МІНІСТЕРСТВО ОСВІТИ І НАУКИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ВИПУСКНА РОБОТА

на тему:
«веб-сайт федерації Сумської області з шахів»

Завідувач кафедри **Довбиш А.С.**

Керівник роботи **Олексієнко Г.А.**

Студента гр. ІН-63 **Ільченко А.М.**

СУМИ 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Затверджую_____

Зав. Кафедрою Довбиш А.С.

“_____” _____2020 р.

ЗАВДАННЯ

до випускної роботи

Студента четвертого курсу, групи ІН-63 спеціальності “Інформатика” денної форми навчання Ільченка Антона Миколайовича.

Тема: “**Веб-сайт федерації Сумської області з шахів**”

Затверджена наказом по СумДУ

№_____ від _____2020 р.

Зміст пояснювальної записки: 1) огляд подібних рішень створення веб-сайтів інших федерацій; 2)вибір методів та засобів для розробки веб-сайтів; 3) практична реалізація.

Дата видачі завдання “_____”_____ 2020р.

Керівник випускної роботи _____ Олексієнко Г.А.

Завдання прийняв до виконання _____ Ільченко А.М.

РЕФЕРАТ

Записка: 63 стор., 13 рис., 5 табл., 2 додатків, 13 джерел.

Об’єкт дослідження — Веб-сайт федерації Сумської області з шахів

Мета роботи — розробка зручного веб-сайту федерації Сумської області з шахів із відображенням єдиного рейтинг-листу шахістів

Методи дослідження — системний, дедуктивний та індуктивний аналіз, комп'ютерний експеримент

Результати — розроблено веб-сайт, що відповідає основним потребам інформаційній діяльності федерації Сумської області з шахів на основі веб-фреймворку Javalin мовою програмування Java. Створено єдиний рейтинг-лист шахістів, що відображує зміну рейтингу у часі.

ВЕБ-ФРЕЙМВОРК, POSTGRESQL, JAVALIN, JAVA, CHART.JS, BOOTSTRAP, БЛОГ.

ЗМІСТ

ВСТУП

На сьогодні майже кожна організація: магазин, спортивна федерація або фірма у сфері послуг мають веб-сайт. Веб-сайт – це гіпертекстові веб-сторінки, що об’єднані між собою у єдину структуру, та які доступні через інтернет-мережу. Мережа Інтернет має велику кількість користувачів, яка росте щороку. Відвідувачі веб-сайтів потрібні різноманітним організаціям, як і навпаки, користувачі потребують швидкий доступ до важливої їм інформації. Тому веб-сайт є дуже ефективним методом обміну інформацією.

Робота присвячена покращенню інформаційної роботи федерації Сумської області з шахів у мережі Інтернет за допомогою веб-сайту. Основними цілями роботи є популяризація та покращення висвітлення різноманітних новин шахової федерації.

1. Огляд існуючих рішень. Постановка задачі

1.1 Огляд веб-сайтів шахових федерацій

Для огляду подібних рішень було взято сайти шахових федерацій Львівської, Дніпровської та Миколаївської областей. Увага зверталася на наступні параметри:

1. Дизайн. Веб-дизайн є важливим пунктом у створенні веб-сайтів. За дослідженнями багатьох вчених стало відомо, що людині потрібно 50 мс на оцінювання чи подобається користувачу веб-ресурс на який він зайшов, чи ні. Тому розуміння сучасних трендів у веб-дизайні може допомогти створити цікавий проект. На сьогодні у сфері веб-дизайну присутні наступні тренди:
    - мінімалізм.
    - присутність динаміки
    - якісний фотоматеріал

    Тому в огляді веб-сайтів буде звертатися увага саме на ці параметри.
2. Структура. Структура сайту або наповнення. Аналізуючи подібні рішення ми будемо звертати увагу на цей пункт. Це дасть можливість більш краще зрозуміти - яке наповнення повинне бути на веб-сайті з відповідною тематикою.
3. Швидкодія. Швидкість відповіді веб-сайту на запити користувачів буде вимірюватися за допомогою онлайн-сервісу Google PageSpeed Insights. Зазначений інструмент заміряє швидкість завантаження веб-додатку. Із метрик даного сервісу є: час завантаження першого контенту, індекс швидкості завантаження і т.д. Після тесту веб-сайту буде отримано бали (від 0 до 100), що оцінюють швидкодію. Оцінювання здійснюється наступним чином: 0-49 балів – погано, 50-89 балів – добре, 90-100 балів – відмінно. Запуск вищезазначеного інструменту буде проводитися де-

кілька разів, для виключення різноманітних проблем із зв’язком з інтернет-мережею на стороні користувача.

