ДИПЛОМНА РОБОТА

ДР.ПІс – 07.00.000 ПЗ

Група ПІс-2015

Касіян Р. В.

2019

ПВНЗ Університет Короля Данила

Кафедра Інформаційних технологій та програмної інженерії

УДК 796.41

ДИПЛОМНА РОБОТА

Тема ***Розробка back-end частини веб сервісу проведення жеребкувань при організації змагань***

Напрям підготовки ***6.050103 «Програмна інженерія»***
(код і назва спеціальності)

ПОЯСНЮВАЛЬНА ЗАПИСКА

ДР.ПІс – 07.00.000 ПЗ
(позначення)

Студент
Касіян Р.В.
(підпис) (дата) (розшифрування підпису)

Керівник проекту
к.т.н.,доц. Ващишак С.П.
(посада) (підпис) (дата) (розшифрування підпису)

Нормоконтроль
к.т.н.,доц. Мануляк І.З.
(посада) (підпис) (дата) (розшифрування підпису)

Допускається до захисту
Завідувач кафедри
д.т.н.,доц. Мельничук С.І.
(посада) (підпис) (дата) (розшифрування підпису)

2019

ПВНЗ УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА

Факультет *Інформаційних технологій*
Кафедра *Інформаційних технологій та програмної інженерії*
Напрям підготовки *6.050103 «Програмна інженерія»*

ЗАТВЕРДЖУЮ:
Завідувач кафедри ІТПІ
_____ *С. І. Мельничук*
"____" _____ 2019 р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ)

Студенту *Касіяну Роману Володимировичу*

1. Тема проекту (роботи) *Розробка back-end частини веб сервісу проведення жереребкувань при організації змагань*

Затверджена наказом ректора Університету Короля Данила від *15.11.2018 р. № 20/4*
2. Термін здачі студентом закінченого проекту (роботи) *10.06.2019 р.*
3. Вихідні дані до проекту (роботи) *Java, Spring framework, MySQL, JavaScript, VueJs фреймворк.*

4. Зміст пояснювальної записки (перелік питань, що їх належить розробити) *1. Призначення проекту, порівняння аналогів та опис використаних технологій. 2. Розробка структури, огляд табилиць бази даних, та середовищ розробки. 3. Розробка серверної частини та тестування функціоналу.*

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) *Постановка завдання, огляд конкурента, та його недоліків ,огляд сторінок веб сервісу, та їх структури, висновок.*

6. Консультанти з проекту (роботи), із зазначенням розділів проекту, що стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 30.10.2018

Керівник _____ Ващишак С.П.

Завдання прийняв до виконання _____ Касіян Р. В.

КАЛЕНДАРНИЙ ПЛАН

Пор №	Назва етапів дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1.	Огляд олгоритму побудови турнірніх сіток сайтів аналогів, та огляд використаних технологій	03.12.2018	
2.	Розробка структури бази даних	25.12.2018	
3.	Огляд середовищ для розробки	18.01.2019	
4.	Розробка back-end частини проекту з необхідним функціоналом	06.02.2019	
5.	Тестування функкіоналу	27.03.2019	
6.	Оформлення пояснювальної записки	11.04.2019	
7.	Оформлення графічного матеріалу та підготовка до захисту дипломної роботи	23.05.2019	

Студент-дипломник _____ *Касіян Р. В.*
(підпис) (розшифровка підпису)

Керівник проекту _____ *Ващишак С.П.*
(підпис) (розшифровка підпису)

АНОТАЦІЯ

В ході виконання завдання було розроблено back-end частину програмного забезпечення, та інтерфейс користувача для демонстрації функціоналу веб-сервісу жеребкування при організації змагань. З допомогою даного веб сервісу можна проводити турнірні змагання з жеребкуванням та побудовою турнірної сітки.

SUMMARY

During the task, a back-end of the software and a user interface were developed to demonstrate the function of the web service during the organization of the contest. With this web service, you can arrange tournament competitions and build a tournament grid.

РЕФЕРАТ

Розрахунково-пояснювальна записка: 56 сторінок, 29 рисунків.

Об’єктом дослідження є процес проведення жеребкувань для турнірних змагань за олімпійською системою.

Метою дипломної роботи є дослідження веб сервісів жеребкувань для турнірних змагань, та побудови турнірної сітки для них, а також написання back-end частини програмного забезпечення для максимально простого проведення турнірів та відображення їхніх результатів в інтернеті.

Згідно до поставленого завдання проводиться дослідження по створенню турнірних змагань з жеребкуванням та побудовою турнірної сітки для визначення максимально підходящої системи проведення турнірних змагань. Згідно даного дослідження розроблено back-end частину для створення турнірних змагань з жеребкуванням що спростить публікацію інформації про турнір, та його результатів.

Турнірна сітка, back-end, турніри, команди, жеребкування, Java, Spring, MySQL.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ 7
ВСТУП 8
1 ПРИЗНАЧЕННЯ ПРОЕКТУ, ПОРІВНЯННЯ АНАЛОГІВ ТА ОПИС ВИКОРИСТАНИХ ТЕХНОЛОГІЙ 10
1.1 Організація турнірних сіток в спортивних змаганнях 10
1.2 Призначення проекту 14
1.3 Порівняння аналогів 15
1.4 Постановка завдання на дипломне проектування 17
1.5 Використані технології 18
2 РОЗРОБКА СТРУКТУРИ, ОГЛЯД ТАБИЛИЦЬ БАЗИ ДАНИХ, ТА СЕРЕДОВИЩ РОЗРОБКИ 21
2.1 Розробка структури бази даних 21
2.2 Огляд таблиць бази даних 23
2.3 Компоненти та бібліотеки користувацької частини веб-додатку 32
2.4 Вибір середовищ розробки програм 33
3 РОЗРОБКА СЕРВЕРНОЇ ЧАСТИНИ, ТА ТЕСТУВАННЯ ФУНКЦІОНАЛУ 39
3.1 Розробка backend частини веб-додатку 39
3.2 Тестування функціоналу 50
ВИСНОВКИ 61
ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 62

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

БД - База даних

СУБД - Система управління базами даних

JS - JavaScript

HTML - HyperText Markup Language

CSS - Cascading Style Sheets

ВСТУП

Актуальність теми. Наступила ера електронних технологій, в даний момент це невід'ємна складова сучасного суспільства. В інтернеті ми спілкуємося, вчимося чомусь новому або ділимося своїм досвідом в чомусь з людьми, робимо покупки тощо. Так чи інакше кожна сфера діяльності пов'язана з новими технологіями, і ми не уявляємо своє життя без них. У світі існує багато різноманітних видів змагань де розподіляються призові місця, та переможці. Це може бути футбол чи шкільна олімпіада з шахмат, або будь яка інша спортивна дисципліна і тут сучасні технології можуть посприяти в вирішенні багатьох проблем, наприклад ми можемо розповсюдити інформацію про захід чи змагання, дати можливість зареєструватися на нього, та подивитися перебіг змагання та фінальні результати, що у моєму випадку і потрібно.

Аналогом мого завдання послугувала платформа з функціоналом по автоматичному створенню турнірних сіток Faceit, який не підходить під поставлені задачі, тому було прийнято рішення розробити свою турнірну сітку.

Турнірні сітки невід'ємна складова будь яких спортивних чи кібер-спортивних змагань. Вона дає нам можливість більш наглядно побачити які команди беруть участь у турнірі, яка команда з якою буде змагатися, хто буде переможцем цього змагання. Тобто, турнірна сітка дає нам максимальне графічне відображення інформації про турнір. Тому що графічна інформація набагато легша у сприйманні ніж суцільна текстова. На турнірна сітці ми бачимо зв'язки між командами які грають, ми можемо побачити варіанти які команди можуть зустрітися в подальшому.

Уявити турніри без турнірної сітки це як уявити футбольний матч, де коментатори будуть розказувати хто на якій позиції в даний момент, куди копнули м'яч і яка в нього траєкторія польоту але при ньому не бачити це на власні очі, це звичайно можна собі уявити, але набагато легше просто побачити це на власні очі, ніж почути чи прочитати в газеті. Тож з впевненістю можна

заявити, що турнірні сітки невід'ємна складова турнірних змагань так як без неї буде набагато складніше вникнути в перебіг подій на змаганні. В даний момент майже кожен турнір чи змагання, при бажанні ми можемо переглянути в інтернеті в режимі реального часу чи коли нам цього захочеться. Відповідно перед нами навіть не постає питання про те що турнірна сітка також має бути доступна для перегляду будь яким бажаючими у будь який час. І з допомогою інтернету ми цю можливість маємо.

У сучасному світі існує безліч ресурсів які розповсюджують інформацію. Якщо раніше ми могли бачити новини по телевізорі, почути по радіо чи прочитати в газеті, то тепер у кожної сучасної людини є смартфон чи комп'ютер з доступом до всесвітньої мережі інтернет, де ми можемо в любий момент знайти потрібну нам інформацію - це може веб чи мобільний додаток. Найпопулярнішим видом розповсюдження інформації є веб додаток. Так як він є найбільш універсальним і інформація може бути переглянута як на комп'ютері чи ноутбуці так і на смартфоні, планшеті з доступом в інтернет. Тому було прийнято рішення розробити саме веб додаток, тому що він дозволяє охопити максимально широку категорію користувачів які зможуть без будь яких труднощів переглянути турнірну сітку будь якого турніру, присутнього на даному ресурсі з будь якої точки світу.

Об'єктом дослідження є процес проведення жеребкувань для турнірних змагань за олімпійською системою.

Мета роботи - дослідження веб сервісів жеребкувань для турнірних змагань, та побудови турнірної сітки для них, а також написання back-end частини програмного забезпечення для забезпечення максимально простого проведення турнірів та відображення їхніх результатів в інтернеті.

Методи дослідження - використовувалися такий метод дослідження як метод порівняння, за допомогою якого порівнювався функціонал сайтів аналогів для проведення турнірів, та вибирався найнеобхідніший функціонал який необхідний для розробки для мого проекту.

1 ПРИЗНАЧЕННЯ ПРОЕКТУ, ПОРІВНЯННЯ АНАЛОГІВ ТА ОПИС ВИКОРИСТАНИХ ТЕХНОЛОГІЙ

1.1 Організація турнірних сіток в спортивних змаганнях

Для того щоб розпочати розробку програмного забезпечення для побудови турнірної сітки, необхідно визначити за допомогою якої системи ця турнірна сітка буде будуватися. Існує три найпопулярніші системи організації турнірів: кругова, олімпійська та швейцарська система. Для побудови турнірної сітки я обрав олімпійську систему оскільки вона являється найбільш підходящою для проведення швидких турнірів, тобто турнірів які можна провести буквально за пару днів та при цьому справедливо визначивши переможця турніру.

Олімпійська система або плей-офф (англ. Playoff) у спортивних змаганнях - система розіграшу (організації змагань), при якій учасник вибуває з турніру після першого ж програшу (за підсумками однієї гри або серії з декількох ігор між двома учасниками, що дозволяє однозначно визначити безумовного переможця). Забезпечує виявлення переможця за мінімальне число турів і сприяє напруженій боротьбі в турнірі [1].

Порядок розіграшу. Кількість учасників розіграшу плей-офф обов'язково має бути ступенем двійки (2, 4, 8, 16, 32 і так далі). У випадку іншого числа команд проводяться один або кілька попередніх кіл розіграшу, в результаті яких загальна кількість учасників скорочується до найближчої ступеня двійки. У багатьох видах спорту цей попередній етап іменується "Регулярним сезоном". В індивідуальних видах спорту, де практикується присвоювання гравцям особистих рейтингів, можливий відбір у плей-офф потрібної кількості гравців з найбільшим на момент відбору рейтингом.

Двійковий логарифм числа учасників визначає число кіл розіграшу (турів): для 2 учасників - один, для 4 - два, для восьми - три, для 16 - чотири. Загальне число ігор на одиницю менше числа учасників. Кола розіграшу зазвичай

називаються за кількістю пар учасників: для 1 пари - "фінал" (він визначає переможця), для 2 пар - "півфінал", для 4 пар - "чвертьфіналу", для 8 пар - "одна восьма фіналу", для 16 пар - "одна шістнадцята фіналу" і так далі.

У кожному колі з учасників складаються пари, що грають між собою (це може бути одна гра, або матч з декількох ігор, в якому перемагає набрав більше очок; принципово важливо, що результат туру завжди певний - нічиїх бути не може).

З кожної пари в наступне коло виходить переможець, а переможений вибуває з турніру.

Учасник, який виграв фінальний коло, стає переможцем, його останній суперник отримує друге місце. Якщо регламент турніру вимагає присвоєння та третього місця, то проводиться додатковий матч за нього між двома учасниками, програли в двох півфіналах.

Принципи відбору пар на перший етап можуть бути різні: найчастіше застосовується жеребкування, хоча можливий відбір за рейтингом. Пари на другому і наступних етапах можуть складатися або за тими ж правилами, що і на першому (на кожному етапі проводиться нове жеребкування пар або відбір за рейтингом), або за принципом "жорсткої сітки" - сітка турніру готується заздалегідь, в ній жорстко задається, як будуть складатися пари з переможців кожного етапу, і всі розподілення пар однозначно визначається порядком заповнення сітки на першому етапі.

До переваг плей-офф можна віднести мінімальну кількість ігор, в порівнянні з іншими варіантами турнірів, а також "безкомпромісність" - у ньому немає ні можливості, ні сенсу в договірних нічиїх. Плей-офф націлений на максимально швидке виявлення найсильнішого і забезпечує справедливе (якщо вважати силу учасників постійною і не залежить від того, хто з ким грає) присвоєння першого місця - його займає той, хто нікому не програв, у той час як всі інші учасники турніру комусь програють.

Незручність плей-офф полягає в жорстких вимогах до кількості учасників. Якщо ця кількість не відповідає нормі, то єдиний вихід - за жеребом видати

частини учасників технічні перемоги або технічні поразки у першому колі, що ще більше збільшує вплив випадкового фактора на результат турніру. Єдина альтернатива - випереджати турнір плей-оф серією попередніх ігор за вихід в основний турнір.

Плей-офф абсолютно не підходить для турнірів, де важливо забезпечити справедливий розподіл всіх місць, а не тільки першого-третього. По-перше, в плей-офф на розподіл місць, крім першого (особливо - останніх), надзвичайно сильно впливає порядок вибору пар. У разі жеребкування останні місця розподіляються практично випадково: слабкий учасник, якому жереб дає порівнянних за силою противників, легко може піднятися вище сильного, якому в першому ж колі дістався ще більш сильний суперник.

Спроба замінити жереб на якусь осмислену систему підбору пар по рейтингах робить турнір передбачуваним тому в моєму завданні я буду використовувати саме його. Є два варіанти такого підбору: або "сильний проти слабкого" - у кожному колі учаснику з високим рейтингом дістається супротивник з низьким (конкретних алгоритмів підбору може бути декілька), або "рівний з рівним" - найсильнішому дають у пару другого, третього - четвертого і так далі. У першому випадку велика частина зустрічей виявляється передбачуваною, а тому нецікавою, у другому - половина найсильніших відсівається на перших етапах і передбачуваним виявляється фінал. Тому завжди використовують перший варіант, щоб глядач у фіналі побачив справжню гру найсильніших, а не сірий фінал нікому не цікавих команд. Якщо необхідно конкретизувати місця, зайняті учасниками, доведеться проводити додаткові ігри, через що втрачається основна перевага плей-офф - швидкість.

Плей-офф широко застосовується в національних та міжнародних змаганнях з ігрових видів спорту. Одна з назв плей-офф - "олімпійська система", пов'язано з тим, що цей порядок розіграшу є основним для ігрових видів спорту на Олімпіаді.

За системою плей-офф розігрується велике число титулів і вищих нагород у командних змаганнях з ігрових видів спорту, таким як хокей, футбол та інші.

Для того щоб краще зрозуміти побудову турнірної сітки при використанні олімпійської системи можна поглянути на приклад турнірної сітки на 16 команд зображеної на рисунку 1.1 [2]. По зображенню (рис. 1.1) можна зрозуміти що вона максимально проста в побудові. В ній немає жодних додаткових матчів за місця, а її перебіг можна дуже просто охарактеризувати, виграв пройшов в наступний етап, програв не пройшов. Таким чином у фінальному матчі зустрінуться найсильніші команди, які не програли жодного матча.

