ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

***«*САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»**

**ИНСТИТУТ КОМПЬЮТЕРНЫХ НАУК И ТЕХНОЛОГИЙ**

**ВЫСШАЯ ШКОЛА ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ И СУПЕРКОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ**

**Отчет о прохождении   
стационарной ознакомительной, учебной практики на тему:**

«Разработка модулей веб-приложения «Домино»

Небылицына Павла Алексеевича, гр. 3530903/00002

**Направление подготовки:** 09.03.03 Прикладная информатика

|  |
| --- |
| **Место прохождения практики:** СПбПУ, ИКНТ, ВШИСиСТ |

*(указывается наименование профильной организации или наименование структурного подразделения*

|  |
| --- |
|  |

*ФГАОУ ВО «СПбПУ», фактический адрес)*

**Сроки практики:** с 15.06.2022 по 10.07.2022.

**Руководитель практики от ФГАОУ ВО «СПбПУ»:**

|  |
| --- |
| Щукин Александр Валентинович, к.т.н., доцент |

*(Ф.И.О., уч. степень, должность)*

|  |
| --- |
| **Консультант практики от профильной организации:**  Щукин Александр Валентинович, к. т. н., доцент ВШИСиСТ |

*(Ф.И.О., должность)*

|  |
| --- |
|  |

**Оценка:**

Руководитель практики

от ФГАОУ ВО «СПбПУ» Щукин А.В.

Консультант практики

от ФГАОУ ВО «СПбПУ»

Обучающийся Небылицын П.А.

Дата:

# Введение

Целью данной практики является приобретение первичных профессиональных умений и навыков в соответствии с направлением подготовки. Для этого была выбрана задача расширения функционала веб-приложения «Домино», а именно: реализация настройки параметров турнира, визуализации конкретной игры и турнирной таблицы, сохранения и восстановления состояния турнира, проверки модулей игроков при их загрузке, доработка модуля MTable с целью улучшения качества отлова жульничества, а также визуальное оформление веб-приложения согласно требованиям.

Актуальность самого web-приложения заключается в автоматизации проведения кибертурнира по домино среди студентов 1 курса с целью мотивации последних в обучении программированию. Данный продукт совмещает в себе соревновательную и обучающую составляющую, что делает его эффективным инструментом в достижении поставленной цели.

# 1. Теоретическая часть

## **1.1 Средства разработки**

Для разработки используется среда ASP.NET Core, которая является кроссплатформенной, высокопроизводительной средой с открытым исходным кодом для создания современных облачных приложений, подключенных к Интернету. Помимо этого, данная среда разработки позволяет создавать веб-приложения и использовать их на Windows, а также имеет обширный выбор хостинг-провайдеров, поддерживающих данную среду разработки. [1]

Кроме того, ASP.NET предоставляет возможность использования Entity Framework — инфраструктуры объектно-реляционного сопоставления (ORM) для обспечения разработчикам автоматизированного механизма доступа и хранения данных в базе данных. EF Core API создает базу данных и таблицы с помощью миграции на основе соглашений и конфигурации, предусмотренных в предметных классах. Этот подход полезен в проектировании, управляемом предметной областью. EF Core, как и EF 6, поддерживает следующие функции и концепции: [2]

1. DbContext и DbSet
2. Модель данных
3. Запросы с использованием Linq-to-Entities
4. Отслеживание изменений
5. Сохранить изменения
6. Миграции

Также в данной работе используется паттерн MVC (рисунок 1), предназначенный для создания динамических веб-сайтов с четким разделением задач. Он обеспечивает полный контроль разметки и использует новейшие веб-стандарты. [3] [4]

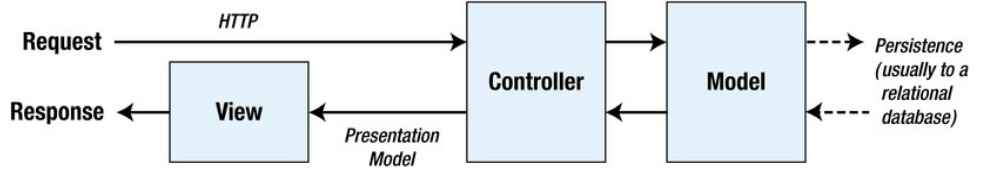


Рисунок 1 – Схема паттерна Model-View-Controller (MVC)

## **1.2 Правила турнира**

В целом основание между двумя игроками проводится по классическим правилам домино [5], однако ввиду программной реализации имеются свои особенности в проведении самого турнира:

1. Каждый игрок предоставляет реализованный им программный модуль, который отвечает за действия игрока.
2. Каждое соревнование между двумя игроками проводится установленное в параметрах количество раз за одну сторону (то есть сначала один модель ходит первым несколько раз, затем такое же количество раз ходит первым второй модуль).
3. Турнир состоит из отборочного этапа и плей-оффа. В отборочном этапе (который проходит автоматически) каждый игрок соревнуется с шаблонным модулем, после чего лучшие игроки отбираются на плей-офф. Каждый шаг в плей-оффе проводится с помощью администратора.
4. Плей-офф состоит из раундов, где в результате каждого дальше выходит лишь половина игроков. Кроме того, проводится отдельное соревнование за третье место из участников, проигравших в предпоследнем раунде, если таковой есть.

## **1.3 Используемые сторонние библиотеки**

Для турнирной таблицы был использован LeaderLine [7] - гибкий инструмент для реализации линий и стрелок с помощью js (рисунок 2). Данный инструмент использовался в визуализации связей таблицы плей-оффа. Он был подключен с помощью среды генерации кода Scaffolding. [6]

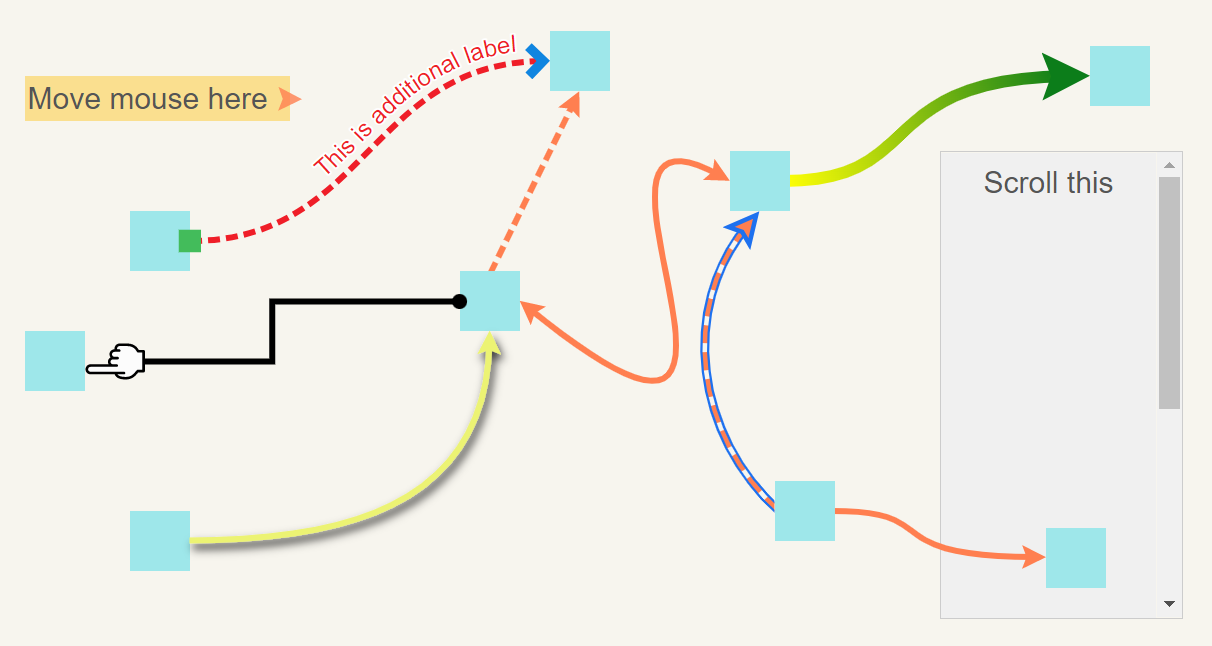


Рисунок 2 – Пример предлагаемых LeaderLine реализаций линий между блоками

Для восстановления/сохранения турнира был использован JsonSerializer, предоставляющий инструменты для преобразования объектов .NET в JSON формат и обратно. [8]

# 2. Практическая реализация

## **2.1. Настройка параметров турнира**

Как было указано в теоретической части, для проведения турнира необходимо, чтобы администратор мог задать количество игр за одну сторону. Для этого была добавлена отдельная страница с данным функционалом (рисунок 3) и ссылка на неё (рисунок 4).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3 – Представление для настройки параметров турнира

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 4 – Представление, содержащее ссылку на настройки турнира

Кроме того, добавлена валидация введённого значения – оно должно быть целым натуральным числом (рисунок 5).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 5 – Валидация количества игр

В данном случае был задействован паттерн MVC, так как в случае POST запроса данные со страницы отправляются в метод действия Settings контроллера Game для установки значения статической переменной, отвечающей за количество игр (рисунок 6). Также была учтена необходимость безопасности и поставлено ограничение по роли на данный функционал – только для администратора.