1.1.1 Веб-сайт шахової федерації Львівської області

Сайт має непоганий дизайн, що має мінімалістичному стилі та виконаний у біло-чорних кольорах. На головній сторінці можна одразу знайти потрібний розділ та знайти шукану інформацію на сайті. Тому юзабіліті сайту добре. Анімації на сайті не має, але добре підібраний фотоматеріал.

В структурі сайту присутні такі розділи:

- новини
- календар шахових подій
- події

Швидкодія веб-сайту непогана. Результати аналізу показані на рисунку нижче:

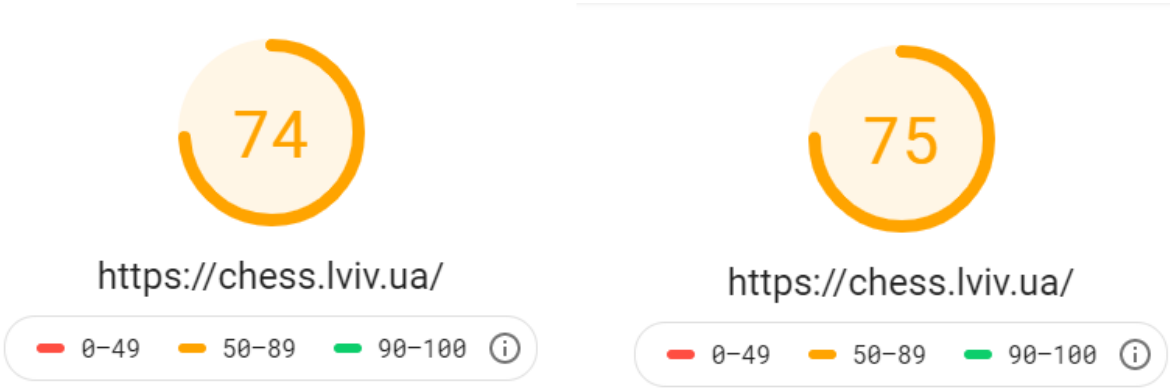


Рисунок 1.1 – результати аналізу швидкодії сайту

1.1.2 Веб-сайт шахової федерації Дніпровської області

Сайт має застарілий дизайн та не виконана у мінімалістичному стилі. Кнопки авторизації та входу для користувача є невеликого розміру, що не підходить до стилю веб-сайту. На головній сторінці зручно розташована інформація.

Сайт має такі основні інформаційні розділи:

- рейтинги шахістів

- результати турнірів
- фотогалерея
- анонси подій
- відведене місце для реклами.

Швидкодія веб-сайту непогана. Результати аналізу показані на рисунку нижче:

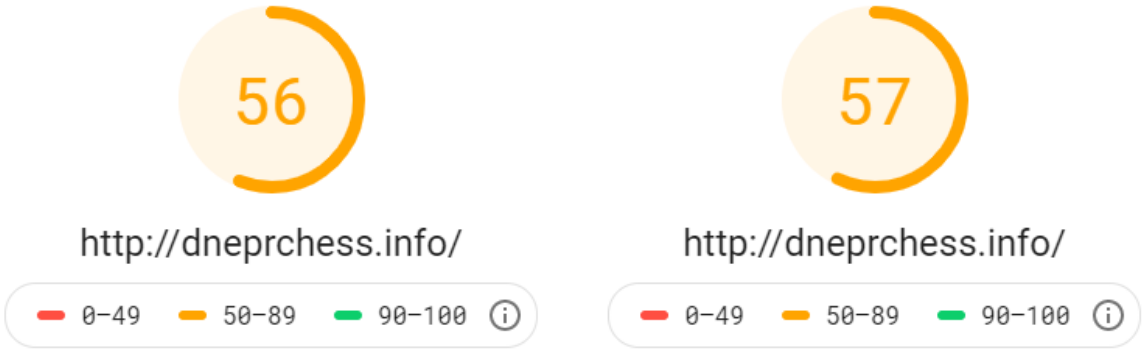


Рисунок 1.2 – результати аналізу швидкодії сайту

1.1.3 Веб-сайт шахової федерації Миколаївської області

Сайт має застарілий дизайн. На головній сторінці погано видно чітке розділення інформації, тому знайти інформацію важко. В структурі сайту присутні багато розділів:

- форум
- новини,
- фото
- архів
- На сайті є функція авторизації користувачів
- пошук на сайті за ключовими словами.
- Відведене місце для реклами.