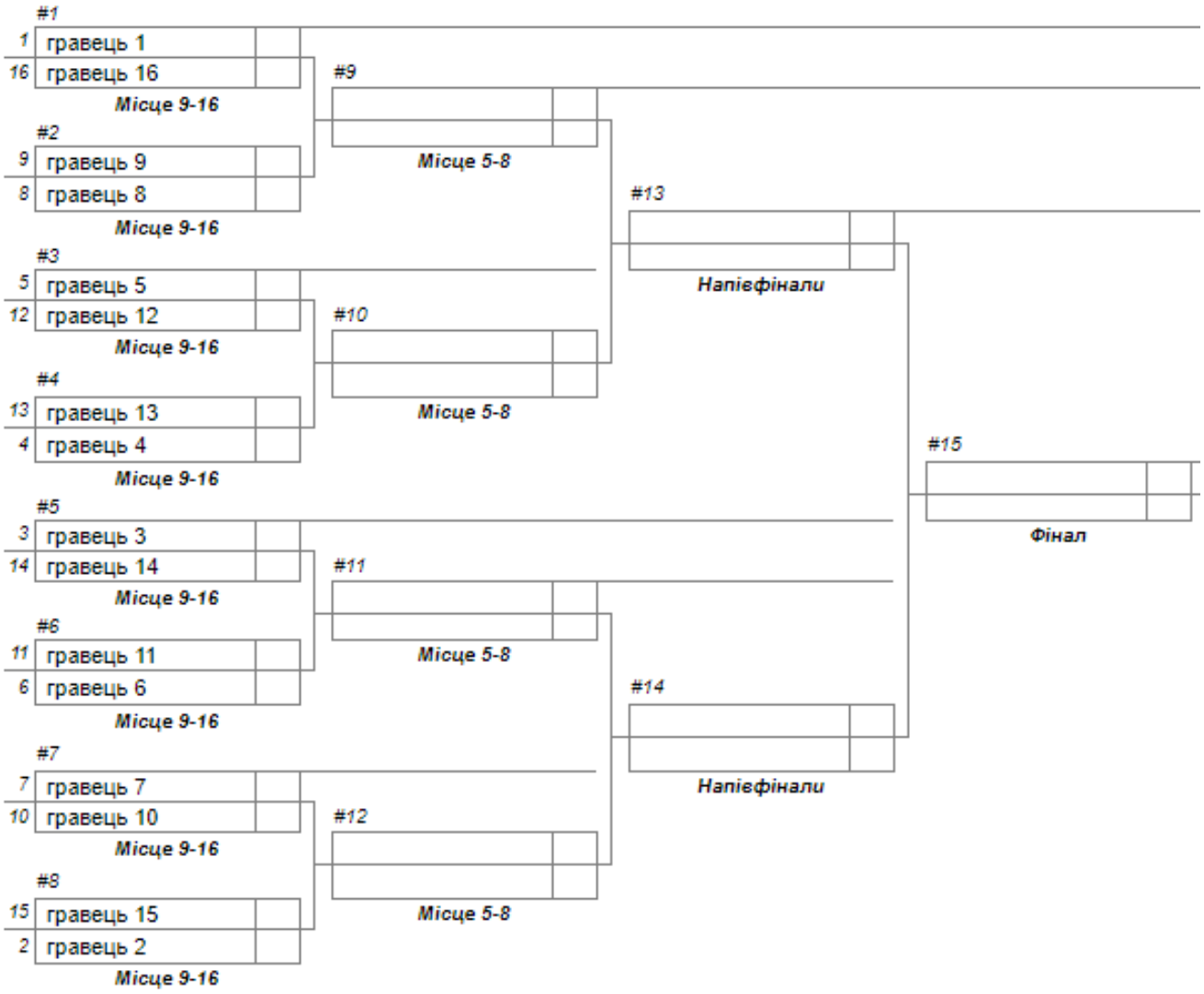

Рисунок 1.1 – Приклад турнірної сітки

1.2 Призначення проекту

Розмовляти про сфери застосування ІТ технологій технологій можна нескінченно довго. ІТ технології заполонили всі сфери діяльності людини навчання, розваги, робота. Якщо раніше люди писали листи і йшли на пошту їх відправляти, після чого вони тільки через декілька днів в кращому випадку приходили відправнику, то зараз інформація циркулює у електронному вигляді. Ми можемо надіслати листа не виходячи з дому за допомогою електронної пошти чи любого сервіса миттєвих повідомлень. Забезпечення проведення турнірів також не стоїть на місці. Якщо на початку мунулого століття щоб переглянути футбольний матч потрібно було прийти на стадіон на якому він і відбувався, то зарази ми можемо переглянути цей матч в інтернеті в режимі реального часу чи тоді коли ми цього захочемо. Так само щоб швидко дізнатися результати конкретного матчу або турніра в цілому з результатами ігор кожної команди на кожному етапі турніру. Таким чином, якщо турнір матч чи змагання проходить в іншому місті чи країні, або просто у вас немає можливості його відвідати, ви однаково зможете переглянути його в інтернеті чи просто подивитися результат матчу в любий зручний вам час.

Тож щоб спростити доступ до даних про перебіг турніру було прийнято рішення розробити турнірну сітку яка б давала можливість переглянути результати перебігу турніру будь-якому бажаючому в будь-який момент. Щоб вона виглядала максимально зрозуміло було використано метод графічного подання даних, оскільки він дозволяє максимально просто і логічно пре доставити інформацію про перебіг змагання користувачу. Таким чином з допомогою турнірної сітки ми можемо надати можливість максимально просто і швидко переглянути перебіг турніра.

Завданням роботи є створення веб сайту для створення турнірних сіток розрахованих на 64 команди. Також повинна бути реалізована система жеребкування за допомогою якої кожній команді буде рандомно визначено порядкове місце серед конкурсантів, та також рандомне визначено конкурента.

Сайт повинен бути максимально простим, та зручним як для користувача, так і для адміністратора.

1.3 Порівняння аналогів

Для порівняння функціоналу я обрав платформу Faceit [3] головна сторінка якого зображена на рисунку 1.2.

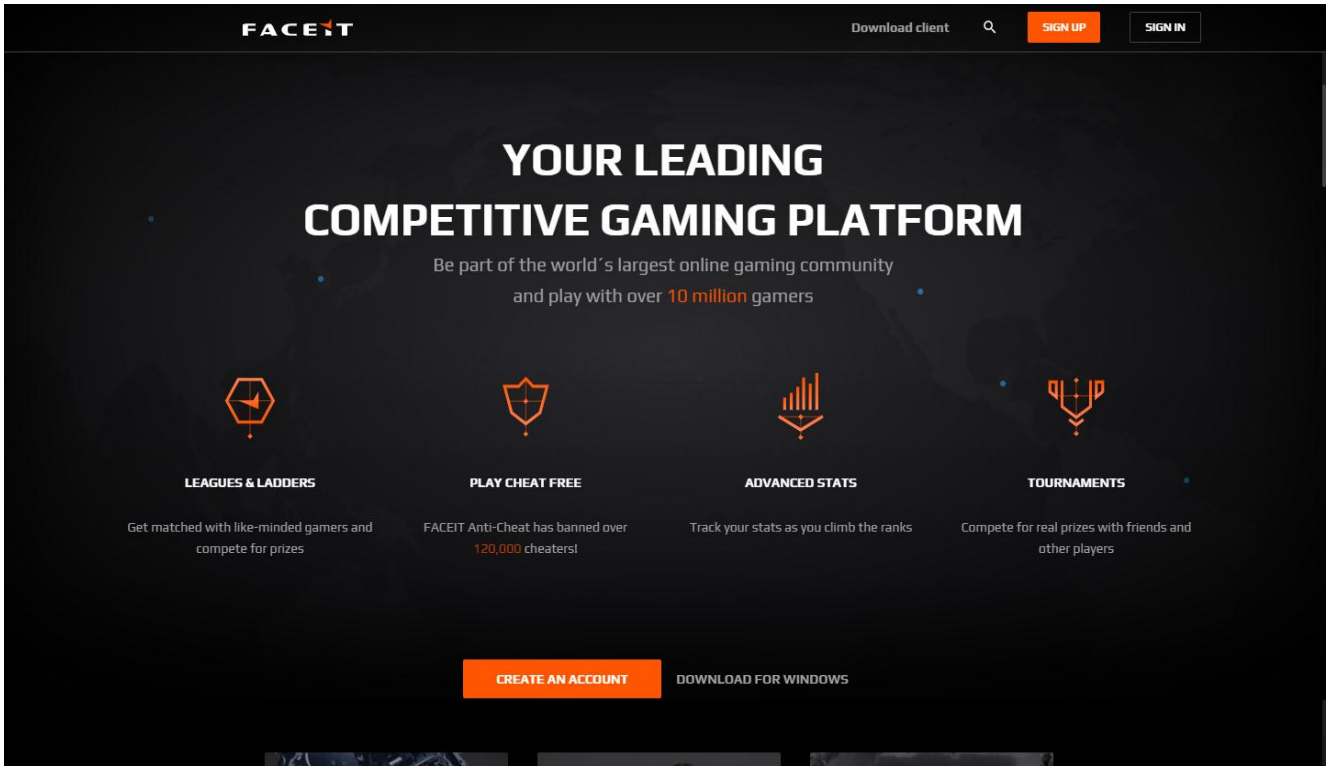

Рисунок 1.2 - Головна сторінка Faceit

Faceit є провідною незалежною конкурентною ігровою платформою для онлайн геймерів з багатокористувацької PvP-програмою з більш ніж вісьмома мільйонами користувачів, і в цілому дванадцять мільйонів сесій онлайн-ігор щомісяця. Дана платформа має дуже широкий функціонал. Тут присутні соціальні функції, що дозволяють користувачам додавати один одного в друзі, та обмінюватися повідомленнями. А оскільки платформа є геймерською, то на ній користувачі збираються в команди, а платформа надає виділені сервери з спортивними настройками для визначених дисциплін, де уже безпосередньо і

збираються гравці. На серверах Faceit окрім звичайних ігор проходять і турнірні змагання. Для цього на Faceit створені всі умови, оскільки на даній площадці є функціонал, пов'язаний з турнірними сітками. Гравці можуть формувати команди, та приймати участь у різноманітних турнірах, де автоматично формуються турнірні сітки і переможці можуть автоматично отримувати призові за здобуте місце, те як виглядає турнірна таблиця зображено на рисунку 1.3.

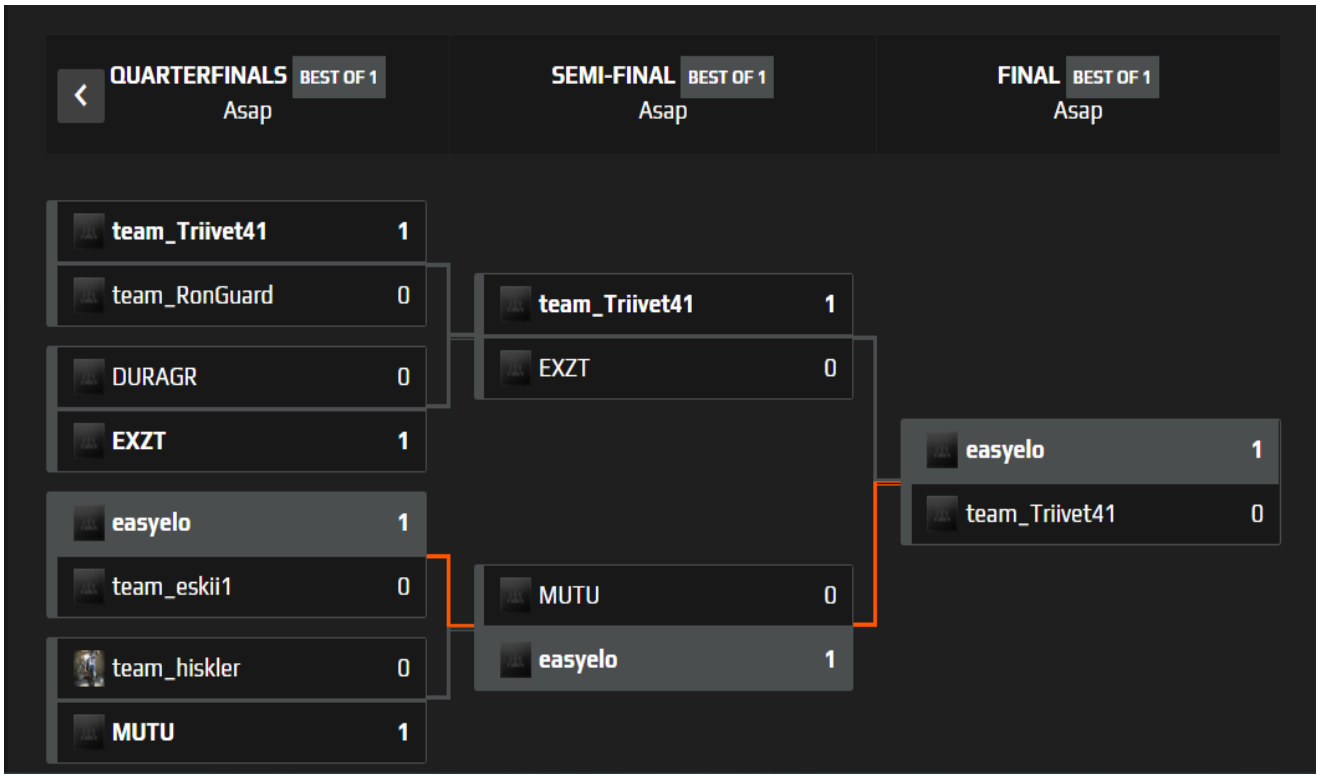

Рисунок 1.3 - Скріншот турнірної сітки на Faceit

На даній платформі кожен бажаючий може створити турнір по певній кібер спортивній дисципліні з заданою кількістю учасників, та по бажанню добавити призовий фонд, задати країни що можуть приймати участь, час та дату коли турнір розпочнеться.

Ми можемо сказати, що Faceit досить непогана платформа, але у неї є недоліки. Головним недоліком даної платформи є те, що турнірні сітки не є універсальними, оскільки вони розраховані тільки на кібер спортивні турніри, де результати автоматично вносяться в сітку. Таким чином, ми не можемо вручну ввести результати матчу, що уже дає нам підстави задуматись над власним

сервісом для створення універсальних турнірних сіток, де можна буде створити турнір в незалежності від того футбольний він, кібер спортивний чи з будь якої інакшої дисципліни.

Наступним недоліком платформи є відсутність української локалізації, що явно не являється плюсом даної платформи, оскільки не всі люди розуміють англійську, та зможуть розібратися в тому що і де знаходиться на сайті. Також, можливо, і незначним але мінусом є те, що при потребі ми не зможемо користуватися нею локально.

1.4 Постановка завдання на дипломне проектування

До виконання було отримано завдання розробити back-end частину для веб-сервісу для організації турнірних змагань, та проведення турнірних жеребкувань. Основною метою проекту є оптимізація проведення турнірних змагань.

Турнірна сітка має бути деревовидною структурою даних, що відображатиме команди або учасників змагання поетапно. Після кожного етапу певна кількість команд повинна вибувати доти, поки не залишиться тільки один переможець.

Жеребкування необхідне для рандомного перемішування учасників турніру на його початку для визначення порядкового місця команди, та того хто буде її суперником.

Також необхідно провести аналіз завдання та потреб користувача, щоб визначити функціонал програмного забезпечення яке необхідно для вирішення поточного завдання, що буде задовольняти потреби як користувача так і адміністратора сайту.

Програмне забезпечення повинно містити такий функціонал:

вхід для адміністратора;

адміністративні функції;

додавання та видалення турнірів;

дату час та місце проведення турніру;

додавання команд до турніру;

рандомне перемішування команд;

внесення даних по переможцях матчу і турніра в цілому.

Щоб розпочати розробку даного програмного забезпечення необхідно визначити найбільш підходящу мову написання, та допоміжні фреймворки які дозволять реалізувати необхідний функціонал, базу даних в якій будуть зберігатися необхідні нам дані, а також необхідно визначити середовища для розробки в якому цей функціонал буде реалізовуватися.

1.5 Використані технології

Для написання серверної частини була мною вибрана мова програмування Java, так як вона відповідає всім вимогам і дозволяє реалізувати розробку серверної частини для турнірної сітки.

Java - об'єктно-орієнтована мова програмування, випущена 1995 року компанією "Sun Microsystems" як основний компонент платформи Java. З 2009 року мовою займається компанія "Oracle", яка того року придбала "Sun Microsystems". В офіційній реалізації Java-програми компілюються у байт-код, який при виконанні інтерпретується віртуальною машиною для конкретної платформи.

Java є компілюючою мовою програмування, тобто створені нею програми за допомогою компілятора перетворюються у виконуваний файл. Процес такого перетворення складається з двох етапів. Спочатку, код програми на мові Java за допомогою програми-транслятора переводиться у так званий байт-код, який може бути перетворений у машинний код за допомогою віртуальної машини Java - Java Virtual Machine (JVM). Це дозволяє в подальшому виконувати програму написану на мові програмування Java на комп’ютерах, які працюють

під керуванням різноманітних операційних систем. Єдиною вимогою для запуску програми є встановлення на комп’ютері відповідної віртуальної машини Java. Таким чином досягається максимальна мобільність створених програм та їх незалежність від платформи виконання [4-7].

З допомогою фреймворка Spring було реалізовано доступ до бази даних, запити та звязок між об’єктами.

Spring Framework - це програмний каркас з відкритим кодом та контейнери з підтримкою інверсії управління для платформи Java. Основні особливості Spring Framework можуть бути використані будь-яким додатком Java, але є розширення для створення веб-додатків на платформі Java EE.

Основні особливості Spring Framework можуть бути використані будь-яким додатком Java, але є розширення для створення веб-додатків на платформі Java EE. Незважаючи на це, Spring Framework не нав'язує якоїсь конкретної моделі програмування, Spring Framework став популярним в спільноті Java як альтернатива, або навіть доповнення моделі Enterprise JavaBean (EJB).

