. Реализовать переход хода от соперника

5) *Диаграмма последовательности*:  
 ****

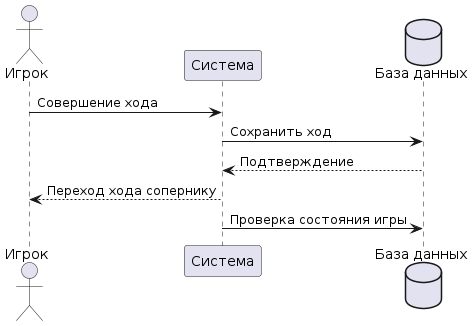
6) *Техническое задание*:

1. При завершении хода соперником должен изменятся Label на панели для отображения текущего хода

**User Case 13**. Поочередное повторение ходов игроков до завершения игры

1. *UserStory13*: Пользователь хочет иметь возможность поочередно совершать ходы до окончания игровой сессии
2. *Пользователи*: Игрок1, Игрок2
3. *Места в программе*: форма игровой сессии, сервис игровой сессии
4. *Задачи*:

1. Реализовать цикл смены ходов

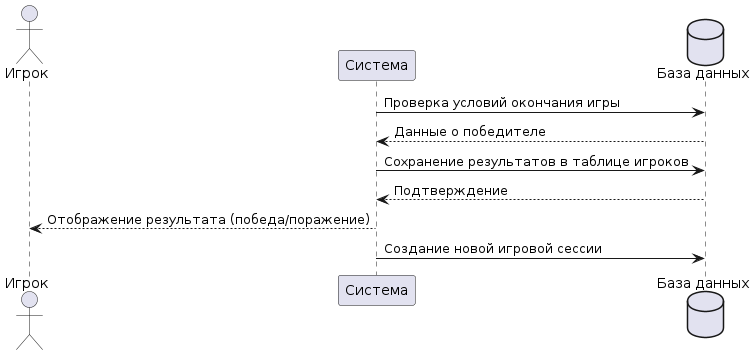
5) *Диаграмма последовательности*:  
 

6) *Техническое задание*:

1. Должен осуществляться поочередный цикл ходов игроков до окончания игровой сессии

**User Case 14**. Завершение игры, выявление победителя и проигравшего

1. *UserStory14*: Пользователь хочет иметь возможность видеть свой статус(победившего/проигравшего) по окончании игровой сессии
2. *Пользователи*: Игрок1, Игрок2
3. *Места в программе*: форма игровой сессии, сервис игровой сессии, сервис аккаунта
4. *Задачи*:
   1. Реализовать выявление победителя и проигравшего
   2. Реализовать вывод MessageBox’ов с соответсвующими статусами
   3. Реализовать сохранение в базе данных итогов игры

5) *Диаграмма последовательности*:  
 

6) *Техническое задание*:

1. В сервис игровой сессии добавить метод для определения победителя на основании того, кто первый вывел фишки за пределы доски, применять его после каждого хода игроков

2. По окончании игры выводить MessageBox с соответствующим статусом игрока(победитель/проигравший), при нажатии на кнопку «ОК» MessageBox’а должна создаваться новая игровая сессия

3. В сервис аккаунта добавить метод для сохранения данных об итоге игры в таблице пользователей в базе данных