Швидкодія веб-сайту є повільною. Результати аналізу показані на рисунку нижче:

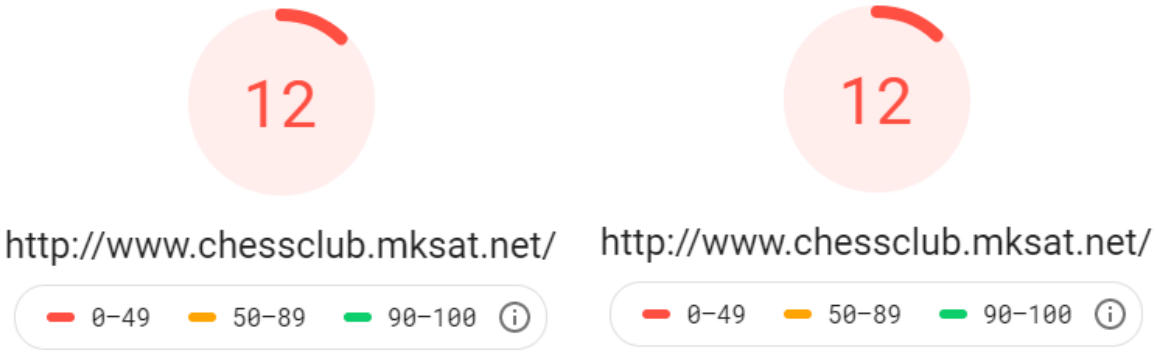


Рисунок 1.3 – результати аналізу швидкодії сайту

1.2 Постановка задачі

Отже, після огляду вибраних веб-сайтів шахових федерацій, що був проведений у попередньому розділі, можна визначити важливі аспекти. Дизайн у розглянутих веб-сайтів в основному є застарілим, що не привертає увагу користувача. Тому у даній роботі потрібно більше зусиль докласти до цією частини сайту. В структурах можна побачити схожі розділи. Це означає, що вони є необхідними для шахового веб-сайту. Нижче наведені розділи, які, можуть бути включені до розроблюваного проекту:

- Новини. (Обов’язково)
- Фотогалерея (Не обов’язково)
- Архів з документами (Не обов’язково)
- Інформація про федерацію. У даному пункті можна розширити функціональність, яка представлена на вищезазначених веб-сайтів: створити невелику інформаційну систему шахової федерації Сумської області. Дані для реалізації цього пункту є на сайті шахової федерації України у відкритому доступі. Більш детально проектування інформаційної системи буде розглянуто у розділі «Програмна реалізація»
- Локалізація. Цей пункт присутній тільки на одному з представлених веб-сайтів. Тому можна зробити локалізацію на української та російської мовами (Не обов’язково)

Щодо швидкодії можна сказати, що вона не є ідеальною на всіх розглянутих шахових веб-сайтів. Оскільки розроблювальний проект не є великим та важким, то з цим пунктом не повинно виникнути проблем.

2. Вибір методу рішення

2.1 Огляд основних способів розроблення

Існує декілька способів розроблення веб-сайтів та веб-додатків. Розглянемо наступні: використання CMS, веб-фреймворку та власне написання, без допомоги CMS та фреймворків:

CMS (Content management system) - система управління контентом. Використовується, коли є тривіальне, стандартне завдання. Наприклад, створення блогу, інтернет-магазину і т.д. Основними функціями CMS є: надання розробнику зручних інтсрументів для створення вмісту веб-сайту та управління вмістом сайту.

Переваги:

- швидкість реалізації проекту
- підтримка проекту CMS системою

Недоліки:

- складність у встановленні авторського дизайну
- складність написання коду для CMS під конкретні завдання

Фреймворк - є програмним середовищем для розробки веб-сайтів, -сервісів, -додатків. Основною задачею веб-фреймворку є автоматизація та полегшення процесу створення веб-додатку, шляхом надання розробникам відповідні бібліотеки для зручної роботи з базою даних, управлінням сесій і т.д.

Переваги:

- легкість у реалізації авторського дизайну
- економія часу програміста на розробку

Власне написання, без допомоги CMS та фреймворків. Можна написати лише невеликі веб-сайти в основному статичні. Одним із плюсів створення веб-сайту власноруч є створення довільної структури сайту. Але мінусів набагато більше:

1. Кількість годин на написання коду збільшується у декілька разів
2. Велика вірогідність написання небезпечного коду
3. Важке масштабування проекту
4. Складність залучення інших програмістів до проекту, тому що архітектуру проекту і його особливості знає тільки власник-розробник

Отже, розуміючи, що наш проект є невеликим та не важким, але у його функціоналі будуть нестандартні функції: наприклад, буде реалізована невелика інформаційна система, що буде потребувати авторського, незвичайного представлення на веб-сайті. Тому буде використовуватися веб-фреймворк для реалізація проекту. Це дасть змогу зекономити час на розробку для програміста відносно того, якщо розробляти веб-сайт без допомоги сторонніх інструментів у вигляді фреймворків або CMS. Система управління контентом теж має можливість набагато швидше прискорити створення веб-ресурсу, але наше завдання є не стандартною, тому буде незручно використовувати її у даному випадку.

Вищезазначена інформація про фреймворк була створена за допомого матеріалу [1, 2, 3].