Spring Framework складається з кількох модулів, які надають широкий спектр послуг:

контейнер Інверсії управління: Конфігурація компонентів додатків і управління життєвим циклом об'єктів Java, здійснюється головним чином через Інверсію управління;

аспектно-орієнтоване програмування: дозволяє реалізувати наскрізні процедури;

доступ до даних: робота з реляційною системою управління базами даних на платформі Java з використанням JDBC і об'єктно-реляційні відображення та інструментів з NoSQL баз даних;

управління транзакціями: об'єднує кілька API, управління транзакціями та координує операції для Java-об'єктів;

модель-Вигляд-Управління (Model-View-Controller): програмний каркас на основі HTTP сервлета, що забезпечує створення веб-додатків і веб-служб RESTful;

аутентифікація і авторизація: налаштовуванні процесів безпеки, які підтримують цілий ряд стандартів, протоколів, інструментів і практик за допомогою під проекту Spring Security (колишня система безпеки AcerI для Spring);

віддалене керування: конфігураційний вплив і управління Java-об'єктами для місцевої (локальної) або віддаленої конфігурації через JMX;

тестування: підтримка класів для написання юніт-тестів та інтеграційних тестів [8].

Spring Data - додатковий зручний механізм для взаємодії з сутностями бази даних, організації їх в репозиторії, вилучення даних, зміна, в яких випадках для цього буде достатньо оголосити інтерфейс і метод в ньому, без імплементації [9].

Для обміну даними між сервером і клієнтом використовував REST - запити. REST (скорочення від англ. Representational State Transfer - "передача стану уявлення") - архітектурний стиль взаємодії компонентів розподіленого додатка в мережі. REST є узгоджений набір обмежень, що враховуються при проектуванні розподіленої гіпермедіа-системи [10].

Для складання проекту використовувався Apache Maven.

Apache Maven - це інструмент для автоматизованого складання роботи, створення звітів, і написання документації до проекту. Він створює проекти за допомогою Object Model Model (POM) і наборів плагінів, які є у всіх проектах, які використовують Maven, забезпечуючи тим самим єдину систему зборки. Як тільки ви дізнаєтеся, як будувати один проект у Maven, ви знаєте, як зібрати всі проекти на Maven. Це зберігає масу часу при спробі побудувати інші проекти [11].

В якості сервера було використано Tomcat.

Tomcat є веб-сервером (може обробляти HTTP-запити / відповіді) і веб-контейнер (реалізує Java Servlet API, також званий servletcontainer) в одному. Деякі можуть назвати це сервером додатків, але він безумовно не є повнофункціональним сервером додатків Java EE [12].

2 РОЗРОБКА СТРУКТУРИ, ОГЛЯД ТАБИЛИЦЬ БАЗИ ДАНИХ, ТА СЕРЕДОВИЩ РОЗРОБКИ

2.1 Розробка структури бази даних

В проекті для управління та зберігання даних я скористався СУБД MySQL.

СУБД - це система керування базами даних (англ. Database Management System, DBMS) що представляє собою набір повязаних між собою баз даних та програм для доступу до цих даних. Також СУБД MySQL надає можливості створення, збереження, оновлення та пошуку інформації в базах даних з контролем доступу до даних [13-14].

MySQL - вільна система керування реляційними базами даних. MySQL був розроблений компанією "TcX" для підвищення швидкодії обробки великих баз даних. Ця система керування базами даних (СУБД) з відкритим кодом була створена як альтернатива комерційним системам. MySQL з самого початку була дуже схожою на mSQL, проте з часом вона все розширювалася і зараз MySQL - одна з найпоширеніших систем керування базами даних. Вона використовується, в першу чергу, для створення динамічних веб-сторінок, оскільки має чудову підтримку з боку різноманітних мов програмування [15].

Перед тим як розпочати розробку проекту було спроектовано базу даних яка б відповідала всім необхідним критеріям. База даних повинна містити дані про турніри, та матчі всередині них, також важлива послідовність додавання даних та їхні зв’язки.

У відповідності до поставленого завдання я розробив схему бази даних в середовищі MySQL. Розроблена схема бази даних з таблицями, полями та їхніми типами даних, а також зв'язками одного з одним зображені на рис 2.1.

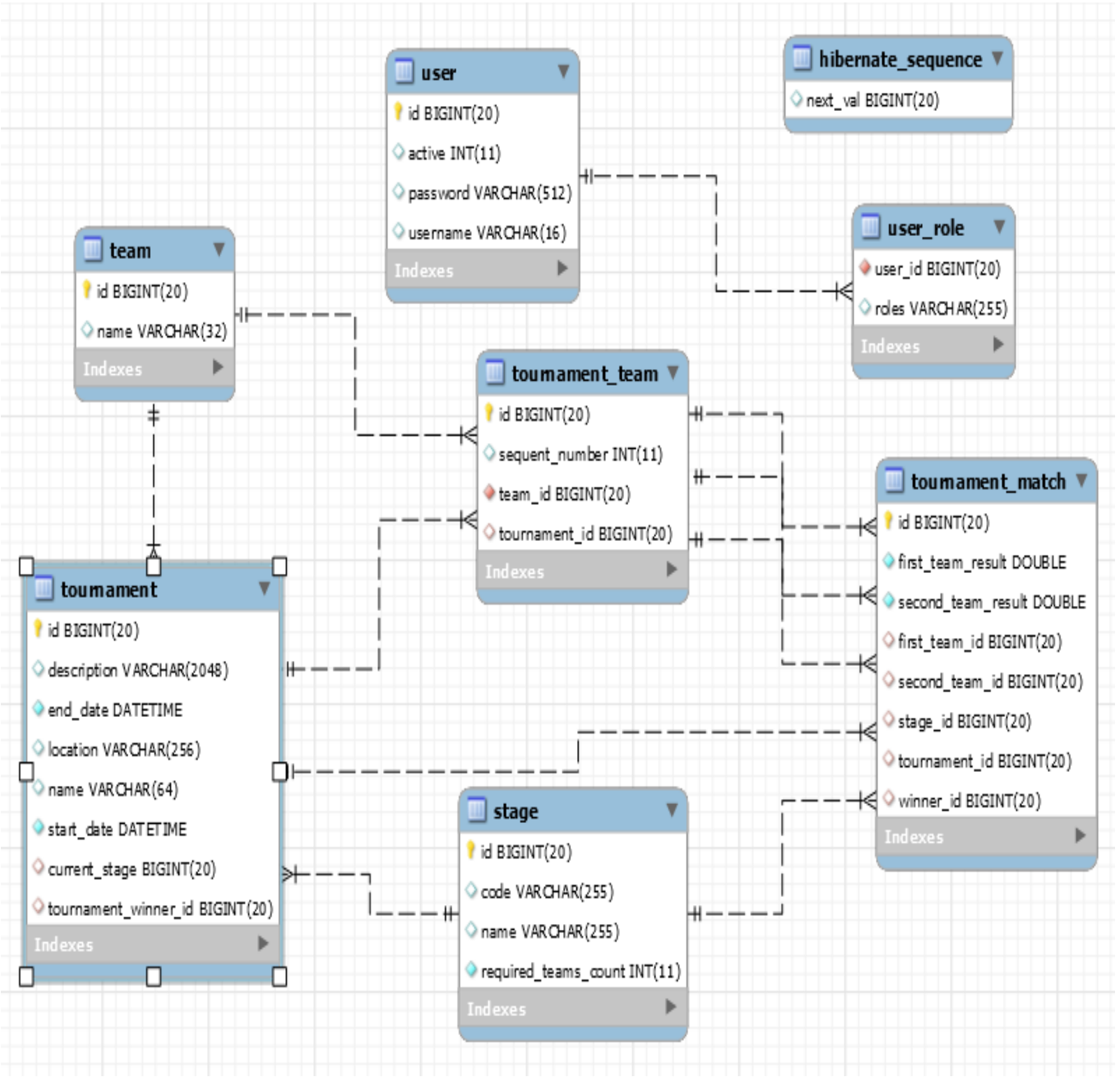

Рисунок 2.1 - Ключові поля таблиць в базі даних

2.2 Огляд таблиць бази даних

В даному розділ буде описано всі таблиці бази даних та їхнє призначення. База даних складається зі 7 таблиць (що зображені на рисунку 2.1):

User;

user role;

tournament;

team;

tournament_team;

tournament_match;

stage.

Першою є таблиця User, (рис. 2.2), що показує, який тип даних у поля та яку довжину символів воно містить.

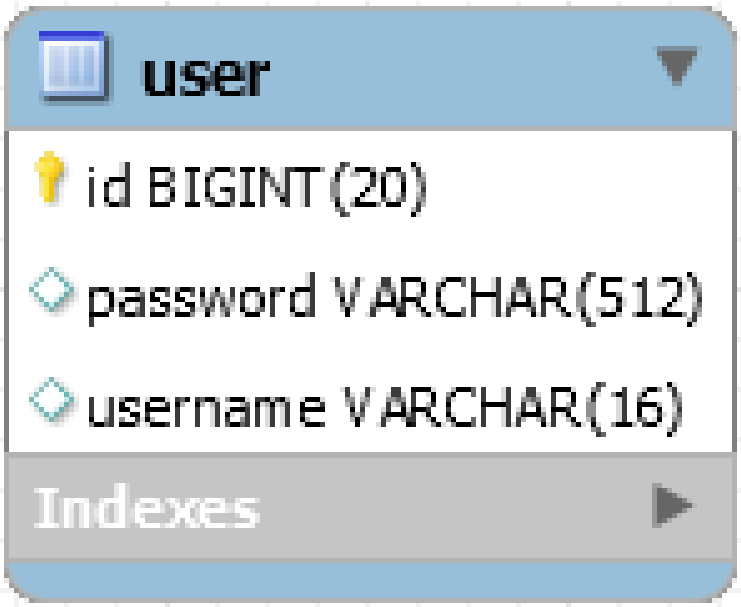

Рисунок 2.2 - Таблиця user

Таблиця на даний момент використовується тільки для входу адміністратора, оскільки крім адміністратора ніхто не може добавляти турніри в таблицю. При потребі в дану таблицю можна добавити поле email, та додати на

сайт регістрацію, але оскільки вданий момент за користувачем не закріплено жадної дії регістрація непотрібна і єдиний користувач це системний адміністратор. Паролі зберігаються в базі даних зашифованими за допомогою bcrypt шифрування.

Bcrypt - адаптивна криптографічна функція формування ключа, що використовується для безпечного зберігання паролів. Розробники: Нільс Провосі David Mazières. Функція заснована на шифрі Blowfish, вперше представлена на USENIX у 1999 році. Для захисту від атак за допомогою райдужних таблиць bcrypt використовує сіль (salt); крім того, функція є адаптивною, час її роботи легко настроюється і її можна сповільнити, щоб ускладнити атаки перебором [16]. Нижче наведений SQL скріпт для створення таблиці:

```
create table user
(
    id       bigint not null,
    active   INTEGER,
    password varchar(512),
    username varchar(16),
    primary key (id)
) engine = InnoDB;
```

Огляд полів таблиці з рисунка 2.2:

id - ідентифікатор, що використовується для вибірки полів з таблиці;

active - булеве значення, яке використовується для перевірки статусу активності користувача;

password - пароль;

username - ім’я користувача по якому буде виконуватися вхід на сайт.

Для того щоб присвоїти користувачу права адміністратора, або можна було добавити ролі користувачам такі як, наприклад, адміністратор, модератор, чи категорію користувачів з певними привілегіями, використовується таблиця user_role (рис. 2.3).

Рисунок 2.3 – Таблиця user_role

Завдяки цьому ми зможемо розподілити доступний функціонал по ролях. В подальшому є змога добавити новий функціонал до певної ролі чи видалити його. Нижче наведений SQL скріпт для створення таблиці:

```
create table user_role
(
          user_id bigint not null,
          roles   varchar(255)
     ) engine = InnoDB;
```

Огляд полів таблиці з рисунка 2.3:

user_id - ідентифікатор, що використовується для вибірки полів з таблиці user;

roles - призначення вибраному користувачу ролей таких як адміністратор або користувач.

Наступна і головна таблиця від якої відштовхується весь функціонал по побудові турнірної сітки це таблиця tournament (рис. 2.4).

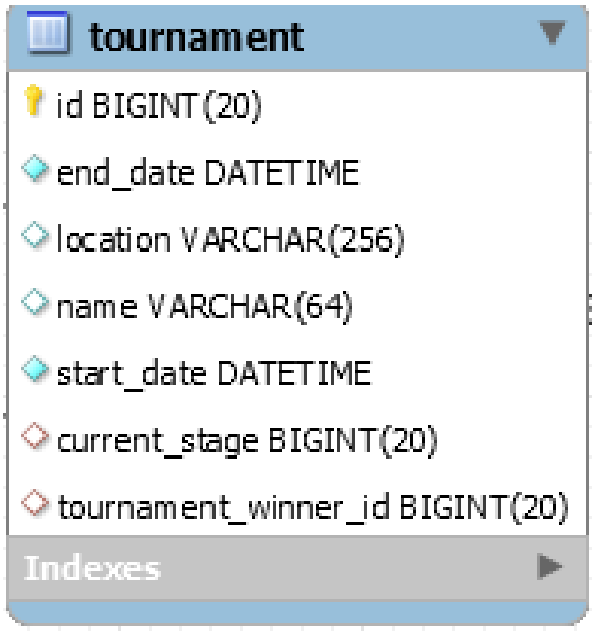

Рисунок 2.4 – Таблиця tournament

В даній таблиці ми створюємо турнірі також вводимо іншу супутню інформацію що стосується турніру таку як дата коли турнір почнеться, та коли закінчиться, місце проведення турніру. В кінці турніру в дану таблицю буде записано переможця фінального матчу турніра, що спростить вивід його даних для користувачів.

Також для спрощення написання проекту в даній таблиці будуть постійно оновлюватися дані про те на якій стадії в даний момент перебуває турнір, що використовується для того щоб будувати сітку. Нижче наведений SQL скріпт для створення таблиці:

```
create table tournament
(
    id                  bigint   not null,
    end_date            datetime not null,
    location            varchar(256),
    name                varchar(64),
    start_date          datetime not null,
    current_stage       bigint,
    tournament_winner_id bigint,
    primary key (id)
) engine = InnoDB;
```

Огляд полів таблиці з рисунка 2.4:

id - ідентифікатор, що використовується для вибірки полів з таблиці;

end_date - поле з типом даних що відповідає за дату і час, де буде додана дата та час закінчення турніру ;

location - поле для запису місця проведення турніру;

name - назва турніру;

start_date - поле з типом даних що відповідає за дату та час, де буде додана дата та час початку турніру;

current_stage - сюди записується поточна стадія турніру;

tournament_winner_id - назва команди що виграла турнір.

Для того щоб додавати команди які будуть у турнірах було створено таблицю team (рис. 2.5).

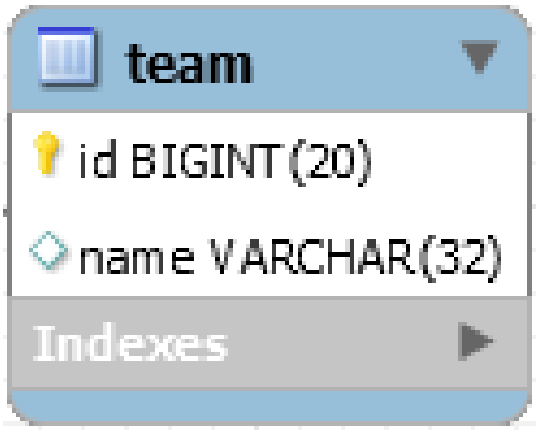

Рисунок 2.5 – Таблиця team

При додаванні команд у турнір якщо команда небула раніше зареєстрована в жодному з турнірів вона додається в таблицю. Команди в таблиці не повторюються, тобто якщо раніше команда уже брала участь у попередніх турнірах її дані заново вноситись в базу не будуть, вони будуть пере використані. Нижче наведений SQL скріпт для створення таблиці:

```
create table team
(
```

```
    id   bigint not null,
    name varchar(32),
    primary key (id)
) engine = InnoDB;
```

Огляд полів таблиці з рисунка 2.5:

id - ідентифікатор, що використовується для вибірки полів з таблиці;

name - назва команди.

Щоб до конкретного турніра добавити команди використовується таблиця tournament_team (рис. 2.6).