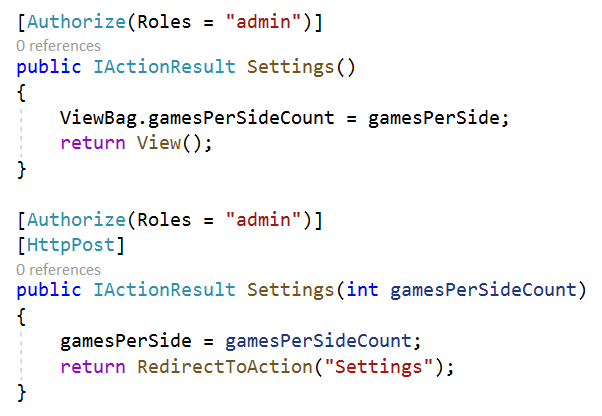


Рисунок 6 – Метод действия Settings контроллера Game

В итоге получен модуль Settings.cshtml и реализованы два метода действий Settings контроллера Game.

## **2.2. Визуализация выбранной игры по домино**

Для большей наглядности, а также дополнительной проверки проведения турнира необходима возможность демонстрации выбранных администратором игр. Визуализация конкретной игры доступна из таблицы результатов соревнования одной пары в плей-оффе (рисунок 7, элемент «Визуализация»).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 7 – Таблица результатов соревнования одной пары в плей-оффе

В появившемся окне доступны действия для начала визуализации, а также установки паузы/продолжения (рисунок 8).

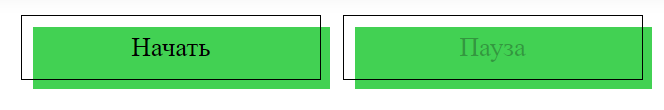


Рисунок 8 – Кнопки начала, паузы/продолжения визуализации

Сама визуализация заключается в пошаговой демонстрации процесса игры с соответствующей анимацией и пояснительными записями (рисунки 9–11).

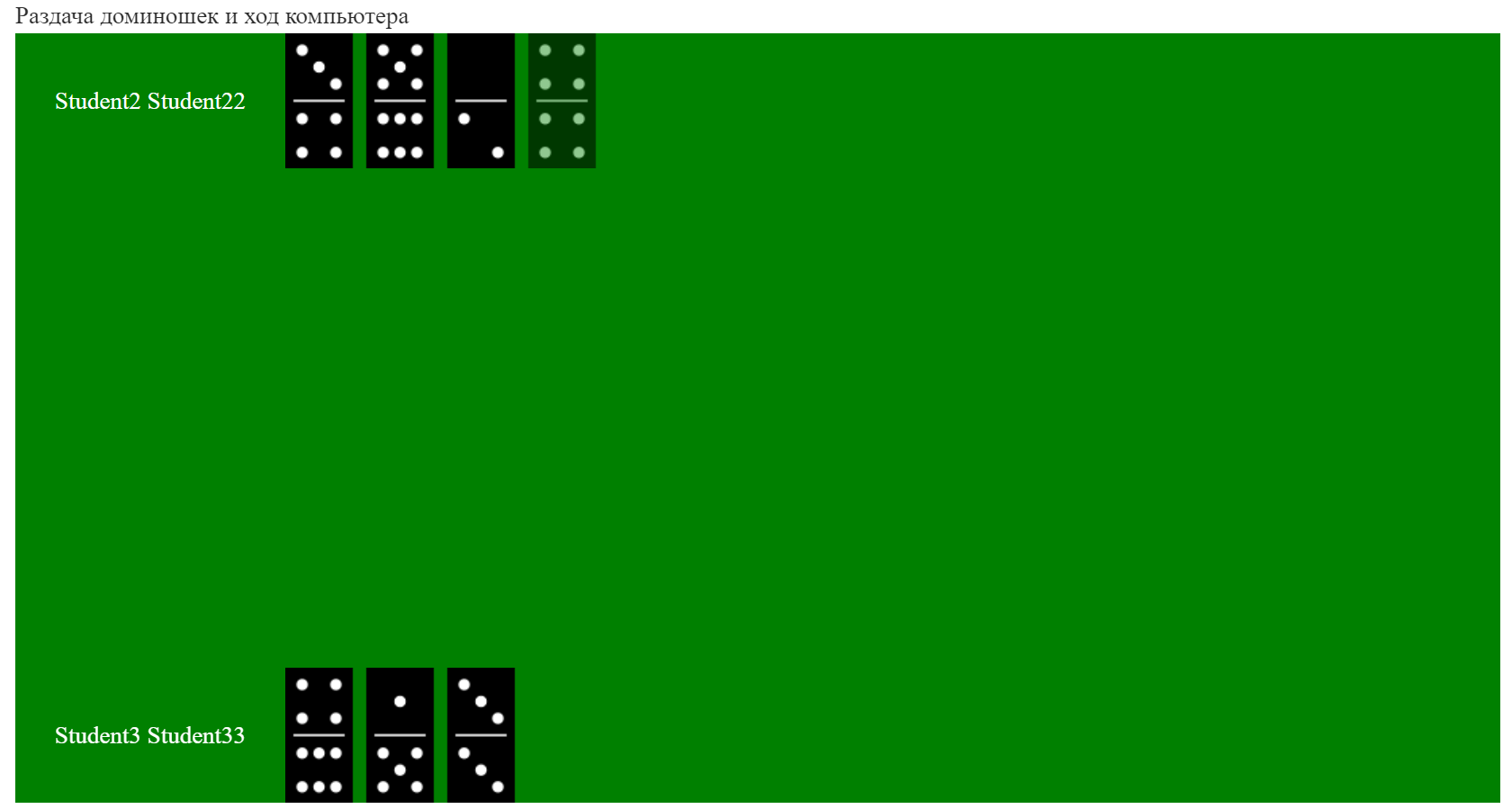


Рисунок 9 – Этап раздачи доминошек в игре

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 10 – Основной этап игры: выкладка домино на поле

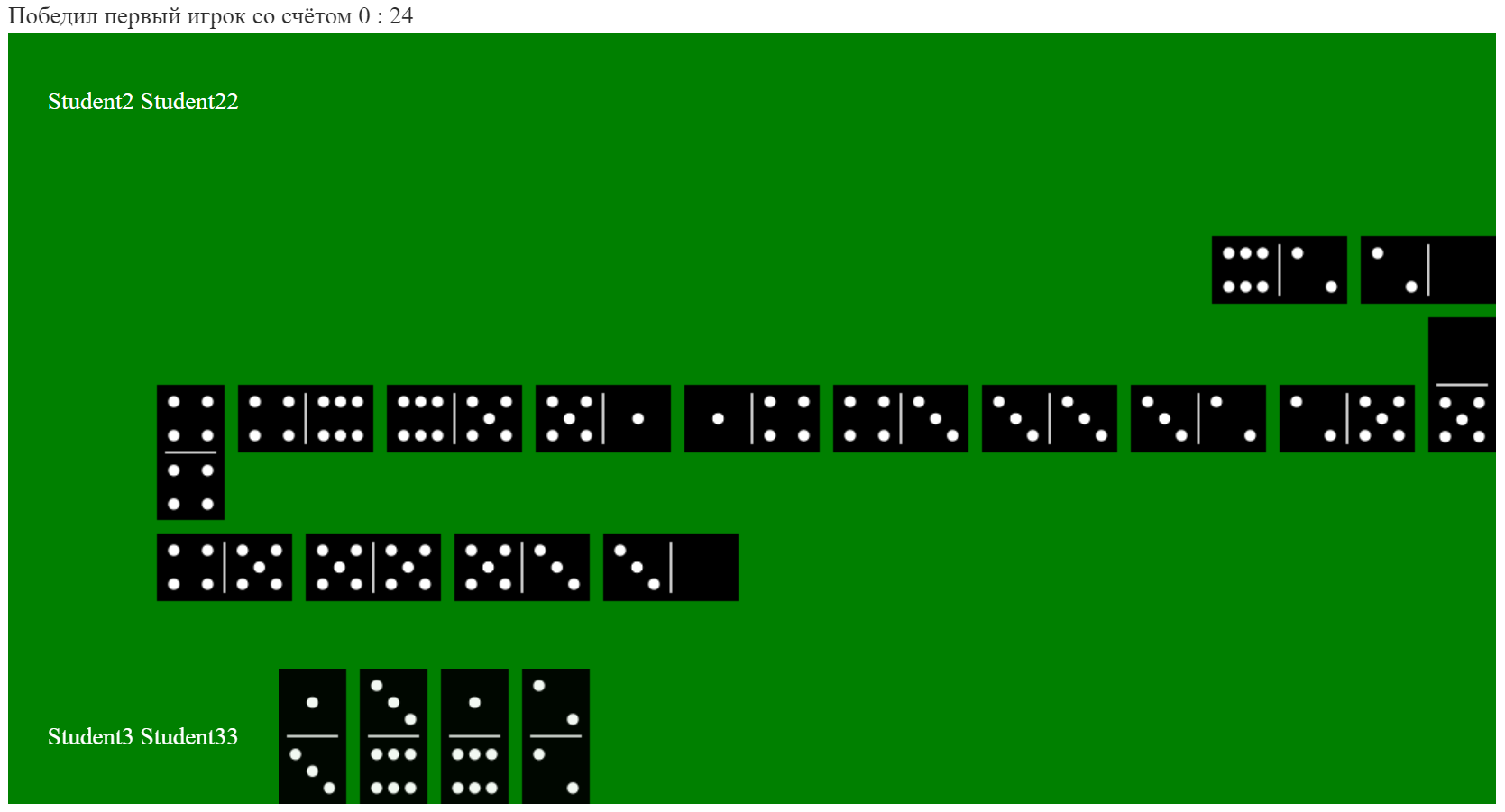


Рисунок 11 – Окончание игры, вывод результата

Для реализации данного функционала был написан метод действия Visualise контроллера Game, принимающий на вход путь к файлу с процессом игры, и возвращающий представление Visualise.cshtml с обработанными данными из файла. Далее представление использует модуль domino.js передавая ему список шагов, имена игроков и результат игры, и уже js файл отвечает за прорисовку/перерисовку каждого домино в зависимости от размеров экрана, анимацию появления домино и обеспечения возможности паузы. На рисунке 12 представлена часть данного модуля, отвечающая за анимацию появления доминошки.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 12 – Метод, отвечающий за анимацию появления доминошки

В итоге получены следующие модули: domino.js, Visualise.cshtml и метод действия Visualise контроллера Game.

## **2.3. Визуализация турнирной таблицы**

Также важно наглядное представление плей-оффа, так как простой табличный вариант не удобен в данном случае из-за структуры данных.

Для оформления блоков был использован css код, находящийся в модуле Playoff.cshtml. Всем блокам был задан id, соответствующий шаблону «N M type», где N – номер раунда, M – текущий номер в раунде и type – значение, показывающее является блок верхним или нижним. Благодаря такому шаблону и использованию js были реализованы подходящие отступы для каждого блока вне зависимости от общего количества участников плей-оффа (рисунок 13).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 13 – Выравнивание блоков турнира с помощью js

Кроме того, необходимо было учесть, что количество игроков в турнире может быть от 2 до любой степени двойки (как правило 16 и не более 64). Следовательно, в конце отборочного этапа необходим выбор количества участников плей-оффа (рисунок 14). Зная заданное число участников, можно правильно расставить отступы и «провести» линии, показывающие блок в следующем раунде для победителя конкретного соревнования (рисунок 15).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 14 – Выбор количества участников плей-оффа



Рисунок 15 – Пример состояния таблицы плей-оффа

Ввиду того, что положение блоков непостоянно линии также должны быть динамичны (рисунок 16). Для этого был использован LeaderLine [?], который подключается с помощью js файла. С помощью другого js файла необходимо было выбрать HTML элементы, которые необходимо соединить – это было реализовано в js функции, которая вызывается при загрузке страницы (рисунок 17).

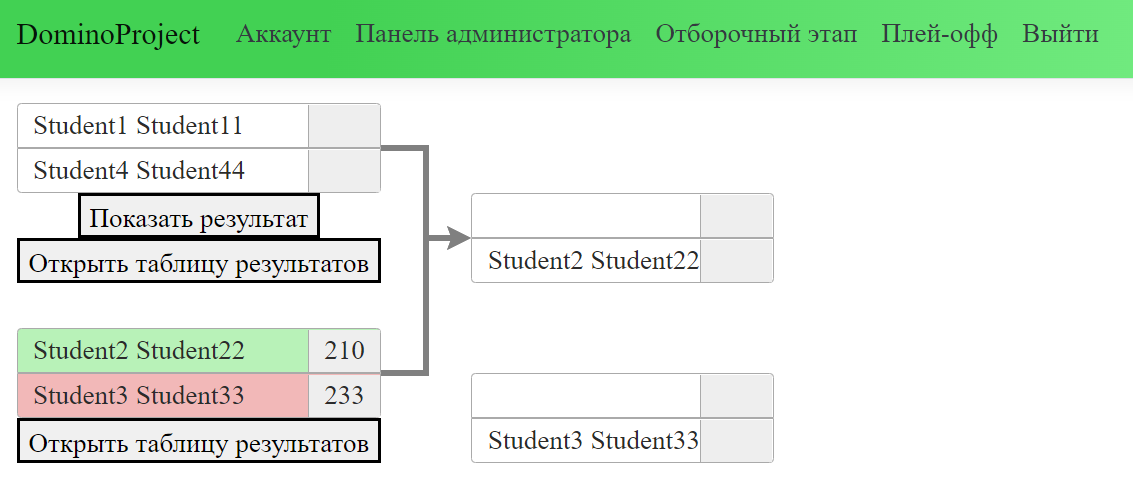


Рисунок 16 - Пример состояния таблицы плей-оффа с изменённым положением линий

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 17 – Часть кода playoff.js, отвечающая за расстановку линий

В итоге получены следующие модули: playoff.js, Playoff.cshtml с учётом уже реализованной функциональности контроллера Game (методы действий Playoff, PlayMatchup, ShowResult, MatchupInfo).

## **2.4. Сохранение/восстановление турнира**

Проведённый турнир содержит достаточно большой список результатов и может быть проведён не за один раз, кроме того, возможно возникновение непредвиденных ошибок из-за качества модулей игроков, поэтому необходимо обеспечить возможность создания «точек сохранения» и в целом возможность восстановить предыдущие результаты.

Для реализации данного функционала была использовала сериализация с помощью JsonConvert [8] в файл, а также возможность десериализации. Были добавлены методы действия контроллера SaveTournamentData и LoadTournamentData, а также вспомогательный метод модуля Helper.cs.

Изображение выглядит как текст

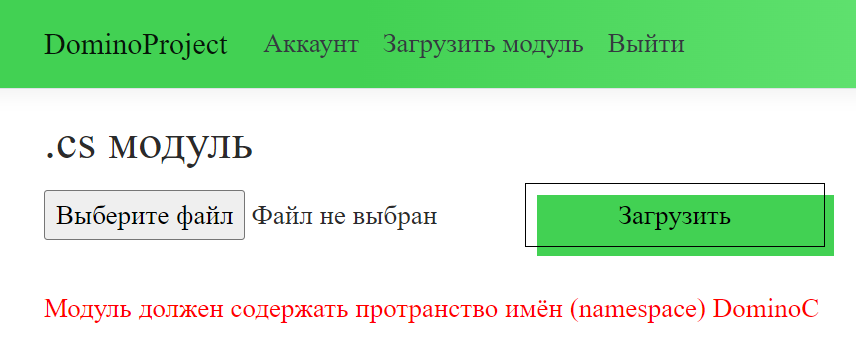
Автоматически созданное описание

Рисунок 18 – Элементы доступа к сохранению/восстановлению турнира

## **2.5. Валидация модулей игроков**

Ввиду того, что игроки, участвуя, разрабатывают не всю систему, а лишь модуль, к последнему есть определённые требования по структуре, поэтому необходима проверка загружаемых модулей на наличие основных элементов. Для этого в котроллере Home был введён метод CheckFile и другие вспомогательные методы с целью предоставления студенту информации о корректности загруженного им файла.

Во-первых, файл должен содержать пространство имён DominoC (рисунок 19), которое в свою очередь должно содержать класс MFPlayer (рисунок 20).