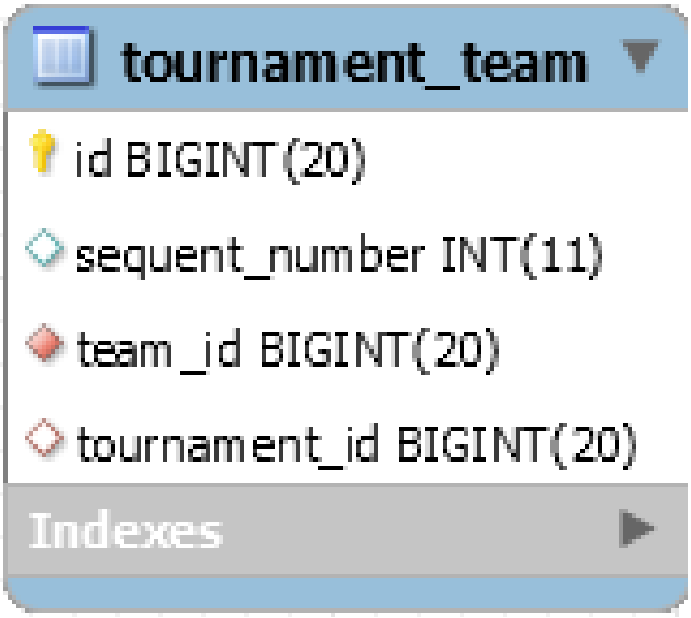

Рисунок 2.6 – Таблиця tournament_team

Після того як ми вносимо команди до турніру і вони добавилися в таблицю team ми їх перемішуємо оскільки одною з головних функцій після турнірних сіток являється система жеребкування і уже перемішаними вносимо їх до даної таблиці де їм присвоюється турнір в якому вони беруть участь та порядковий номер який являється зарандомленим числом на певну кількість команд що дозволяє виставляти команди в турнірній сітці відповідно до порядкового номера. Нижче наведений SQL скріпт для створення таблиці:

```
create table tournament_team
(
    id             bigint not null,
    sequent_number integer,
    team_id        bigint not null,
    tournament_id  bigint,
    primary key (id)
) engine = InnoDB;
```

Огляд полів таблиці з рисунка 2.6:

id - ідентифікатор, що використовується для вибірки полів з таблиці;

sequent_number - порядковий номер;

team_id - ідентифікатор, що використовується для вибірки полів з таблиці team;

tournament_id - ідентифікатор, що використовується для вибірки полів з таблиці tournament.

Для того щоб в турнірах проводити матчі постало необхідним створення окремої таблиці tournament_match (рис. 2.7), де будуть записуватися матчі між командами.

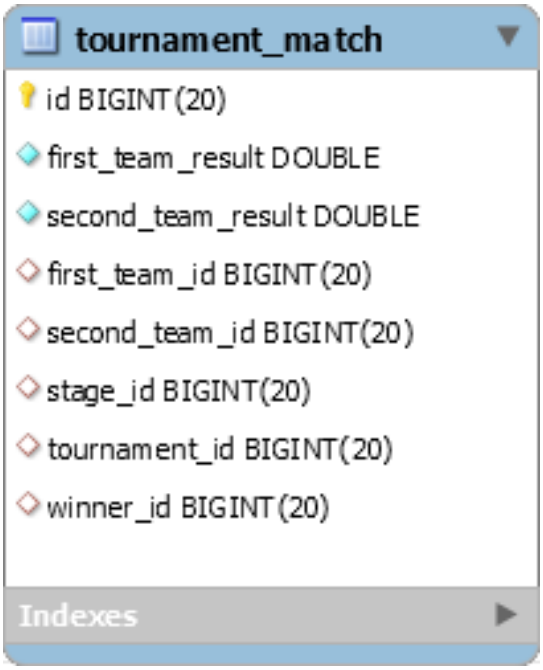

Рисунок 2.7 – Таблиця tournament_match

В даній таблиці записуються команди між якими проводився матч та результат цього матчу, відповідно до якого сервер записує переможця цього матчу. Також в даній таблиці матч прив'язуються до турніру в якому цей матч проводився, та до стадії турніру на якій цей матч проведено для того щоб була можливість сформувати турнірну сітку до вибраного турніру. Нижче наведений SQL скріпт для створення таблиці:

```
create table tournament_match
(
    id                 bigint           not null,
    first_team_result  double precision not null,
    second_team_result double precision not null,
    first_team_id      bigint,
    second_team_id     bigint,
    stage_id           bigint,
    tournament_id      bigint,
    winner_id          bigint,
    primary key (id)
) engine = InnoDB;
```

Огляд полів з рисунка 2.7:

id - ідентифікатор, що використовується для вибірки полів з таблиці;

first_team_result - результат матчу першої команди;

second_team_result - результат матчу другої команди;

first_team_id - ідентифікатор першої команди;

second_team_id - ідентифікатор другої команди;

stage_id - ідентифікатор, що використовується для вибірки полів з таблиці stage_id;

tournament_id - ідентифікатор, що використовується для вибірки полів з таблиці tournament_id;

winner_id - назва команди що виграла в матчі.

Остання таблиця stage (рис. 2.8) використовується для позначення стадій турніру.

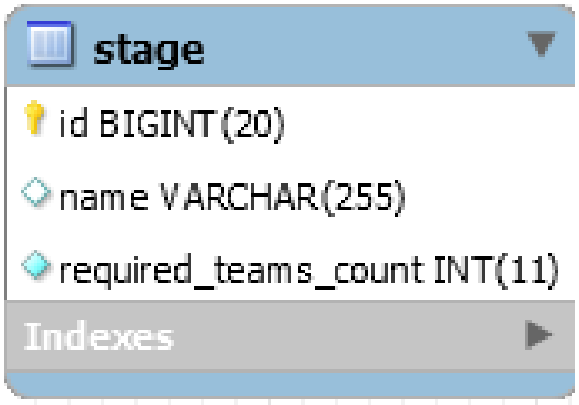

Рисунок 2.8 – Таблиця stage

Дана таблиця використовується як для виводу стадії уже безпосередньо користувачу так і для взаємодій з іншими таблицями. Також таблиця містить необхідну кількість команд для кожної з стадій турніру. Турнірна сітка розрахована на турніри від 2 до 64 команди. Якщо взяти турнір на 64 команди то він буде включати 6 стадій таких як: фінал, півфінал, чвертьфінал, 1/8 фіналу, 1/16 фіналу та 1/32 фіналу. Нижче наведений SQL скріпт для створення таблиці:

```
create table stage
(
    id                   bigint  not null,
    name                 varchar(255),
    required_teams_count integer not null,
    primary key (id)
) engine = InnoDB;
```

Огляд полів з рисунка 2.8:

id - ідентифікатор, що використовується для вибірки полів з таблиці;

name - назва стадії матчу;

required_teams_count - присвоєна до кожної стадії кількість учасників.

2.3 Компоненти та бібліотеки користувацької частини веб-додатку

Для написання користувацької частини веб-додатку було використано декілька основини бібліотек vue.js та декілька додаткових .

Одною з таких є Vue-router [17], що представявляє собою офіційну бібліотеку vue.js, основним функціоналом якої є маршрутизація користувачів по сторінках веб додатку. Без маршрутизації посилань не обходиться жоден веб додаток. Маршрутизація по веб сайту являє собою посилання яке ми бачимо після основної назви. Для прикладу адреса головної сторінки мого веб-додатку виглядатиме так https://tournament.ua/, але перейшовши для прикладу на сторінку з турнірами ми замітимо що до головного посилання добавився надпис з нашим поточним місцем перебування на сайті і тепер посилання матиме такий вигляд https://tournament.ua/tournaments. Це свідчить про те що ми вже не на головній сторінці і, відповідно, контент завантажується той, який знаходиться на даній сторінці.

Наступною бібліотекою vue.js є бібліотека Vuex [18], що являє собою контейнер для в зберігання поточного стану веб-додатку. Дана бібліотека дозволяє робити глобальні змінні та функції реактивним, та відслідковувати їхню мутацією. Що має на увазі те що кожна зміна на сайті відбувається відразу без перезагрузок сторінки.

Ще одною бібліотекою яка використовується стала Axios [19] що являє собою бібліотеку для надсилання POST, GET, DELETE, PUT запитів [20]. Тобто використовується для доступу до API сервера. Тепер розберемо кожен із вказаних запитів.

Запит POST використовується для створення ресурса. Завдяки ньому я буду реалізувати додавання турнірів, та результатів матчу турнірів у веб додатку.

Запит GET використовується для отримання даних. Тобто для завантаження будь яких уже існуючих ресурсів. Для прикладу того ж турніру та його матчів.

Запит DELETE використовується для видалення. Завдяки даному запиту адміністратор може видалити будь який турнір.

Для того щоб спростити та оптимізувати написання веб-додатку я використовую Vue компоненти [21]. Vue компонент це файл, в якому зберігається шаблон сторінки, дані та методи компоненту та CSS (SCSS) стилі. Завдяки ньому heder веб додатку не буде кожен раз завантажуватися заново. Тому, що він використовується на кожній сторінці.

Для того, щоб проводити перевірку відповідності до вимог даних які вносяться в форму використовується vuelidate що являє собою бібліотеку для простої валідації форм, з вбудованими валідаторами.

2.4 Вибір середовищ розробки програм

Для розробки Веб-додатку використовував два середовища розробки. IntelliJ IDEA для написання серверної частини та WebStorm для написання клієнтської.

Обидва середовища розробки являються програмними продуктами компанії JetBrains.

JetBrains - компанія з розробки програмного забезпечення з офісами у Празі, Санкт-Петербурзі, Бостоні, Москві та Мюнхені. Відома розробкою інтегрованого середовища розробки для Java IntelliJ IDEA. Компанія була заснована в 2000 році як приватна компанія Сергієм Дмитрієвим, Євгеном Бєляєвим і Валентином Кіпятковим [22].

Першим я розгляну середовище розробки IntelliJ IDEA інтерфейс якої зображений на рисунку 2.9.

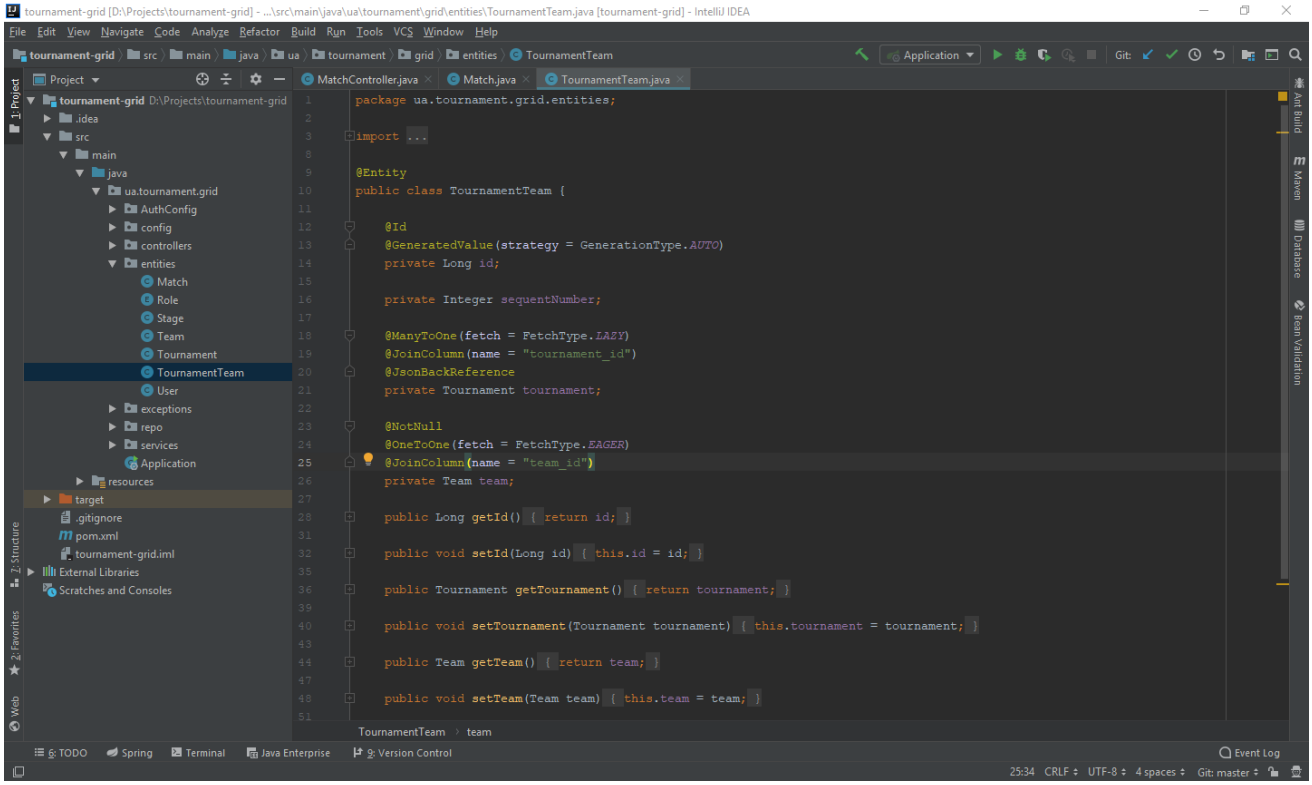

Рисунок 2.9 - Інтерфейс IntelliJ IDEA

IntelliJ IDEA - інтегроване середовище розробки програмного забезпечення для багатьох мов програмування, зокрема Java, JavaScript, Python, розроблена компанією JetBrains.

Перша версія з'явилася в січні 2001 року і швидко набула популярності як перша середа для Java з широким набором інтегрованих інструментів для рефакторінга, які дозволяли програмістам швидко реорганізовувати вихідні тексти програм. Дизайн середовища орієнтований на продуктивність роботи програмістів, дозволяючи сконцентруватися на функціональних завданнях, в той час як IntelliJ IDEA бере на себе виконання рутинних операцій.

Починаючи з шостої версії продукту IntelliJ IDEA надає інтегрований інструментарій для розробки графічного інтерфейсу користувача. Серед інших можливостей, середа добре сумісна з багатьма популярними вільними інструментами розробників, такими як CVS, Subversion, Apache Ant, Maven і JUnit. У лютому 2007 року розробники IntelliJ анонсували ранню версію плагіна для підтримки програмування на мові Ruby.

Починаючи з версії 9.0, доступна в двох редакціях: Community Edition і Ultimate Edition. Community Edition є повністю вільною версією, доступною під ліцензією Apache 2.0, в ній реалізована повна підтримка Java SE, Kotlin, Groovy, Scala, а також інтеграція з найбільш популярними системами управління версіями. В редакції Ultimate Edition, доступною під комерційною ліцензією, реалізована підтримка Java EE, UML-діаграм, підрахунок покриття коду, а також підтримка інших систем управління версіями, мов та фреймворків.

Створення нового проекту в IntelliJ IDEA. Програма пропонує широкий вибір мов програмування.

Вона підтримує такий перелік мов програмування:

Java, JavaScript, CoffeeScript, HTML / XHTML / HAML, CSS / SASS / LESS, XML / XSL / XPath, YAML, ActionScript / MXML, Python, Ruby, Haxe, Groovy, Scala, SQL, PHP, Kotlin, Clojure, Ci,C ++, Go [23].Єдиним конкурентом IntelliJ IDEA являється середовище розробки програмного Eclipse інтерфейс якої зображений на рисунку 2.10.

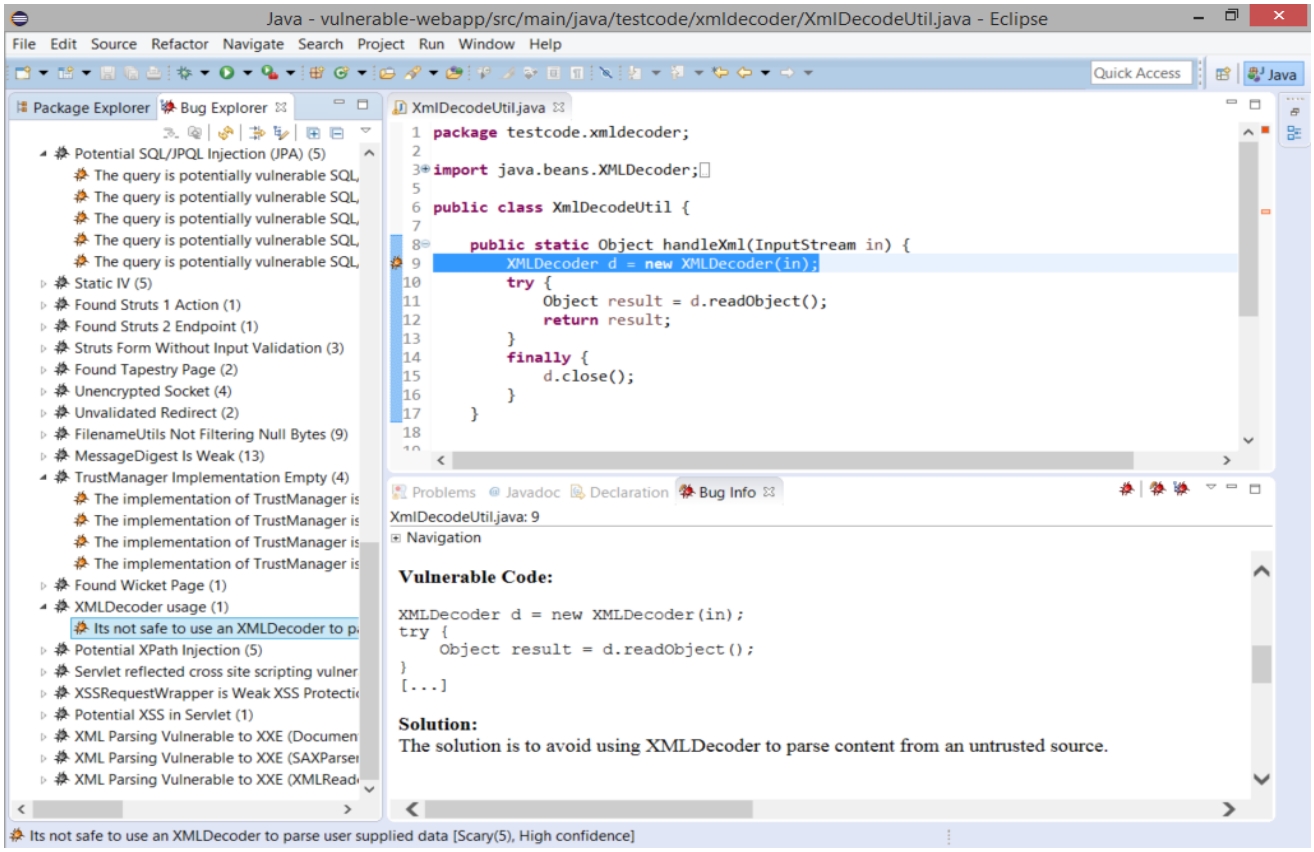

Рисунок 2.10 - Інтерфейс Eclipse

Eclipse - вільне модульне інтегроване середовище розробки програмного забезпечення. Розробляється і підтримується Eclipse Foundation і включає проекти, такі як платформа Eclipse, набір інструментів для програмістів на мові Java, системи контролю версій, конструктори GUI тощо. Написаний в основному на Java, може бути використаний для розробки застосунків на Java і, за допомогою різних плагінів, на інших мовах програмування, включаючи Ada, C, C++, COBOL, Fortran, Perl, PHP, Python, R, Ruby (включно з каркасом Ruby on Rails), Scala, Clojure та Scheme. Середовища розробки зокрема включають Eclipse ADT (Ada Development Toolkit) для Ada, Eclipse CDT для C/C++, Eclipse JDT для Java, Eclipse PDT для PHP [24].

IntelliJ IDEA являється більш досконалим продуктом ніж Eclipse і це виражається в простоті користування та більш широкому набору функціоналу. Для більшості задач вам не прийдеться інсталювати будь які плагіни, так як більшість з них працює з коробки в порівнянні з Eclipse.

Єдиним досить вагомим плюсом Eclipse являється те що він являється абсолютно безкоштовним середовищем для розробки програмного забезпечення з відкритим кодом, в порівнянні з IntelliJ IDEA ціна якого не відрізняється особою дешевизною.

Для розробки користувацької частини використовував середовище розробки.

JetBrains WebStorm - інтегроване середовище розробки на JavaScript, CSS & HTML від компанії JetBrains, розроблена на основі платформи IntelliJ IDEA.

WebStorm забезпечує автодоповнення, аналіз коду на льоту, навігацію по коду, рефакторинг, налагодження, і інтеграцію з системами управління версіями. Важливою перевагою інтегрованого середовища розробки WebStorm є робота з проектами (в тому числі, рефакторинг коду JavaScript, що знаходиться в різних файлах і папках проекту, а також вкладеного в HTML). Підтримується множинна вкладеність (коли в документ на HTML вкладений скрипт на Javascript, в який вкладено інший код HTML, всередині якого вкладено Javascript) - тобто в таких конструкціях підтримується коректний рефакторинг [25].

Інтерфейс WebStorm який зображений на рисунку 2.11. є таким же як і у IntelliJ IDEA, оскільки розробником виступає та сама компанія що і IntelliJ IDEA.

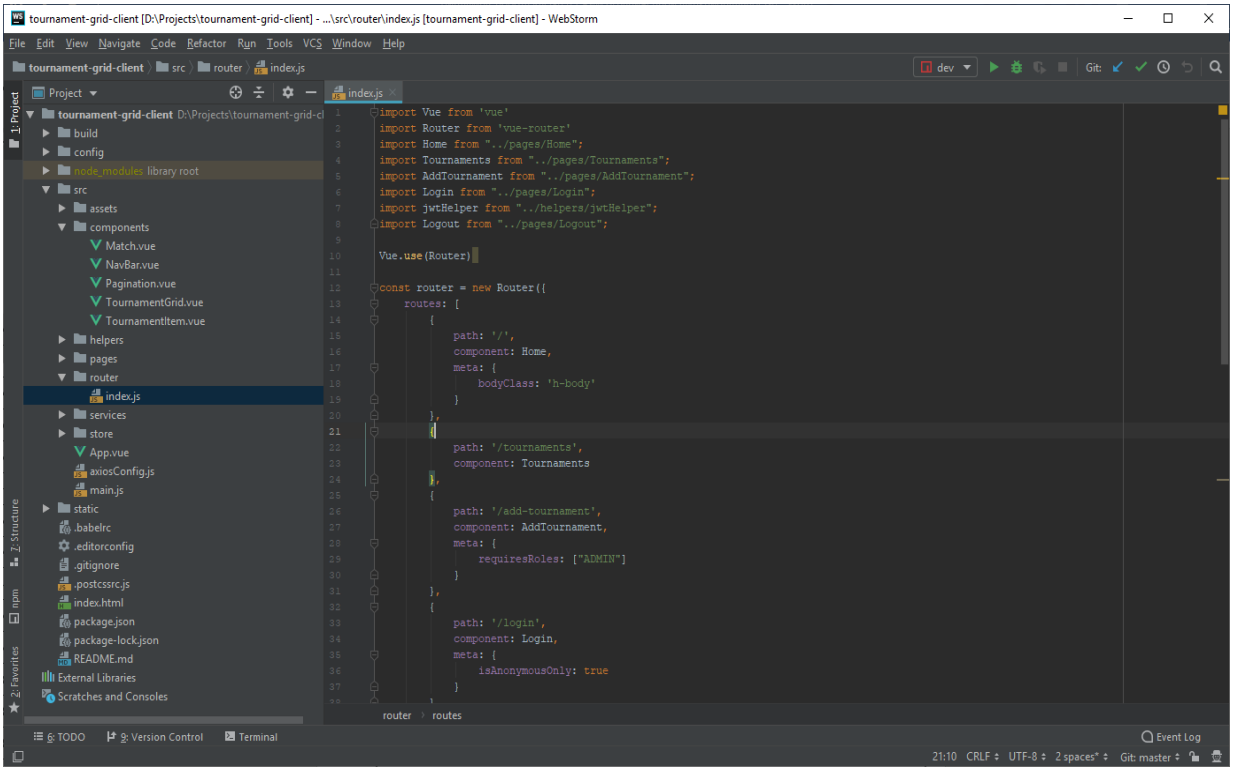

Рисунок 2.11 - Інтерфейс WebStorm

У WebStorm є безліч аналогів хоча і жоден з них не є йому альтернативою по функціоналу та зручності у використанні. Але виключно для порівняння було обрано Visual Studio Code оскільки дане середовище розробки найбільш приближено до WebStorm, та являється безкоштовним та з відкритим кодом, то для порівняння було обрано саме його.

Visual Studio Code - редактор вихідного коду, розроблений Microsoft для Windows, Linux і macOS. Позиціонується як «легкий» редактор коду для кроссплатформенной розробки веб-і хмарних додатків. Включає в себе відладчик, інструменти для роботи з Git, підсвічування синтаксису, IntelliSense і засоби для рефакторинга. Має широкі можливості для кастомізації: призначені для користувача теми, поєднання клавіш і файли конфігурації його інтерфейс зображений на рисунку 2.12 [26].

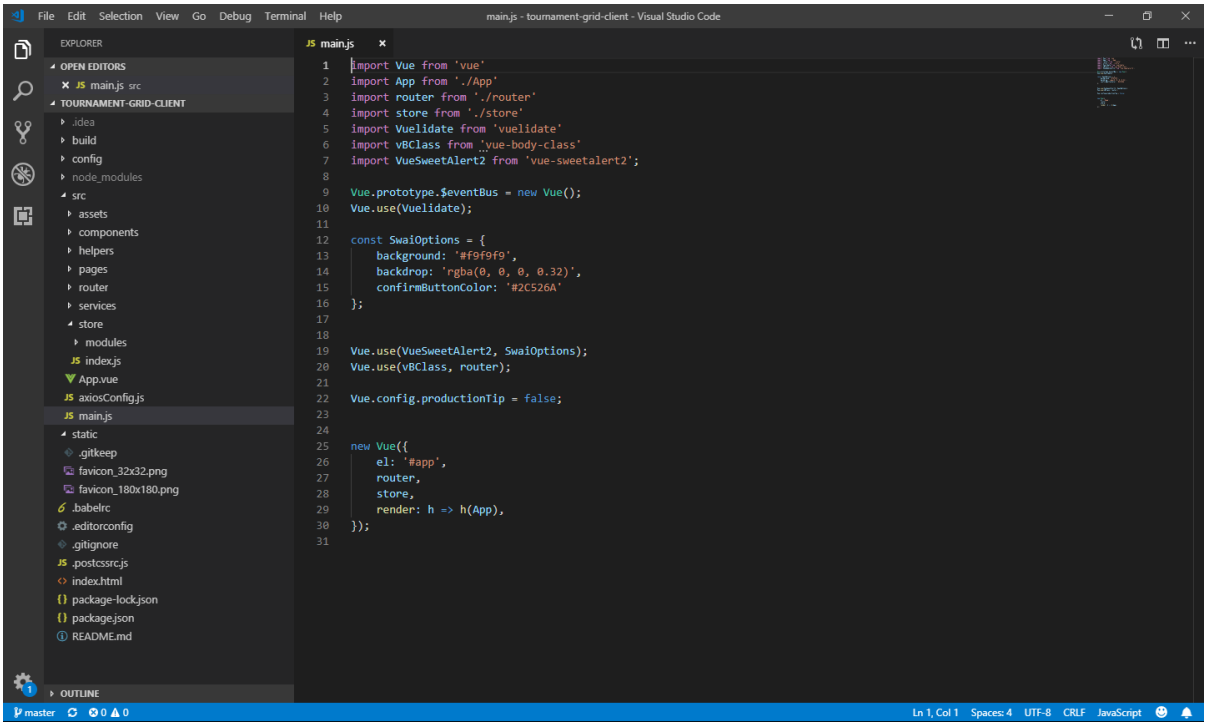

Рисунок 2.12 - Інтерфейс Visual Studio Code

Visual Studio Code являється менш функціональним середвищем для розробки програмного забезпечення єдиною перевогою над конкурентом функціонал якого являється більш розвиненим та зручним для користувача є те що Visual Studio Code являється безкоштовним середовищем для розробки.

3 РОЗРОБКА СЕРВЕРНОЇ ЧАСТИНИ, ТА ТЕСТУВАННЯ ФУНКЦІОНАЛУ

3.1 Розробка backend частини веб-додатку

В написанні веб додатку з певним набором функціоналу основну роль виконує backend. Backend - це серверна частина яка включає в себе роботу сервера з базою даних, обробку запитів та інший набір функцій які в ньому прописані. Тобто без серверної частини frontend який виконує функцію інтерфейсу для зв'язку користувача з сервером буде просто інтерфейсом який не матиме жодного функціоналу до того моменту поки не буде працювати в парі з сервером на якому цей функціонал реалізований.

Моє завдання полягає в написанні серверної частини для створення турнірів з жеребкуванням та побудові турнірних сіток тому в даному підрозділі будуть описані алгоритми для створення та керування турнірами та турнірними сітками.

Для створення турнірів та запису команд в нього я написав метод createTournament, що приймає екземпляр класу Tournament та, який приходить з клієнту, а також оголосив змінну, що приймає в себе значення кількості команд які будуть брати участь у турнірі. Код для реалізації описаних дій зображено нижче:

```
PUBLIC VOID CREATETOURNAMENT(TOURNAMENT TOURNAMENT) THROWS NOTFOUNDEXCEPTION {
    INT TEAMSCOUNT = TOURNAMENT.GETTOURNAMENTTEAMS().SIZE();
```

Оскільки згідно до вимог завдання турнірна сітка повинна бути розрахована на турніри до 64. Для реалізації було обрано олімпійську систему з вибуванням після першого програшу. Після того як всі команди проведуть матч половина команд, тобто всі команди які програють вилітають. Таким чином після кожного етапу команд повинно ставати вдвічі менше. Тому я вирішив що в

турнір повинен містити кільськість команд яка буде відповідати: 64, 32, 16, 8, 4 та 2. Для цього робиться наступна перевірка, що перевіряє кількість команд на відповідність що написана в коді:

```
if (teamsCount == 2 || teamsCount == 4 || teamsCount == 8 || teamsCount ==
16 || teamsCount == 32 || teamsCount == 64)
```

Та в разі невідповідності до заданих параметрів сервер поверне виключення виключення яке буде вказувати на цю невідповідність та покаже скільки команд ми хотіли добавити.

```
ELSE THROW NEW NOTFOUNDEXCEPTION("COUNT OF TEAMS IN TOURNAMENT MUST BE 2,
4, 8, 16, 32 OR 64. IN YOUR TOURNAMENT " + TEAMSCOUNT + " TEAMS");
```

Коли ми добавляємо турнір ми також додаємо інші суміжні дані про нього на деякі з них також потрібна перевірка. У мене такими даними являється дата та час початку і кінця турніру. В наступному фрагменті коду перевіряться чи дата початку турніру встановлена перед датою закінчення турніру. В іншому випадку сервер поверне виключення в якому буде вказано що дата початку турніру повинна бути раніше дати кінця.

```
IF (TOURNAMENT.GETSTARTDATE().BEFORE(TOURNAMENT.GETENDDATE())){
} ELSE THROW NEW ILLEGALARGUMENTEXCEPTION("TOURNAMENT START DATE MUST BE
BEFORE END DATE");
```

Для збереження підтверджених команд турніру проводиться ініціалізація структури типу set. А оскільки крім створення турнірів, та побудови турнірної сітки основним функціоналом є жеребкування то для його реалізації створюємо перемішаний список який відповідає розмір якого відповідає кількості команд турніру, елементи що являються порядковими номерами які в подальшому будуть закріплені за командою.

```
SET<TOURNAMENTTEAM> VERIFIEDTOURNAMENTTEAMS = NEW HASHSET<>();
LIST<INTEGER> SEQUENTNUMBERS = GETSHUFFLEDNUMBERSARRAY(TEAMSCOUNT);
```

Наступною дією за допомогою оператора цикла for перебираються дані отримані з клієнта для перевірки команд у турнірі. Оскільки у нас уже міг бути турнір, де могли брати участь ті самі команди які хочуть прийняти участь в поточному, і оскільки нам не потрібно щоб назви команд не повторювалися за допомогою класу Optional ми звертаємось до бази і пробуємо знайти команди які уже є в базі даних для того щоб получити імена команд що уже записані, але якщо такі не будуть знайдені за допомогою класу Optional ми зможемо перевірити чи присутня команда в базі даних не отримавши NullPointerException в разі її відсутності. Наступним кроком ми присвоюємо команди що беруть участь в турнірі до цього турніру та присвоюємо їм рандомні порядкові номера з раніше згенерованого списку. Після чого інкементуємо індекс:

```
for (TournamentTeam tournamentTeam : tournament.getTournamentTeams()) {
    Optional<Team> opTeam =
teamRepo.findByName(tournamentTeam.getTeam().getName());
    tournamentTeam.setTournament(tournament);
    tournamentTeam.setSequentNumber(sequentNumbers.get(index));
    index++;
```

Для того щоб запобігти записанню двох однакових команд за допомогою умовного оператора if я створюю екземпляр класу Optional за допомогою якого виконується вибірка команди з бази даних по імені команди, і якщо вона існує, то дані не додаються а беруться з бази даних, та присвоюються в клас tournamentTeam:

```
if (opTeam.isPresent()) {
  tournamentTeam.setTeam(opTeam.get());
```

Наступна перевірка запобігає записанню двох однакових команд яких ще немає в базі даних в структуру сет verifiedTournamentTeams в якій зберігається

список підтверджених команд турніру. Якщо команда проходить дані перевірки то вона буде записана в verifiedTournamentTeams.У випадку коли команда не пройде дану перевірки сервер відправить виключення що буде вказувати на те що ми не можемо добавити дві одинакові команди в турнір:

```
if (!isTeamAlreadyAdded(verifiedTournamentTeams, opTeam.get())) {
    verifiedTournamentTeams.add(tournamentTeam);
} else throw new NotFoundException("Cannot add 2 same teams");
```

В іншому випадку буде використано умовний оператор else, в якому, якщо команди немає в базі даних, її буде туди записано, а після збереження даних об'єкту newTeam у поле id буде автоматично присвоєно id, яке створено у БД. Також була написана ще одна запасна перевірка яка перевірятиме чи команда не була додана раніше. Вірогідність даного сценарію дуже мала так як команда яку додали пройде по попередній умові. Після чого для запобігання перезапису існуючих даних команди в турнірі було вказано що id команди турніру являється null. Після чого у команди турніру зберігається раніше збережена у базі даних команда. Так як у команди турніру присутній і порядковий номер і команда - її можна додати у список підтверджених команд. Якщо команда не пройде дану перевірки сервер відправить виключення, що буде вказувати на те, що ми не можемо добавити дві одинакові команди в турнір:

```
    } else {
        Team newTeam = tournamentTeam.getTeam();
        teamRepo.save(newTeam);
        if (!isTeamAlreadyAdded(verifiedTournamentTeams,
tournamentTeam.getTeam())) {
            tournamentTeam.setId(null);
            tournamentTeam.setTeam(newTeam);
            verifiedTournamentTeams.add(tournamentTeam);
        } else throw new NotFoundException("Cannot add 2 same teams");
    }
}
```