Изображение выглядит как текст, человек, снимок экрана

Автоматически созданное описание

Рисунок 19 – Сообщение об ошибке пространства имён

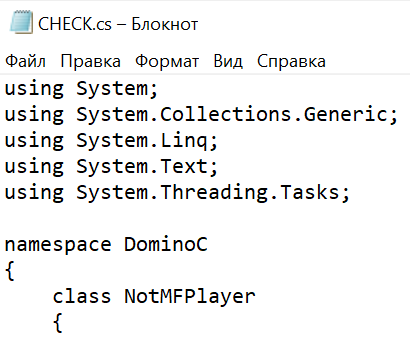
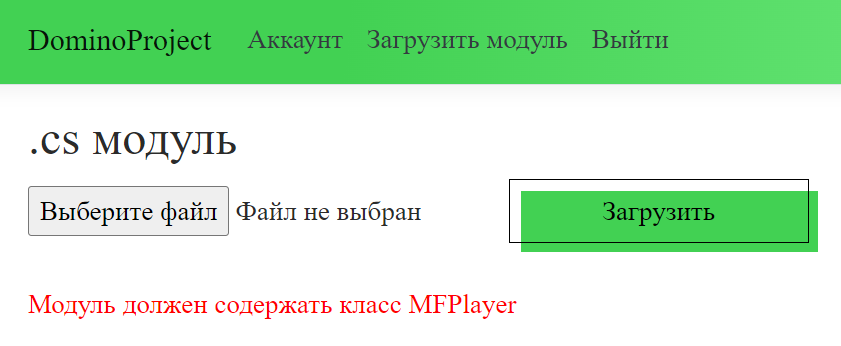


Рисунок 20 – Сообщение об ошибке имени класса

Во-вторых, была сделана проверка на наличие необходимых для главного модуля методов (рисунок 21).

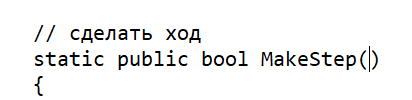
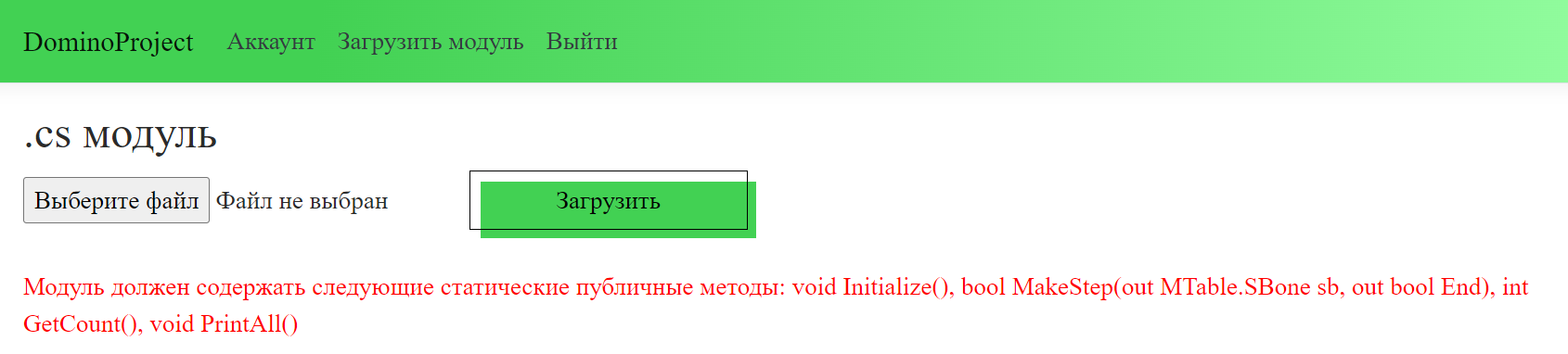
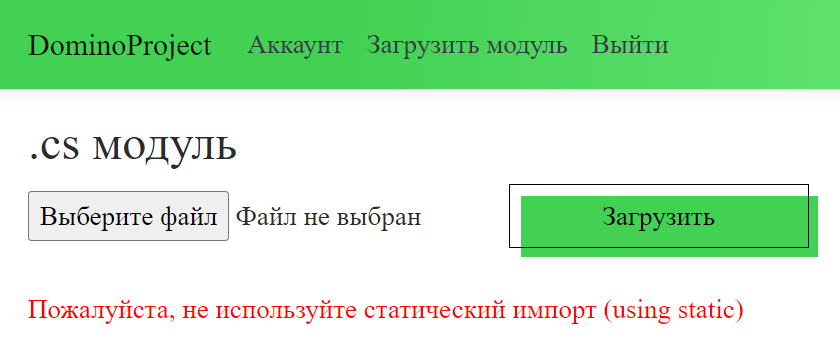


Рисунок 21 – Сообщение об ошибке наличия методов или сигнатуры

Кроме того, была добавлена проверка, связанная с особенностями автоматической компиляции сборки (рисунок 22).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 22 – Сообщение о необходимости внесения изменений в код

В итоге были доработаны следующие модули: HomeController.cs, File.cshtml.

## **2.6. Добавление проверок на жульничество в модуле MTable**

Помимо непосредственной работы с веб-приложением была проведена доработка главного модуля, участвующего в сборке, необходимой для проведения соревнования между двумя игроками. Ранее уже была реализован отлов попыток игрока пристроить костяшку, которая не подходит в данный момент, или не ходить вовсе, что является нарушением правил игры [5].

Позднее появилась новая задача: необходимо реализовать проверку на тот случай, если игрок попытается походить костяшкой, которой у него нет в руке. Для этого главный модуль должен иметь свои наборы костяшек модулей (рисунок 22), так не имеет прямого доступа к ним, но может узнать о них в процессе игры. В связи с этим необходимо было внести изменения в методы GetFromShop, GetHands, SetBone и основной метод Launch – то есть, когда пользователь добавляет себе костяшку, главный модуль добавляет её в свой набор данного игрока, когда выставляет – удаляет из набора. Также после каждого хода главный модуль должен определять, мог ли пользователь походить данной костяшкой основываясь на своих данных о наборах игроков. Для этого были введены вспомогательные функции checkIsInHand, findIndex, removeFromHand.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 22 – Наборы игроков в главном модуле

Кроме того, ранее счёт рассчитывался внутри модулей игроков, что может привести к жульничеству. Данная проблема была исправлена созданием необходимой функции (рисунок 23) внутри главного модуля, которая получает итоговый счёт основываясь на своих копиях наборов игроков, о которых было сказано выше.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 23 – Функция получения итогового счёта модуля MTable

В итоге, был доработан следующий модуль: MTable.cs.

## **2.7. Визуальное оформление веб-приложения**

К визуальному оформлению были предоставлены требования: использование зеленого цвета и его оттенков, серого и белого как основную палитру и использование шрифта из семейства PTSans. Все необходимые уточнения были взяты из руководства по использованию элементов фирменного стиля Санкт-Петербургского политехнического университета Петра Великого [9] (рисунок 24)

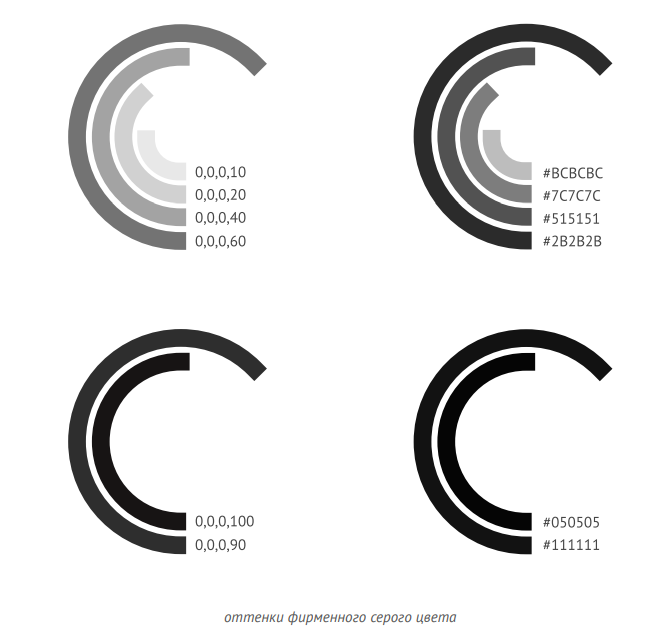
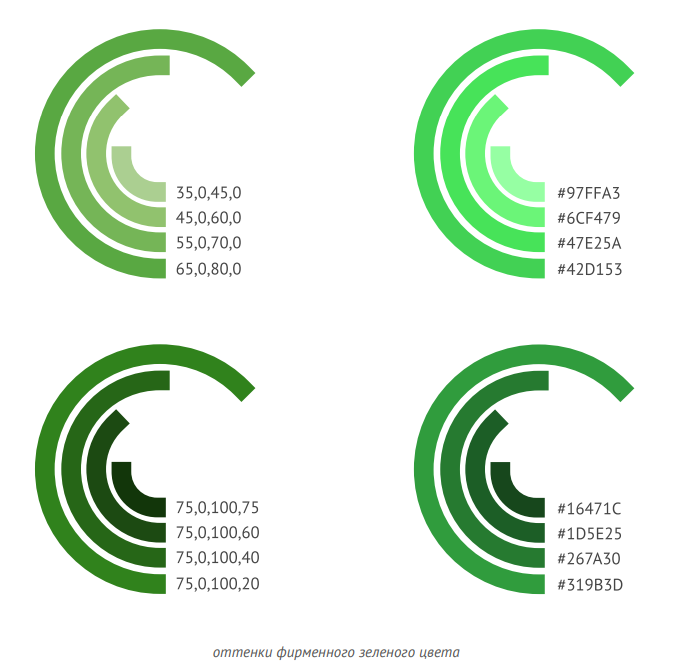