Наступний метод getShuffledNumbersArray використовується для генерації перемішаного списку порядкових номерів команд. В даному методі створюється список з циклом який додає у список цілі числа де кількість проходів це кількість команд у турнірі. Так як початок відліку починається з нуля а не з одиниці було вказано, що числа в список добавляються з індексом +1. Після якого додавання числа у список починається з 1. Наступним кроком за допомогою статичного класу Collections з допомогою методу shuffle проходить перемішування елементів вище створеного списку. Також вказано що даний метод повертає list що і є нашим згенерованим та перемішаним списком цілих чисел що починається з одиниці:

```
private List<Integer> getShuffledNumbersArray(int maxNumber) {
    ArrayList<Integer> list = new ArrayList<>();
    for (int i = 0; i < maxNumber; i++) {
        list.add(i+1);
    }
    Collections.shuffle(list);
    return list;
}
```

Для того щоб можна було запросити всі турніри на сторінку створено метод getAllTournaments який повертатиме всі турніри які уже присутні в базі даних. Для зручності на одну сторінку буде виводитися 16 турнірів з можливістю переключатися між сторінками, які будуть посортовані по id турнірів по спаданню. Тобто від турніру з найбільшим id до турніру з найменшим, що на практиці означатиме що на сторінці турніри будуть посортовані від найновіших до найстаріших:

```
public Page<Tournament> getAllTournaments(int page) {
    return tournamentRepo.findAll(PageRequest.of(page, 16,
Sort.by("id").descending()));
```

Для того щоб можна було отримати конкретний турнір написано метод getTournament який по id повертає турнір. Та в разі якщо такий не було знайдено, сервер відправить виключення що буде вказувати на те що він не знайшов даного турніру:

```
public Tournament getTournament(Long id) throws NotFoundException {
    return tournamentRepo.findFirstById(id).orElseThrow(() -> new
NotFoundException("Cannot find tournament"));
}
```

Окрім додавання турнірів та їх заповнення, на адміністратора також покладено функціонал по видаленню турнірів що в разі необхідності дозволить йому видалити його з бази даних. Тому було добавлено метод deleteTournament який перевіряє чи існує об’єкт який ми шукаємо, та якщо такий буде знайдено видаляє його. У разі, якщо об’єкта який ми шукали не знайдено, сервер відправить виключення що буде вказувати на те що він не знайшов турніру на який цей запит відправляли:

```
public void deleteTournament(Long tournamentId) throws NotFoundException {
    if (tournamentRepo.existsById(tournamentId)) {
        tournamentRepo.deleteById(tournamentId);
    } else throw new NotFoundException("Cannot find tournament");
}
```

Для перевірок команд на їхнє повторення в інших списках було написано метод isTeamAlreadyAdded із типом boolean що робить перевірку наявності команди яка передається як другий параметр, на наявність у списку який передається першим параметром. Та у випадку якщо команда присутня в оголошеному списку то метод повертатиме true, якщо команда не повторюється в оголошених списках то повертатиметься false.

```
    private boolean isTeamAlreadyAdded(Set<TournamentTeam> teams, Team team) {
        for (TournamentTeam tournamentTeam: teams) {
            if (tournamentTeam.getTeam().equals(team)) return true;
        }
        return false;
    }
```

Для турнірної сітки окрім назв турнірів, інформації про них та команди які в них беруть участь необхідні матчі за допомогою результатів яких ця турнірна сітка і буде будуватися.

Щоб реалізувати матчі для турнірів я створив метод createMatchesForTournament що використовується для створення матчів до турніру, та робиться тільки один раз при створенні турніру. за допомогою об'єкту класу Optional ми звертаємся до бази даних і пробуємо знайти турнір по id, але якщо такий турнір не буде знайдено за допомогою об'єкту класу Optional ми зможемо перевірити чи присутній турнір в базі даних не отримавши NullPointerException в разі йог відсутності.

```
public void createMatchesForTournament(Long tournamentId) {
    Optional<Tournament> opTournament = tournamentRepo.findById(tournamentId);
```

Після того як було отримано id турніру і за допомогою умовного оператора if перевірено чи такий турнір існує йому присвоюється посилання на об'єкт з Optional у екземпляр класу Tournament після чого ініціалізує порожній список для матчів. Для того щоб добавити до матчів рандомно перемішані команди створюється список який буде містити команди з рандомно присвоєними порядковими номерами що посортовані за зростанням порядкового номера

```
if (opTournament.isPresent()) {
        Tournament tournament = opTournament.get();
        List<Match> matches = new ArrayList<>();
        List<TournamentTeam> tournamentTeams =
```

```
tournamentTeamRepo.findByTournamentIdOrderBySequentNumberAsc(tournamentId);
```

Якщо турнір проходить умови оператора if виконується цикл for що робить перебір команд, та виконується з кроком 2 команди за турнір. Після чого у список матчів додається матч в який передається турнір, поточний етап турніру, а також перша команда та друга команда. А оскільки було використано дві команди з поточним, та наступним індексом була використана додаткова інкрементація індекса. Після проходження циклу, та генерації списку матчів в ньому цей список зберігається в базі даних за допомогою репозиторія matchRepo.

```
        for (int i = 0; i < tournamentTeams.size(); i++) {
            matches.add(new Match(tournament, tournament.getCurrentStage(),
tournamentTeams.get(i), tournamentTeams.get(i+1)));
            i++;
        }
        matchRepo.saveAll(matches);
    }
```

Моя турнірна сітка повинна бути розрахованою на 64 команди. І якщо в випадку якщо команди є тільки дві і переможець визначається одразу, то для того щоб з 64 команд залишився тільки переможець необхідно 6 стадій тому необхідно реалізувати механізм який дозволяє в залежності від кількості учасників визначати, та створювати етапи турніру. Задля цього я написав метод createNextStage що використовується для створення матчів для наступної стадії та виконується кожного разу після вказання результату всіх матчів попередньої стадії. Після чого за допомогою об’єкту класу Optional ми звертаємся до бази даних і пробуємо знайти турнір по id, але якщо такий турнір не буде знайдено за допомогою об’єкту класу Optional ми зможемо перевірити чи присутній турнір в базі даних не отримавши NullPointerException в разі йог відсутності:

```
public void createNextStage(Long tournamentId) throws NotFoundException {
    Optional<Tournament> opTournament = tournamentRepo.findById(tournamentId);
```

Після того як турнір було знайдено за допомогою умовного оператора if проводиться перевірка чи дійсно такий турнір існує, і чи в турнірі ще немає переможця. Якщо в турніру є переможець турнір вважається завершеним і нові стадії не генеруються Для початку виконуються присвоєння об'єкту отриманого з бази даних у екземпляр класу Tournament, та запис поточної стадії у екземпляр класу. Для отримання всіх переможців поточної стадії по id турніру, та по поточній стадії створено список в якому переможців стадії буде записано. Якщо у турніру уже є переможець сервер відправить виключення що скаже що у турніру уже є переможець.

```
if (opTournament.isPresent() && opTournament.get().getTournamentWinner() == null)
{
        Tournament tournament = opTournament.get();
        Stage currentStage = tournament.getCurrentStage();
        List<TournamentTeam> stageWinners =
matchRepo.findStageWinners(tournament, currentStage);
```

За допомогою наступного умовного оператора if проводиться перевірка чи переможців поточної стадії достатньо для початку наступної стадії турніру. Дана перевірка проходить таким чином що у кожної наступної стадії в два рази менше команд оскільки половина команд після кожної стадії вибуває, і полягає в тому що якщо кількість команд переможців в два рази менша ніж розмір поточної стадії, то буде виконуватися тіло умови. Встановлення наступної стадії турнір виконується за допомогою кількості переможців в поточній, тому що кількість переможців поточної стадії являється кількістю учасників наступної стадії яку і потрібно згенерувати. А так як у кожної стаді є зазначена кількість команд, то нам не складе проблем цю стадію визначити, але якщо не буде стадій з такою кількістю учасників встановлюється значення null

```
if (stageWinners.size() == currentStage.getRequiredTeamsCount() / 2){
      tournament.setCurrentStage(stageRepo.findByRequiredTeamsCount(currentStage.
      getRequiredTeamsCount() / 2).orElse(null));
```

Наступна умова виконується якщо стадія встановилась, і нова стадія не є тою, на якій зараз перебуває турнір. Після чого створюється пустий список для матчів наступної стадії після якого виконується цикл for в якому робиться прохід по всім переможцям попередньої стадії, та створення матчів для них. Після чого з допомогою репозиторія matchRepo в базу даних зберігається список матчів, та за допомогою репозиторія tournamentRepo виконується оновлення всіх змін які було внесено в турнір. Вразі якщо запит не пройде умовний оператор if сервер відправить виключення що вказуватиме на те що стадія неправильна.

```
if (tournament.getCurrentStage() != null && tournament.getCurrentStage() !=
currentStage) {
    List<Match> nextStageMatches = new ArrayList<>();
    for (int i = 0; i < stageWinners.size(); i++) {
        nextStageMatches.add(new Match(tournament,
tournament.getCurrentStage(), stageWinners.get(i), stageWinners.get(i+1)));
        i++;
    }
    matchRepo.saveAll(nextStageMatches);
    tournamentRepo.save(tournament);
} else throw new NotFoundException("Wrong stage");
```

Для того щоб створювати наступні етапи турніру необхідно що щоб на кожній стадії по результатам матча визначався переможець. Для реалізації даного функціоналу я створив метод setMatchWinner в якому даний функціонал і буде написаний. В умові даного методу передається id матчу, та порівнюються результати першої та другої команди. За допомогою об’єкту класу Optional ми звертаємся до бази даних і пробуємо знайти матч по id, за допомогою класу Optional ми зможемо перевірити це не отримавши NullPointerException в разі йог відсутності.

```
public void setMatchWinner(Long matchId, double firstTeamResult, double
secondTeamResult) throws NotFoundException {
    Optional<Match> opMatch = matchRepo.findById(matchId);
```

Після того як матч було отримано з бази даних він перевіряється по декількох умовах. Перша умова робить перевірку чи такий матч існує, та в негативному випадку сервер поверне виключення що вкаже на те він не може знайти матч по даному id. Друга умова проведе перевірку чи у вибраному з бази даних матчі ще немає переможця, і якщо він уже є сервер поверне виключення що вказуватиме на це. Наступна умова перевірятиме правильність написання результатів на відсутність рахунків значення яких являються однаковим, або рахунок яких є меншим нуля. Якщо результат не пройде по даному критерію сервер відправить виключення що вкаже на це. Якщо Матч пройде по всім вище вказаних критеріях то ми порівняємо результати першої команди з другою. Де якщо результати першої команди будуть більшими ніж у другої то вона назначається переможцем, якщо другої то відповідно другу команду записується як переможця.

```
if (opMatch.isPresent()) {
    if (opMatch.get().getWinner() == null) {
        Match match = opMatch.get();
        if (firstTeamResult != secondTeamResult && firstTeamResult > -1 &&
secondTeamResult > -1) {
            if (firstTeamResult > secondTeamResult) {
                match.setWinner(match.getFirstTeam());
            } else match.setWinner(match.getSecondTeam());
```

Після того як переможця матчу матчу отримано результати матчів першої, та другої команди записуються, та за допомогою репозиторія matchRepo зберігаються в базу даних.

```
match.setFirstTeamResult(firstTeamResult);
match.setSecondTeamResult(secondTeamResult);
matchRepo.save(match);
```

Також для для зручності користувача, я вважаю що щоб у нього не було обов’язкової потреби переглядати турнірну сітку заради того щоб дізнатися

переможця турніру необхідно реалізувати функціонал за допомогою якого будуть визначатися переможці. Для цього в першу чергу за допомогою умовного оператора if зробилась перевірка чи у турніру даний матч являється останнім, та у ньому беруть участь дві команди. Якщо матч останній - потрібно зберегти переможця турніру, турнір вибирається з бази даних по id. Наступна умова перевіряє чи турнір матча на який ми посилаємось існує. Та якщо так присвоює йому посилання на турнір в екземплярі класу. Після чого по результатах останнього матчу отримується переможець, та з допомогою репозиторія tournamentRepo дані турніру в бд оновлюються. У випадку, якщо матч не останній, в турнірі створюється новий етап турніру.

```
if (match.getStage().getRequiredTeamsCount() == 2) {
    Optional<Tournament> opTournament =
    if (opTournament.isPresent()) {
        Tournament tournament = opTournament.get();
        tournament.setTournamentWinner(match.getWinner().getTeam());
        tournamentRepo.save(tournament);
    }
} else {
    createNextStage(match.getTournament().getId());
}
```

3.2 Тестування функціоналу

Хоча метою мого завдання є написання серверної частини для побудови турнірної сітки з жеребкуванням, я вирішив, що для тестування, та повноцінної демонстрації необхідний інтерфейс користувача в якому буде можливість застосувати весь функціонал який реалізований на сервері.

Для звичайного користувача на головній сторінці сайту, що зображена на рисунку 3.1 описано, що собою являє мій проект по побудові турнірних сіток, та показані приклади того як турнірна як турнірна сітка виглядає. Користувач може перейти з головної сторінки на сторінку з турнірами з допомогою посилання

"Турніри" на навігаційній панелі або кнопок що розташовані в описі. Також в верхньому правому кутку розміщена іконка для входу адміністратора.

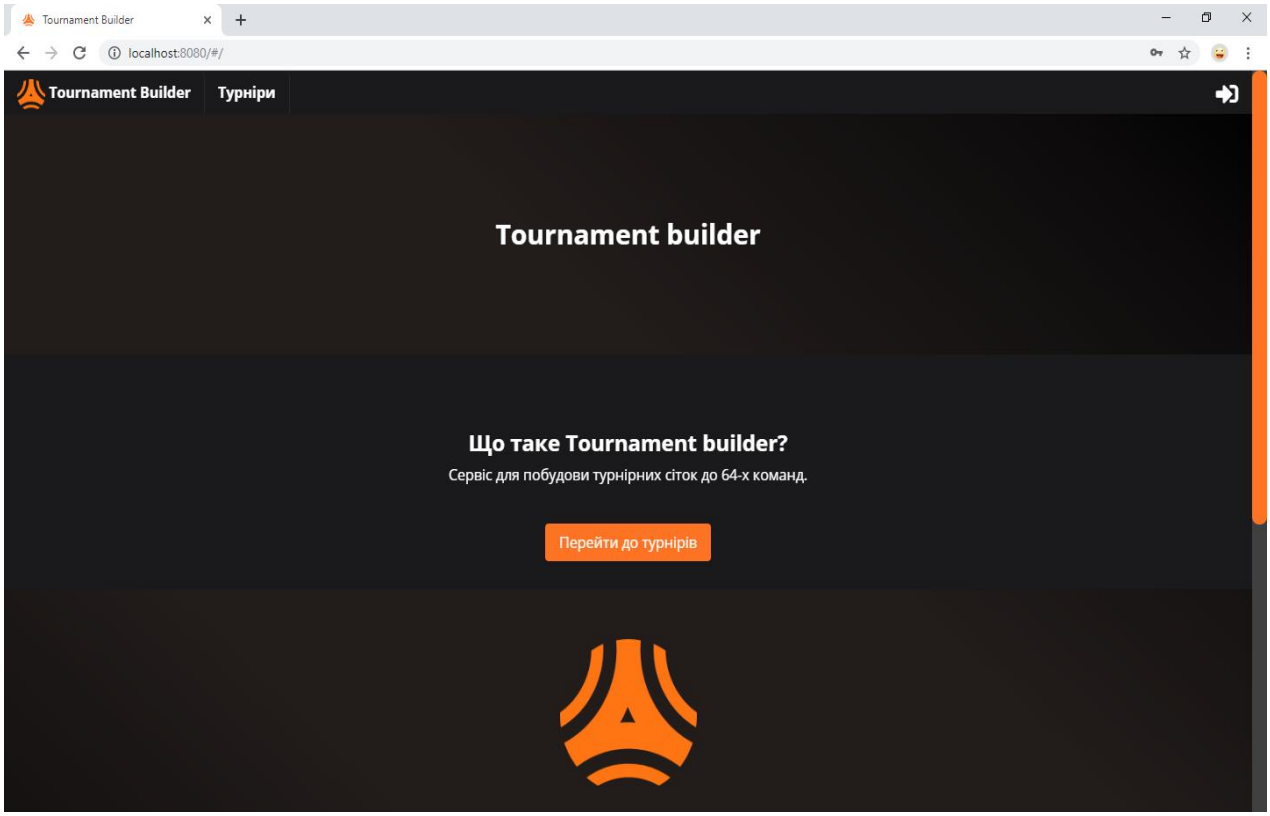

Рисунок 3.1 - Головна сторінка

Для того щоб проводити адмініструванням сайтом на ньому розміщено вхід для адміністратора, для якого був реалізований функціонал по додаванню турнірів із даними про них, та командами які беруть в них участь, додаванні результатів матчів що проводяться в турнірі та видаленні турнірів. Так як в даний момент всі функції лежать на адміністраторі, і користувачі не роблять ніяких дій чи змін на сайті, і можуть тільки переглядати інформацію, то регістрація відсутня і користувач може залогінитись на сайті тільки якщо являється адміністратором, та ввівши вірний логін і пароль в форму для входу, що зображена на рисунку 3.2 перехід на яку здійснюється за допомогою іконки для входу про яку було написано раніше. Паролі записуються в базу даних, та перевіряються в зашифрованому вигляді.