Рисунок 24 – Допустимые цвета оформления

Полученный результат представлен на рисунках 25–26. Изменения внесены во все .cshtml файлы и в основную таблицу стилей site.css.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 25 – Пример оформления страницы авторизации

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 26 – Пример оформления страницы настроек

# 3. Тестирование и апробация

## **3.1. Настройка параметров турнира**

Число игр за сторону – натуральное число (рисунки 27–28).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 27 – Ошибка при вводе отрицательного значения

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 28 – Ошибка при вводе дробного значения

Проверка установки значения (рисунки 29–30).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 29 – Установка значения отличного от значения по умолчанию

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 30 – Количество игр соответствует удвоенному установленному количеству игр за сторону

Таким образом результат соответствует заявленным требованиям.

## **3.2. Визуализация выбранной игры по домино**

Примеры проведения нескольких случайно выбранных игр (рисунки 31–33)

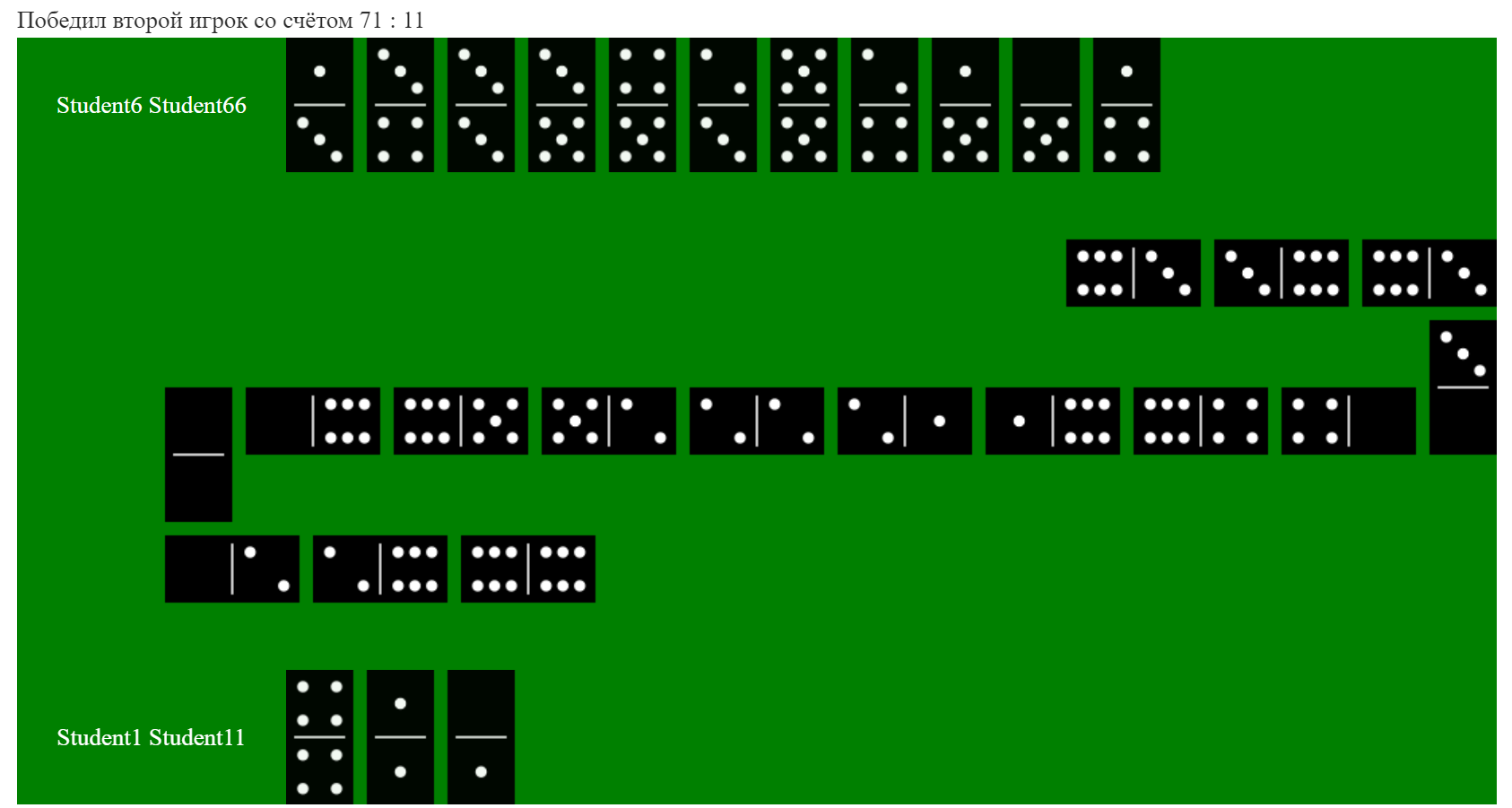


Рисунок 31 – Игра №23 первой пары

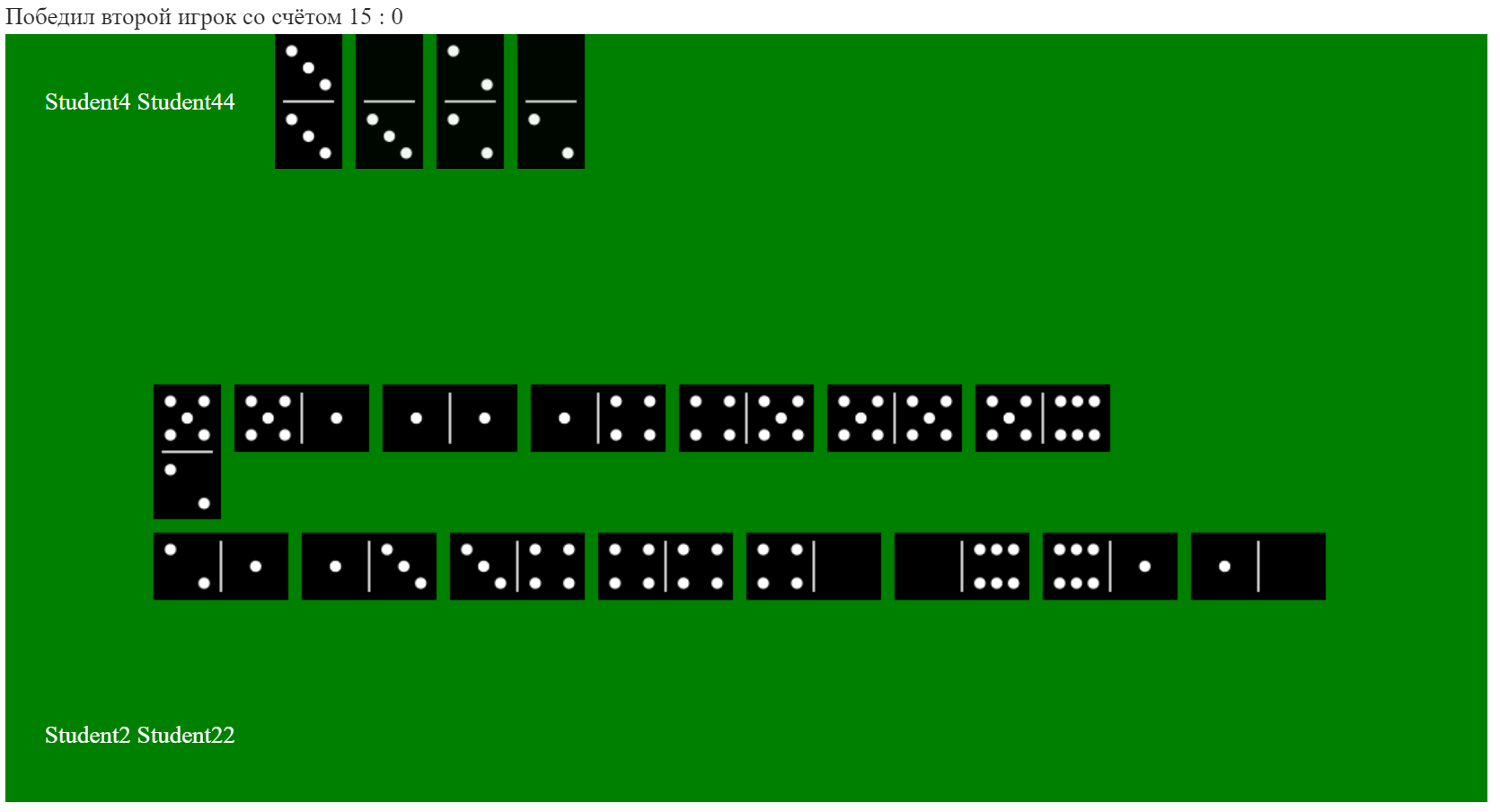


Рисунок 32 – Игра №2 второй пары

Изображение выглядит как текст, табло

Автоматически созданное описание

Рисунок 33 – Игра №7 пары в борьбе за третье место

Пример установки паузы (рисунок 34) и продолжения игры (рисунок 35)

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 34 – Игра на паузе

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 35 – Продолжение поставленной на паузу игры

Таким образом визуализация игры соответствует заявленным требованиям.

## **3.3. Визуализация турнирной таблицы**

Вид турнирной таблицы при разных количествах игроков и в разном состоянии (рисунки 36–41)

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 36 – Турнирная таблица из двух игроков

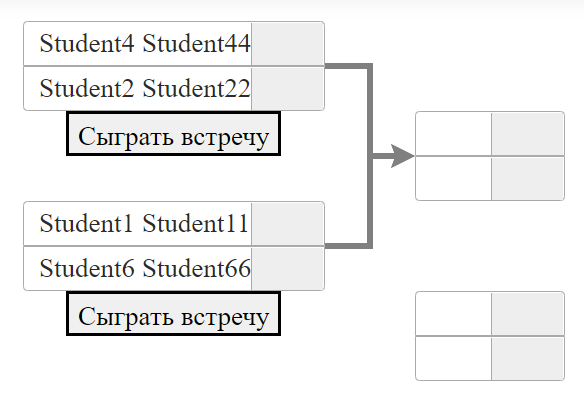


Рисунок 37 – Турнирная таблица из четырёх игроков

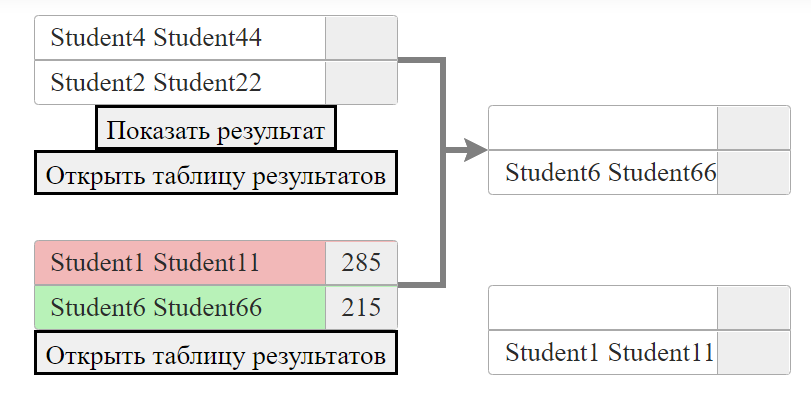


Рисунок 38 – Турнирная таблица из двух игроков в процессе турнира

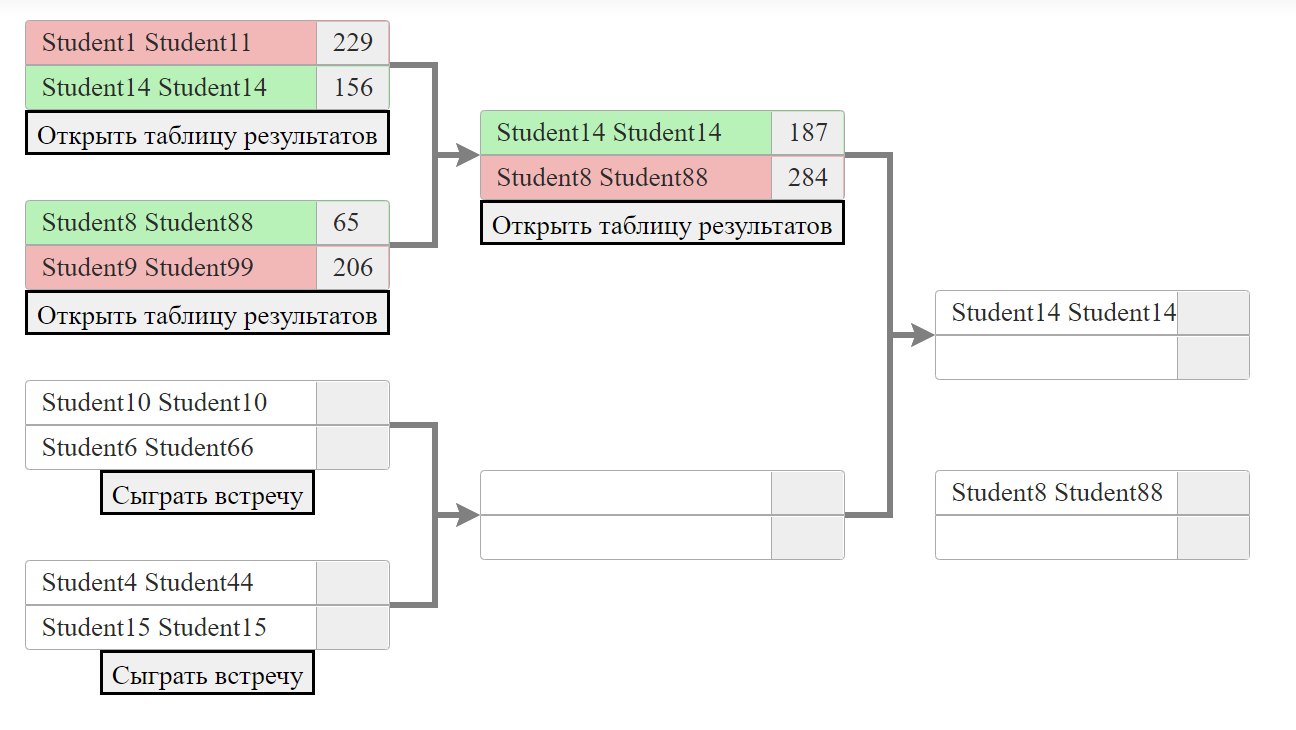


Рисунок 39 – Турнирная таблица из восьми игроков

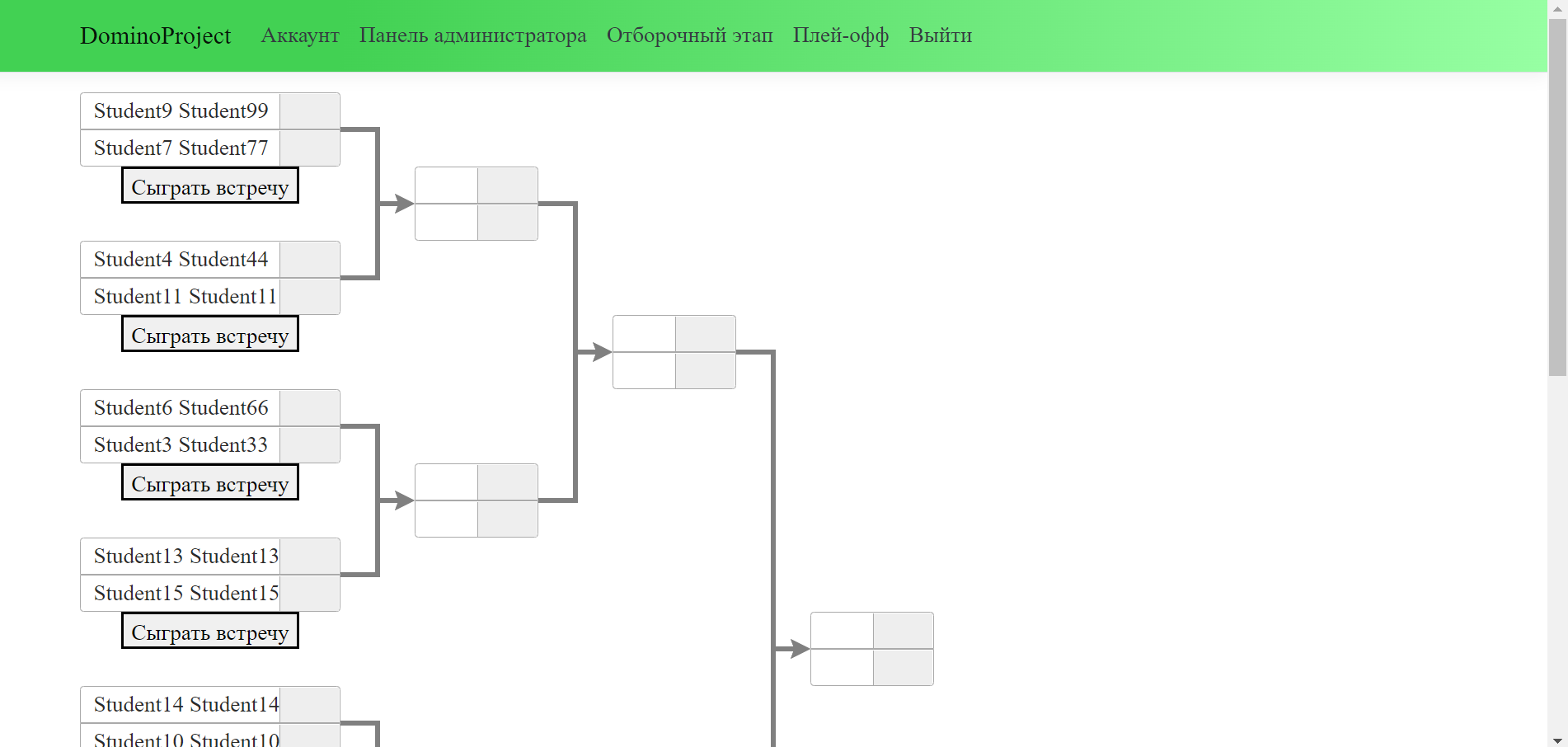


Рисунок 40 – Турнирная таблица из 16 игроков. Часть 1.

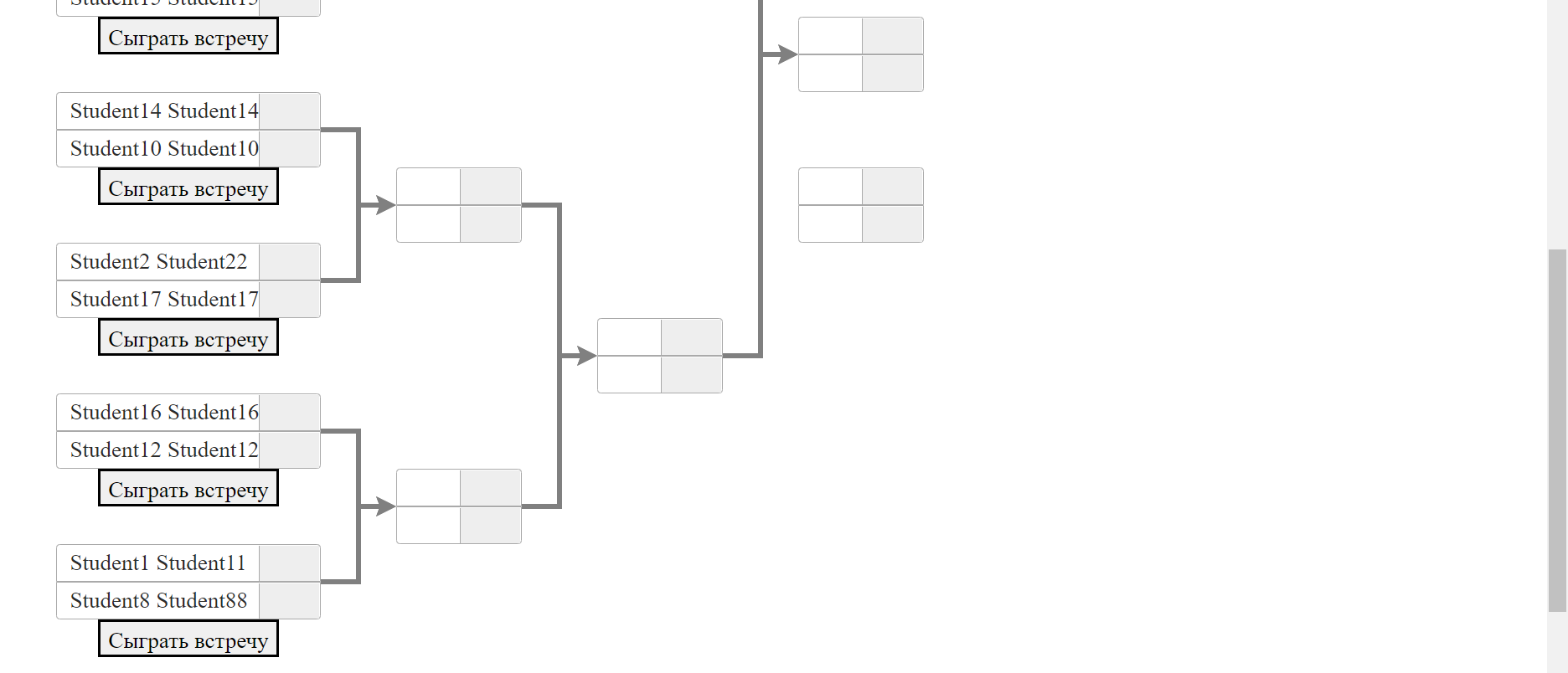