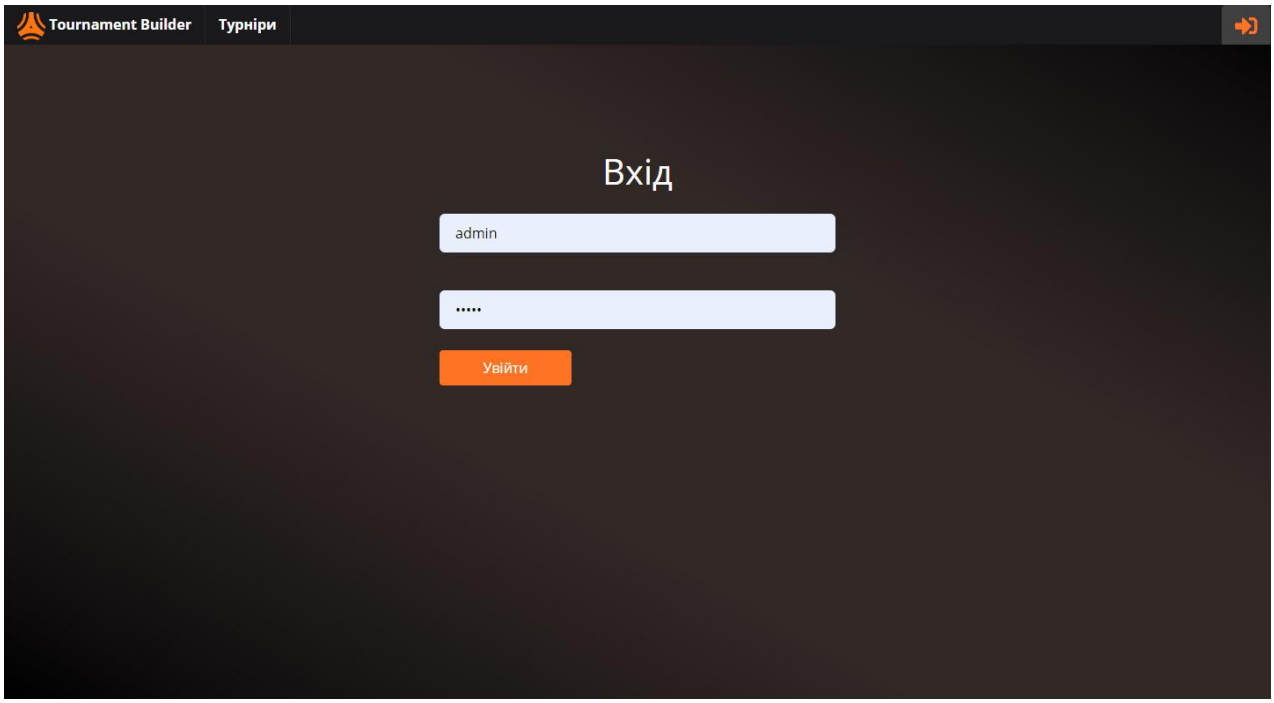

Рисунок 3.2 - Сторінка входу

Перед тим як відправляти запит серверу з допомогою js провіряється чи є в полях три, або більше символів, і якщо в полі login або password їх буде, то кнопка "увійти" буде неклікабельною, та сірою до того моменту, поки поля не пройдуть по даному валідаторі. Якщо буде введено та відправлено невірний логін або пароль, буде отримано повідомлення, що зображено на рисунку 3.3.

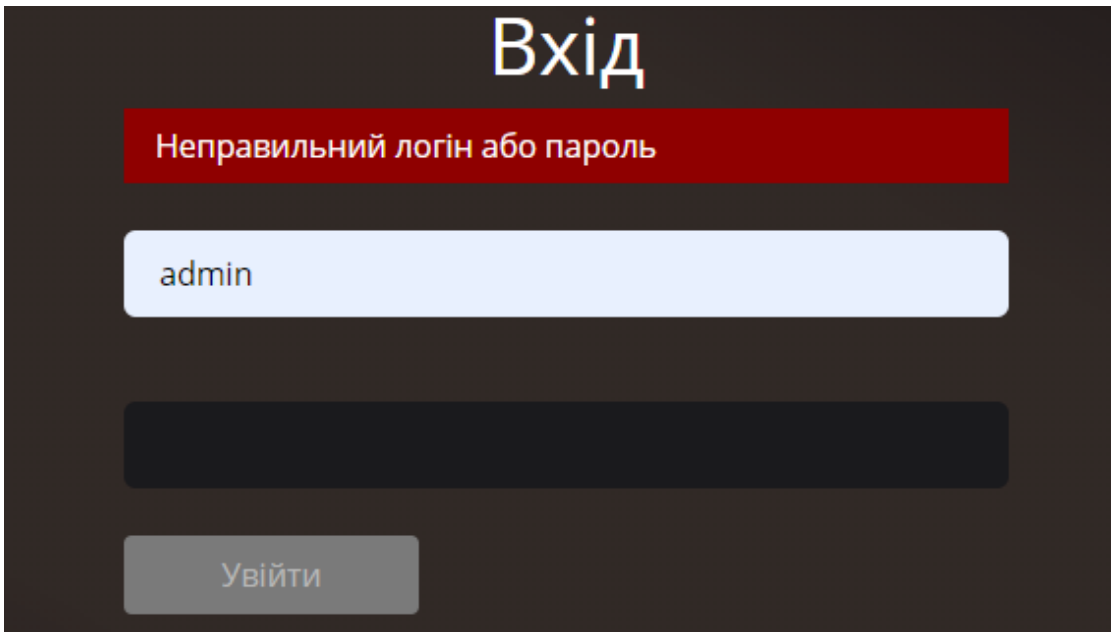

Рисунок 3.3 - Повідомлення про невірно введені дані

Після того як адміністратор здійснить вхід правильно ввівши login та password, для нього появиться додаткова вкладка "добавити турнір" в навігаційній панелі як зображено на рисунку 3.4, перейшовши по якій адміністратор зможе добавити турнір, заповнити дані про нього, та добавити команди які будуть брати у цьому турнірі участь . Також іконка для входу заміниться надписом вихід.

Рисунок 3.4 - Навігаційна панель після входу

Для того щоб оптимізувати відображення на мобільних телефонах при зменшенні ширини надпис додати турнір та вихід заміняються іконками як зображено на рисунку 3.5, так як дані надпису уже не поміщаються на екрані середньостатистичного мобільного телефона.

Рисунок 3.5 - Навігаційна панель на мобільних телефонах

Коли адміністратор клікне на навігаційній панелі по "додати турнір" він перейде на сторінку додати турнір, що зображена на рисунку 3.6 На даній сторінці адміністратор побачить поля які він повинен добавити дані для створення турніру.

Для того щоб користувач зміг дізнатися де буде відбуватися турнір добавлено поле з місцем проведення турніру. А також добавлена дата та час початку та закінчення турніру, щоб кожен відвідувач сайту зміг відвідати турнір якщо така можливість є. Або орієнтуючись на дату та час завершення, користувач знав коли появиться результат останнього матчу що відповідно буде

являтися переможцем турніру. Оскільки серверна частина розрахована тільки на турніри з фіксованою кількістю учасників, то необхідно цю кількість учасників вибрати.

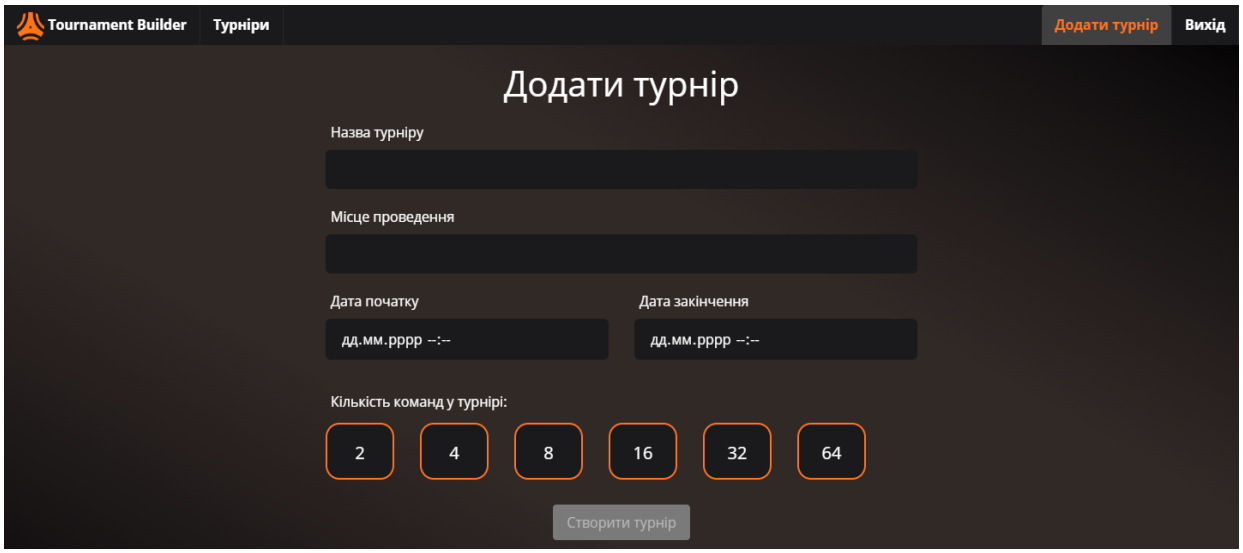

Рисунок 3.6 - Сторінка додавання турнірів

Після того як всі дані введено, та вибрано кількість учасників, в залежності від того яка кількість учасників обрана появляться поля для їх вводу як на рисунку 3.7.

Рисунок 3.7 - Додавання команд

Після того як всі команди введено в поля кнопка "створити турнір" стане активною за умови якщо в всіх полях буде не менше ніж три символи. У випадку з датою та часом, перевірка проходить уже на сервері. Для того, щоб дата та час кінця турніру не були раніше від початку, оскільки це неможливо в принципі, ми отримаємо повідомлення з помилкою, що турнір не вдалося створити, як зображено на рисунку 3.8.

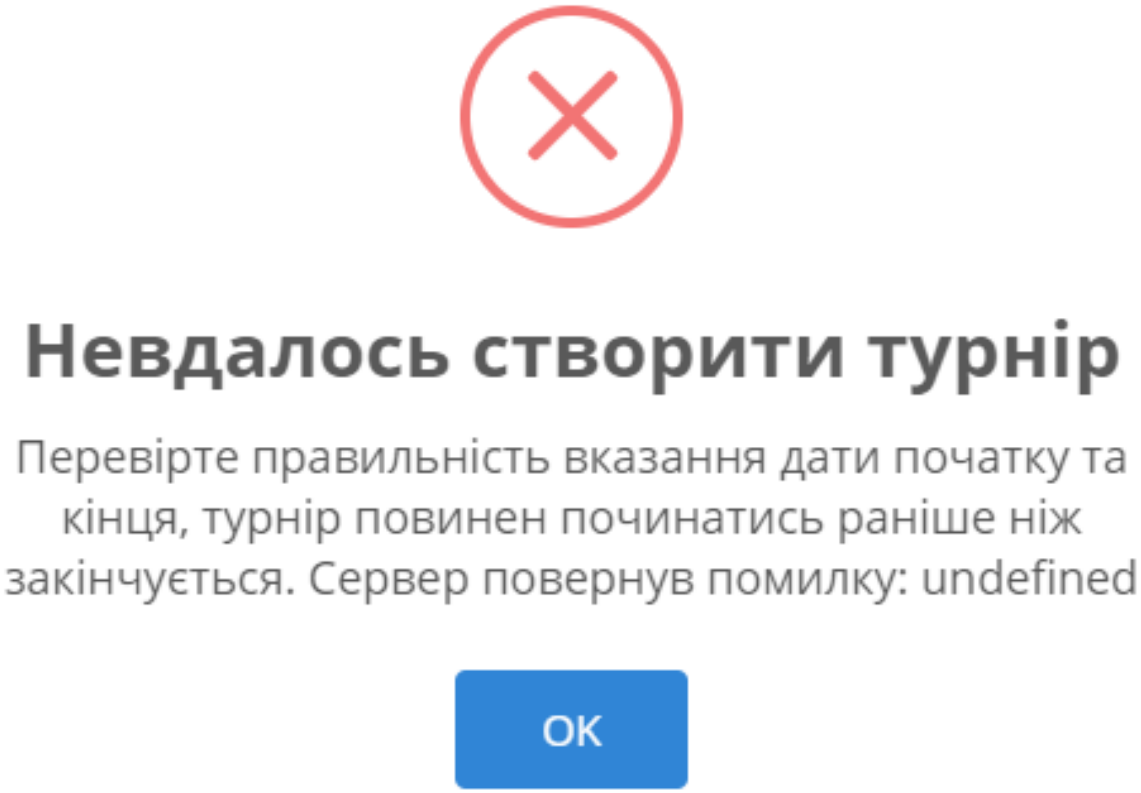

Рисунок 3.8 - Повідомлення з помилкою

Якщо створення турніру проходить успішно користувач побачить повідомлення на якому буде це написано, та його перекине на сторінку з турнірами на якій цей турнір і появиться, як і зображено на рисунку 3.9 що буде доказом того що наш турнір добавився.

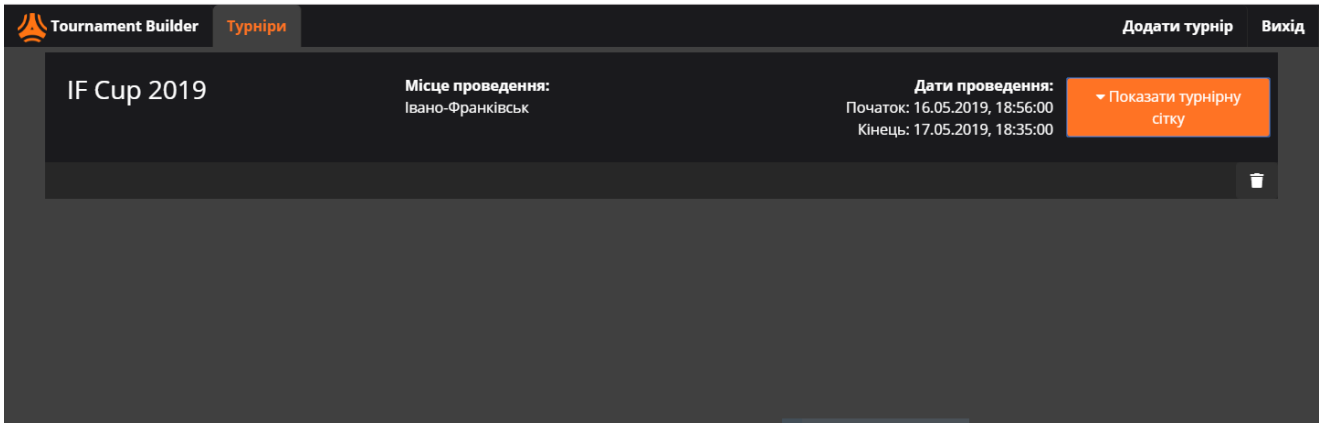

Рисунок 3.9 - Створений турнір

На сторінці, що зображена на рисунку 3.9 будуть відображатися всі турніри які адміністратор буде добавляти. Дані турніри будуть бачити всі користувачі які зайдуть на сайт.

Добавити турнір на сайт це тільки початок роботи, тому що необхідно ще вносити результати матчів для її формування та подальшого відображення користувачам. Для того щоб заповнювати результати матчів адміністратору, або для того щоб їх побачити користувачу необхідно на тиснути на кнопку "Показати турнірну сітку" після чого, не переходячи на інакші сторінки, відкриється пуста турнірна сітка в якої ще відсутні результати матчів, що зображена на рисунку 3.10, з можливістю вводити дані адміністратору, та просто для перегляду користувачів.

Також ключовою можливістю турнірної сітки являється те що вона проводить жеребкування учасників. Тому без різниці в якій послідовності будуть введені команди, бо їх під час створення програма рандомно перемішає, та присвоїть рандомного суперника, а також рандомне порядкове місце серед них. Завдяки цьому ніхто не зможе сказати, що організатори несправедливо назначили суперника чи порядковий номер для проведення матчу.