Рисунок 41 – Турнирная таблица из 16 игроков. Часть 2.

Таким образом визуализация турнирной таблицы соответствует заявленным требованиям.

## **3.4. Сохранение/восстановление турнира**

Проверка сохранения и восстановления турнира (рисунки 42-)

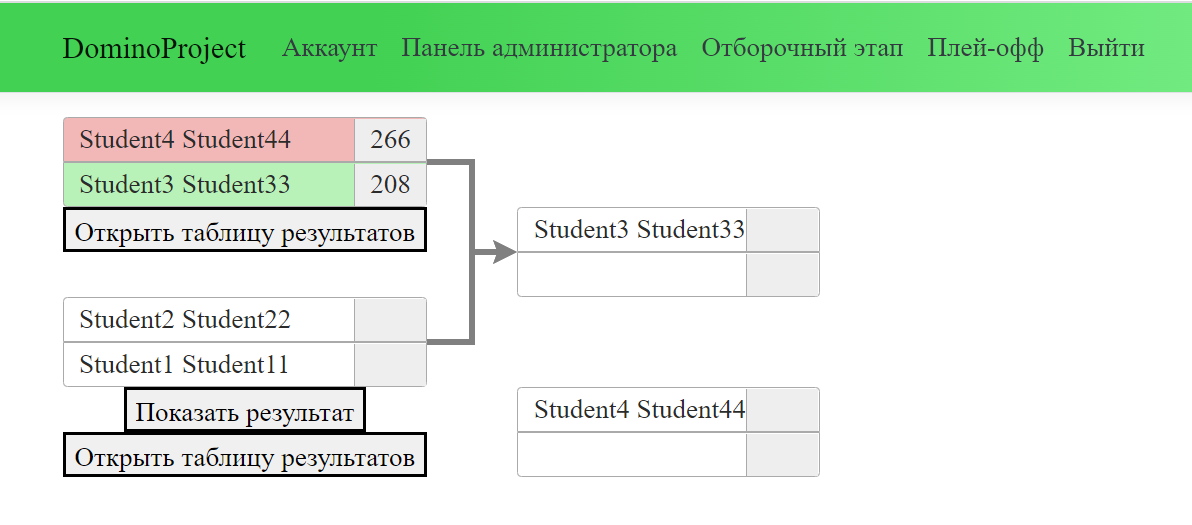


Рисунок 42 – Состояние турнира на момент сохранения

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 43 – Использование панели администратора для сохранения турнира