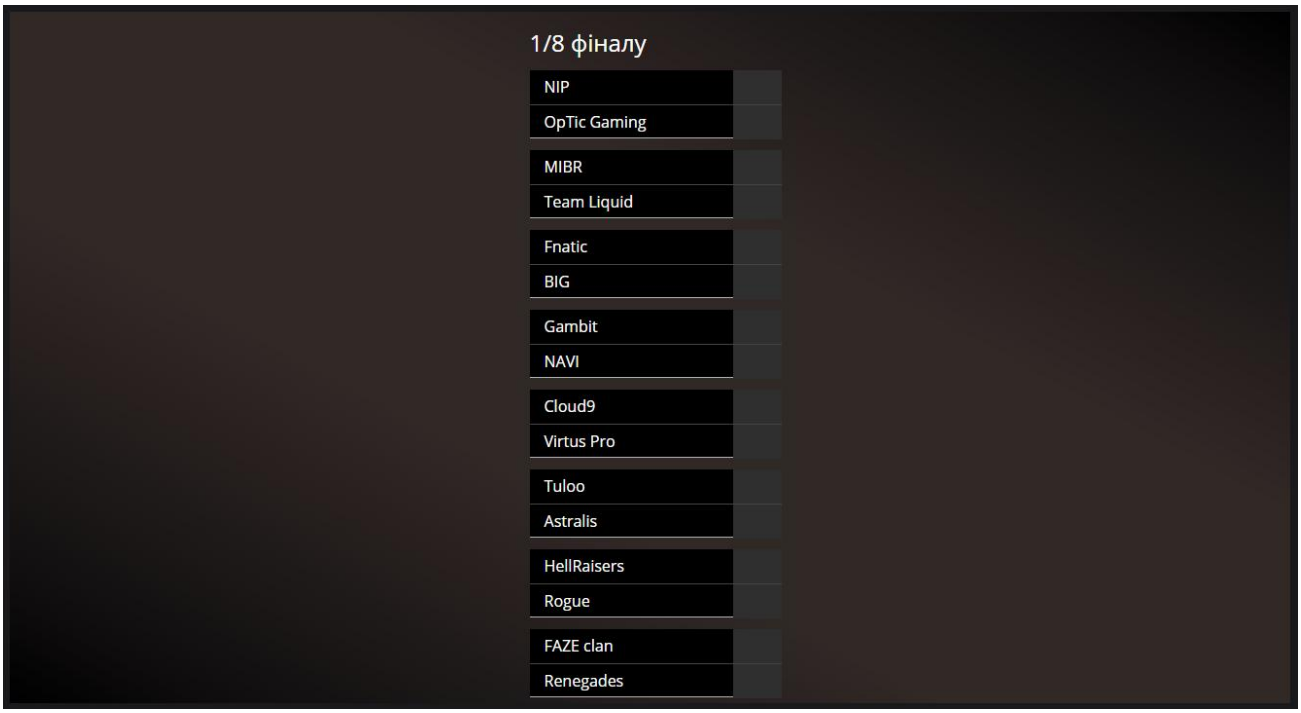

Рисунок 3.10 - Створений турнір

Заповнюючи результати матчів, необхідно обов'язково вказувати тільки числа, валідація не дозволить зберегти результати матчу якщо будуть введені будь що крім чисел, також якщо результати обох команд однакові, оскільки тоді ми не зможемо дізнатися переможця матчу. Тому поки результати матчу не пройдуть цю валідацію адміністратор не зможе їх зберегти, оскільки значок збереження що відображається ліворуч матчу буде неактивним. Більшість перечислених видів валідацій зображені на рисунку 3.11. Після нажаття на кнопку збереження адміністратор отримає невеличке спливаюче повідомлення, що буде про це свідчити, назва команди переможця буде відображатися на оранжевому фоні а сама кнопка збереження зникне як і на рисунку 3.11.

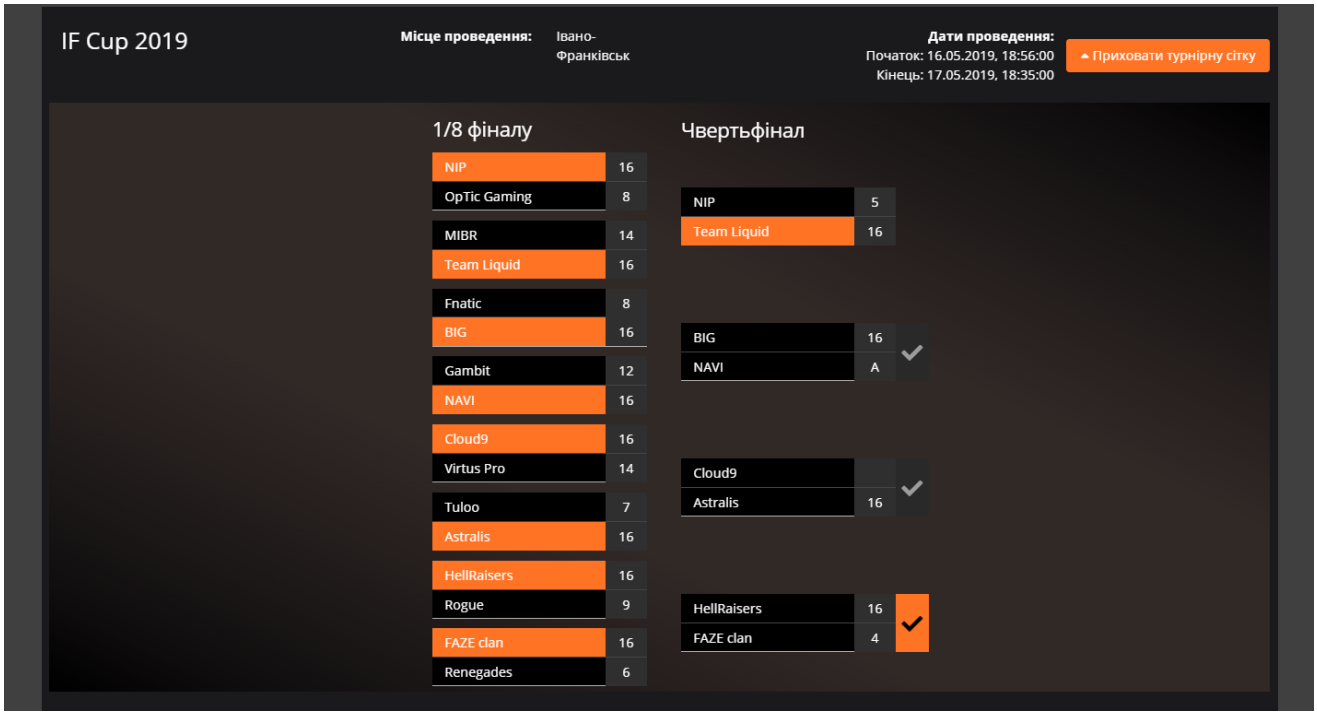

Рисунок 3.11 - Заповнення результатів матчу

Неможна також забувати що турнірна сітка будується поетапно тобто в даній сітці неможливо розпочати наступний етап не завершивши попередні етап. Це запобігає неочікуваних ситуаціям, які в випадку відсутності перевірок можуть призвести до порушення структури бази даних, та до неможливості в подальшому цю турнірну сітку правильно відобразити.

Після того як всі матчі і стадії турніру пройдені переможця турніру буде записано, та добавлено в інформацію про турнір. Фінальний вигляд турнірної сітки на 16 команд з визначеним переможцем зображений на рисунку 3.12.

Рисунок 3.12 - Готова турнірна сітка

Також якщо на сторінку буде знайдено від імені адміністратора, то знизу під кожним турніром буде відображатися кнопка для видалення яку звичайний користувач на відміну від адміністратора бачити не буде Після нажаття адміністратора спитають підтвердження як зображено на рисунку 3.13.

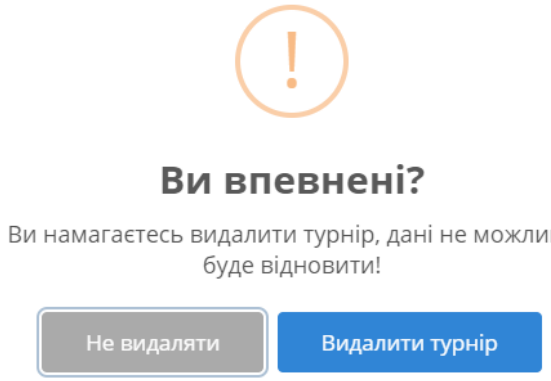

Рисунок 3.13 - Підтвердження видалення

На випадок якщо турнірів буде багато, на сервері була реалізована функція що дозволяє кидати по 16 турнірів на одну сторінку тому обов'язково необхідна можливість між ними переключатися, і ця можливість була реалізована. Тому

якщо зроблено більше ніж 16 турнірів будуть автоматично створені пронумеровані сторінки з можливістю на них перейти. Під час переходу на сторінку фон номера сторінки мінятиме колір на оранжевий як зображено на рисунку 3.14.

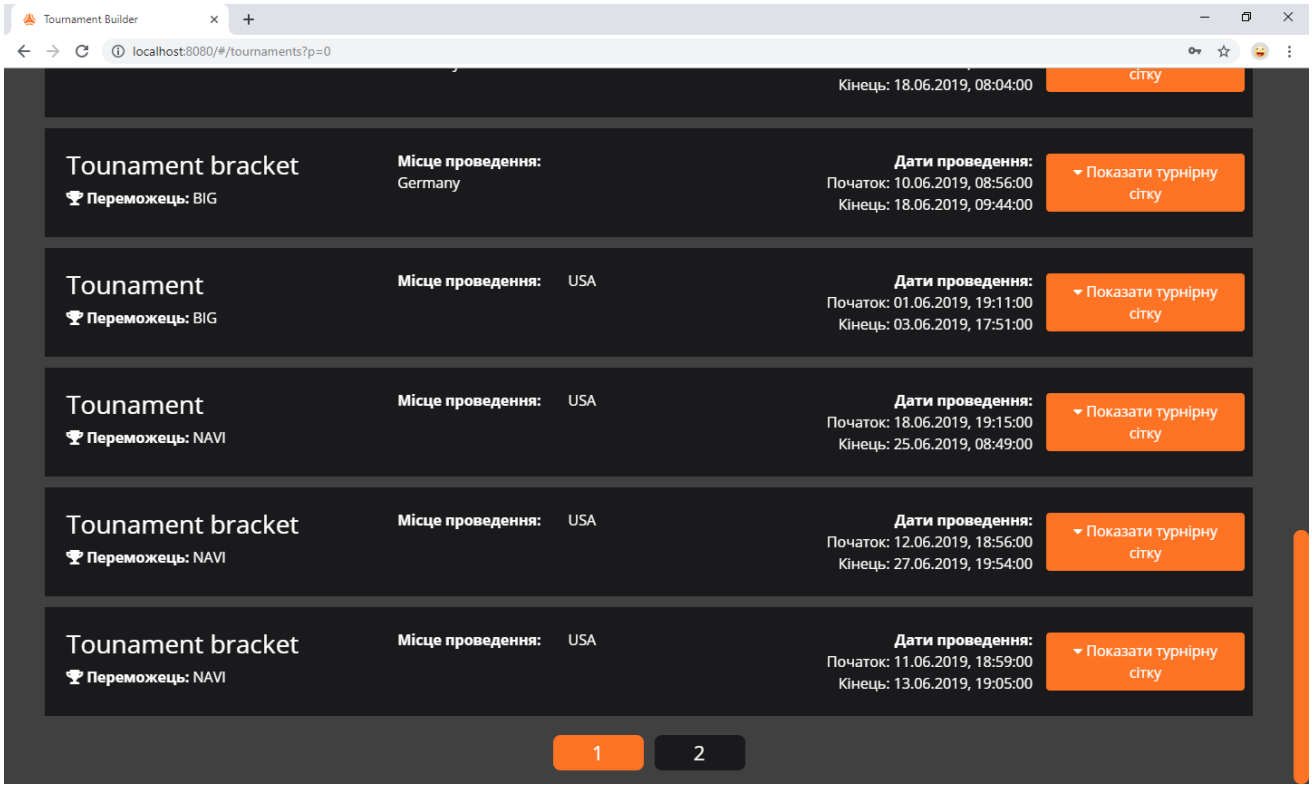

Рисунок 3.14 - Перехід між сторінками

Наступним кроком проект було протестовано з використанням різних браузерів, та з допомогою інструментів розробника браузера Google Chrome, було протестовано відображення на різноманітних пристроях з різною шириною та висотою екрану.

Після того, як розроблений веб сервіс було протестовано на повну функціональну справність можна заявити, що проект являється повністю готовим до розміщення на хостингу, та може бути використаний як користувачами так і адміністратором.

ВИСНОВКИ

В ході виконання дипломної роботи було вибрано мову програмування java та фреймворк Spring, за допомогою якої було розроблено back-end частину веб-сервісу програмного забезпечення для проведення жеребкувань при організації змагань за олімпійською системою.

Розроблене програмне забезпечення містить функціонал для входу адміністратора, додавання турнірів, та команд для турнірів, а також додаванні результатів матчу, та побудові турнірної сітки по вказаних адміністратором результатам.

Для того щоб провести тестування функціоналу back-end з допомогою JS в парі з фреймворком VueJS, мови розмітки HTML, та таблиць каскадних стилів CSS було розроблено інтерфейс веб-додатку для звязку між користувачем та сервером.

Коли для проекту було розроблено front-end та в ньому було використано весь розроблений функціонал back-end, проект було протестованано на повну працездатність та відсутність багів зв’язаних з функціональною частиною. Після чого можна заявити, що проект являється повністю робочим та готовим до використання.

Дипломна робота виконана в відповідності до завдання, та включає в себе теоретичний, та практичний опис проекту. Проект містить функціонал що на даний момент повністю задовольняє адміністратора, та дозволяє йому додавати

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Знаймо. Олімпійська система. URL: http://znaimo.com.ua/Олімпійська_система (дата звернення: 03.12.2018)
2. Турнирные сетки для 16 участников. URL: https://tournamentservice.net/charts.php?chart=5 (дата звернення: 03.12.2018)
3. Faceit. URL: https://www.faceit.com. (дата звернення: 04.12.2018)
4. Эккель Б. Философия Java. 4-е вид., СПб : Питер, 2009. 640 с.
5. Хорстманн К. С., Корнелл Г. Библиотека профессионала. Java 2. В 2т. / Том 1. Основы, 7-е вид., Пер. с англ. Москва : Издательский дом "Вильямс", 2007. Т.1. 896 с.
6. Хорстманн, К., С., Корнелл, Г. Библиотека профессионала. Java 2. В 2т. / Тонкости программирования, 7-е изд., Пер. с англ. Москва : Издательский дом "Вильямс", 2007. Т.2. 1168 с.
7. Шилдт Г. Полный справочнык по Java. Java SE 6 Edition, 7-е вид., Пер. с англ. – Москва: Издательский дом "Вильямс", 2007. 1034 с.
8. Spring Framework — Вікіпедія. URL: https://uk.wikipedia.org/wiki/Spring_Framework (дата звернення: 11.12.2018)
9. Spring Data JPA / Хабр. URL: https://habr.com/ru/post/435114/ (дата звернення: 12.12.2018)
10. REST — Википедия. URL: https://ru.wikipedia.org/wiki/REST (дата звернення: 15.12.2018)
11. Maven і PHP - Блог PHP Academy. URL: https://php-academy.kiev.ua/uk/blog/maven-i-php (дата звернення: 17.12.2018)
12. Tomcat - це веб-сервер або сервер додатків?. URL: http://webukraina.com/questions/9185/tomcat-tse-veb-server-abo-server-dodatkiv-zachineno (дата звернення: 18.12.2018)
13. Юрчишин В.М., Клим Б.В., Кропивницька В.Б. Організація баз даних: навч. посіб. Івано-Франківськ: Факел, 2009. 224 с.

14. Пасічник В.В., Резніченко В.А. Організація баз даних та знань. Київ : Видавнича група BHV, 2006. 384 с.
15. MySQL — Вікіпедія. URL: https://uk.wikipedia.org/wiki/MySQL (дата звернення: 26.12.2018).
16. Bcrypt — Вікіпедія. URL: https://uk.wikipedia.org/wiki/Bcrypt (дата звернення: 28.12.2018).
17. Введение Vue Router. URL: https://router.vuejs.org/ru/ (дата звернення: 03.01.2019).
18. Введение Vuex. URL: https://router.vuejs.org/ru/ (дата звернення: 03.01.2019)
19. Используем Axios для доступа к API — Vue.js. URL: https://ru.vuejs.org/v2/cookbook/using-axios-to-consume-apis.html (дата звернення: 04.01.2019).
20. HTTP-методи запиту - HTTP MDN. URL: https://developer.mozilla.org/uk/docs/Web/HTTP/Methods (дата звернення: 04.01.2019).
21. Основы компонентов — Vue.js. URL: https://uk.wikipedia.org/wiki/JetBrains (дата звернення: 05.01.2019).
22. JetBrains — Вікіпедія. URL: https://ru.vuejs.org/v2/guide/components.html (дата звернення: 20.01.2019).
23. IntelliJ IDEA — Вікіпедія. URL: https://uk.wikipedia.org/wiki/IntelliJ_IDEA (дата звернення: 20.01.2019).
24. Eclipse — Вікіпедія. URL: https://uk.wikipedia.org/wiki/Eclipse (дата звернення: 22.01.2019).
25. WebStorm — Вікіпедія. URL: https://uk.wikipedia.org/wiki/WebStorm (дата звернення: 25.01.2019).
26. Visual Studio Code — Вікіпедія. URL: https://uk.wikipedia.org/wiki/Visual_Studio_Code (дата звернення: 04.02.2019).

БІБЛІОГРАФІЧНА ДОВІДКА

Тема дипломної роботи: Розробка back-end частини веб сервісу проведення жереребкувань при організації змагань.

Обсяг пояснювальної записки:. ___56___аркушів

Перелік графічних матеріалів:

- таблиць_____-_____
- рисунків_____29_____
- додатків_____-_____аркушів.

Дата закінчення дипломного проекту: “__”_____2019 р.

т – дипломник _____
(підпис) (розшифровка підпису)