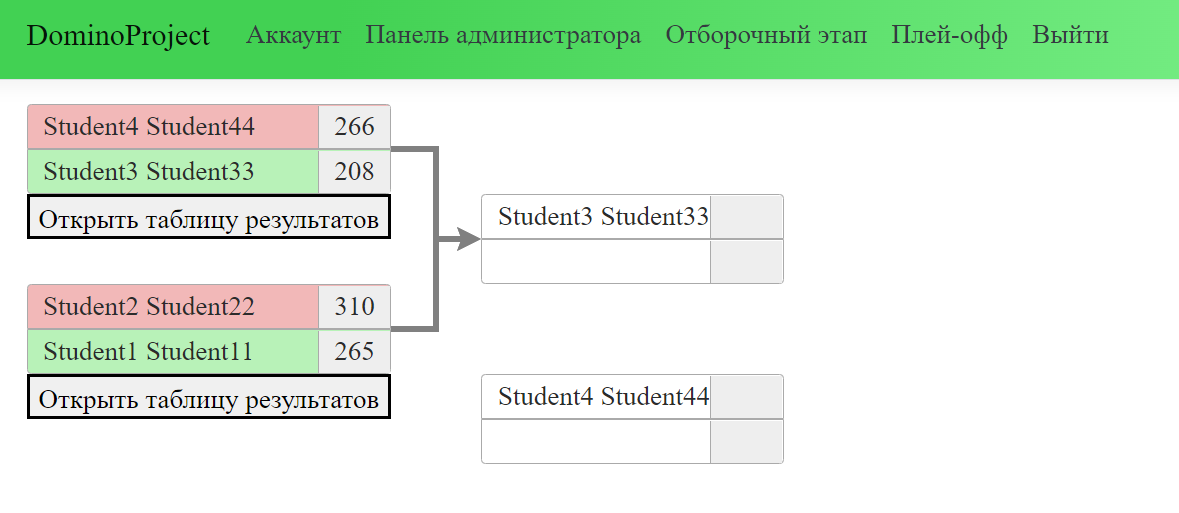


Рисунок 44 – Внесение изменений в турнир

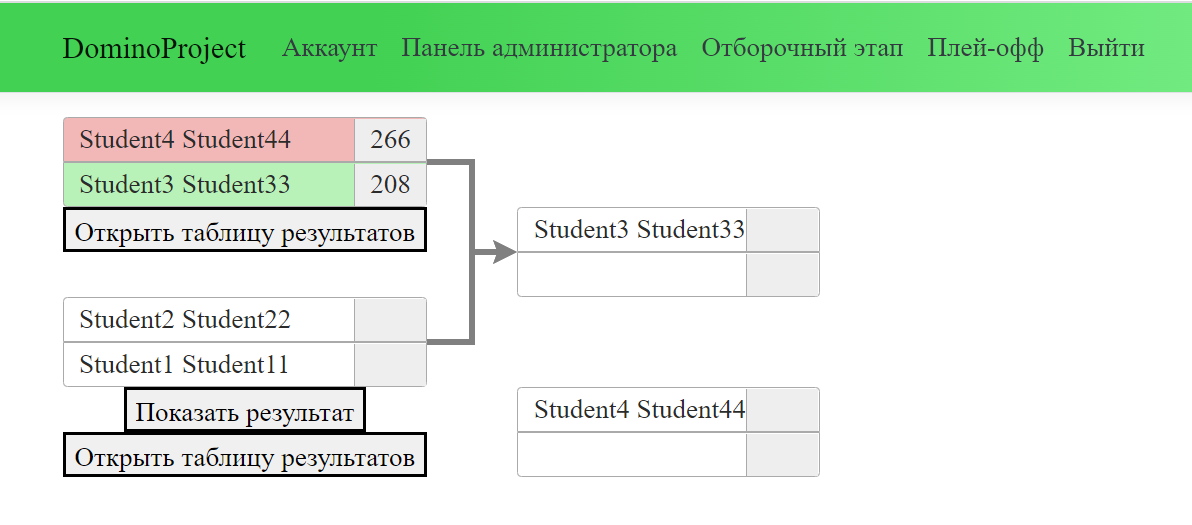


Рисунок 45 – Возврат к исходному состоянию турнира

Таким образом сохранение и восстановление турнира соответствует заявленным требованиям.

## **3.5. Валидация модулей игроков**

Таблица 1 – Результат тестирования валидации модулей игроков

|  |  |  |  |
| --- | --- | --- | --- |
| **Критерий** | **Входные данные** | **Ожидаемый результат** | **Полученный результат** |
| Наличие пространства имён DominoC | Пространство имён Domino | Вывод ошибки | Соответствует ожидаемому |
| Наличие пространства имён DominoC | Пространство имён DominoC | Отсутствие ошибки | Соответствует ожидаемому |
| Наличие пространства имён DominoC | Отсутствие пространства имён | Вывод ошибки | Соответствует ожидаемому |
| Наличие класса MFPlayer | Класс MSPlayer | Вывод ошибки | Соответствует ожидаемому |
| Наличие класса MFPlayer | Класс MFPlayer | Отсутствие ошибки | Соответствует ожидаемому |
| Наличие класса MFPlayer | Отсутствие классов | Вывод ошибки | Соответствует ожидаемому |
| Наличие методов \* | Ошибка в возвращаемом типе | Вывод ошибки | Соответствует ожидаемому |
| Наличие методов \* | Ошибка в сигнатуре метода | Вывод ошибки | Соответствует ожидаемому |
| Наличие методов \* | Ошибка в названии метода | Вывод ошибки | Соответствует ожидаемому |
| Наличие методов \* | Все методы корректны | Отсутствие ошибки | Соответствует ожидаемому |
| Наличие методов \* | Методы отсутствуют | Вывод ошибки | Соответствует ожидаемому |
| Отсутствие статического импорта | Статический импорт присутствует | Вывод ошибки | Соответствует ожидаемому |
| Отсутствие статического импорта | Статический импорт отсутствует | Отсутствие ошибки | Соответствует ожидаемому |

\* статических публичных методов void Initialize(), bool MakeStep(out MTable.SBone sb, out bool End), int GetCount(), void PrintAll()

Таким образом валидация модулей игроков соответствует заявленным требованиям.

## **3.6. Доработка модуля MTable**

Проверка проведена на неавтоматизированной платформе с целью наглядности результатов, а также ручным просмотром кода (рисунок 46).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 46 – Наличие жульничества, выявленного модулем MTable

Таким образом отлов жульничества в модуле MTable соответствует заявленным требованиям.

## **3.7. Визуальное оформление веб-приложения**

Визуальное оформление веб-приложения проводилось в соответствии с правилами [9], таким образом соответствует заявленным требованиям.

# Заключение

По результатам практики были выполнены поставленные задачи: проведён обзор средств разработки, разработаны модули, отвечающие за настройку параметров турнира, визуализацию выбранной игры, турнира по домино и приложения в целом, сохранение/восстановление турнира, валидацию модулей игроков при их загрузке, проведена доработка модуля MTable, а также тестирование и апробация полученного результата. Таким образом была достигнута цель практики в приобретении первичных профессиональных умений и навыков в соответствии с направлением подготовки.

# Список источников

1. ASP.NET documentation // Microsoft. – 2022. – URL: https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-6.0. – (дата обращения: 05.07.2022).
2. Entity Framework Core // Entity Framework Tutorial. – 2020. – URL: https://www.entityframeworktutorial.net/efcore/entity-framework-core.aspx. – (дата обращения: 05.07.2022).
3. Общие сведения ASP.NET Core MVC // Microsoft. – 2022. – URL: https://docs.microsoft.com/ru-ru/aspnet/core/mvc/overview?view=aspnetcore-5.0. – (дата обращения: 05.07.2022).
4. Адам Фримен, Стивен Сандерсон, ASP.NET MVC 4 Framework с примерами на C# для профессионалов. – М.: Вильямс, 2014. – 666 с.
5. Домино // Википедия. – 2022. – URL: https://ru.wikipedia.org/wiki/%D0%94%D0%BE%D0%BC%D0%B8%D0%BD%D0%BE. – (дата обращения: 05.07.2022).
6. ASP.NET Scaffolding in Visual Studio 2013 // Microsoft. – 2022. – URL: https://docs.microsoft.com/ru-ru/aspnet/visual-studio/overview/2013/aspnet-scaffolding-overview. – (дата обращения: 05.07.2022).
7. LeaderLine // anseki. – 2022. – URL: https://anseki.github.io/leader-line/. – (дата обращения: 05.07.2022).
8. Serializing and Deserializing JSON // Json.NET. – 2022. – URL: https://www.newtonsoft.com/json/help/html/SerializingJSON.htm. – (дата обращения: 05.07.2022).
9. Фирменный стиль // Санкт-Петербургский политехнический университет Петра Великого (СПбПУ). – 2022. – URL: https://www.spbstu.ru/university/organizational-documents/corporate-identity/. – (дата обращения: 05.07.2022).