2.2 Вибір технологій для розробки веб-сайту

2.2.1 Front-end

HTML/CSS. Серед стеку технологій обов’язково буде міститися мова розмітки HTML та таблиця каскадних стилів CSS. Тому що аналогів для цих технологій існує не багато. Ці технології використовуються майже на 100% в усіх веб сайтів та веб додатків.

CSS фреймворк. Створення адаптивного сайту із зручним та добре виконаним дизайном займає багато сил. Тому будемо використовувати css фреймворк з системою сіток

- Bootstrap 4. Bootstrap – являє собою набором безкоштовних стилів та має інструменти для створення адаптивних веб-додатків. Він містить шаблони CSS та HTML. Цей інструмент можна використовувати для створення непоганого дизайну, якщо не має авторського.
- Tailwind CSS**.** Є CSS фреймворком, задачею якого є швидке створення та імплементування авторського дизайну. Головною різницею цього фреймворку від інших є те, що він не містить набір елементів інтерфейсу.
- Material Design**.** Містить у собі всі функції css фреймворку: має адаптивний дизайн за допомогою сіток, набір елементів користувацького інтерфейсу. Відрізняється від інших фреймворків дизайном UI Kit: має дизайн продуктів Google.

У даному розділі був огляд front-end технології для реалізації проекту. Обов’язково було вибрано використовувати HTML/CSS тому що, на сьогодення без цих технологій неможливо створити навіть статичний веб-сайт. Також вибрано Bootstrap 4 серед CSS фреймворків. Цей вибір зумовлено тим, що він є популярним open-sourse проектом, тобто його розвиток підтримує багато кваліфікованих веб-розробників, а також він має достатньо велику функціональність, що може пришвидшити розробку. Для аналізу вищезазначених css-фреймворків була використана література [4, 5]

2.2.2 Back-end

Php - скриптова мова програмування (МП), що інтерпретується на стороні сервера. Використовується для розробки динамічних веб-сайтів.

PHP має наступні переваги:

- Зберігання стану запиту

- Паралельність
- підтримується більшістю хостинг-провайдерами
- один з найпопулярніших мов програмування, що дозволяє краще модифікувати мову

Але є і наступні мінуси:

- Неоднозначність у перетворенні різних типів даних
- протиріччя у найменуванні у стандартній бібліотеці. В одних випадках використовується стиль написання коду CamelCase, в інших випадках з'являється символ _, що нагадує венгерську нотацію. Ця розбіжність збиває логіку найменування функцій, змінних у розробника.

На мові PHP є наступні фреймворки:

1. Laravel - є open-sourse фреймворком, що призначений для створення як простих статичних або динамічних веб-сайтів та веб-додатків. Реалізує шаблон проектування MVC (Model-View-Controller). Має вбудовані модулі для створення безпечної аутентифікації і авторизації користувачів, управління сесій і т.д.
2. Symfony - являє собою набором компонентів, що багаторазово використовуються на МП PHP. Ця особливість цього фреймворку дає можливість створювати як невеликі проекти так і дуже великі, високопродуктивні і водночас масштабовані Використовує версію PHP5 та вище. Підтримує роботу з багатьма системами управління базами даних (СУБД). Наприклад: PostgreSQL, MySQL, SQLite та інші. Є найбільш низькорівневим веб-фреймворком. Багато розширень для інших фреймворків, наприклад laravel та yii2, написано на symfony.
3. Yii2 – є дуже швидким та безпечним веб-фреймворком, написаний на мові програмування PHP, що як і більшість веб-фреймворків реалізує архітектурну модель MVC (model – view – controller). Для обробки залежностей та встановлених модулей використовує диспетчер залежностей Composer. Свою високу швидкість роботи досягає завдяки техніці

ледачого завантаження. Використовується для невеликих так і для високопродуктивних веб-додатків.

NODE js - це опенсорська програмна кросплатформерна платформа на JavaScript, що застосовується, в основному, у ролі веб-сервера. Node js використовує подієво-орієнтоване, що відрізняється від загальної моделі в якій головну роль грають паралельні потоки OS. Також ця МП використовує асинхронне програмування, що дозволяє реагувати на вхідні запити до сервера та водночас виконувати інші дії, наприклад виконання AJAX запиту. Але це не зовсім відповідає дійсності.

Переваги:

- Легкість у написання
- За рахунок використання JavaScript, швидке входження front-end розробників

Недоліки:

- вимагає високої кваліфікації для розробки великих проектів
- досить повільний

ASP.NET core - являє собою opensource-фреймворк для розробки, в основному, великих веб-сервісів та веб-додатків. Дана технологія може працювати як на платформі .net framework так і на .net core, де остання є кроссплатформенною. Використовується МП c#. Присутня модульність фреймворка, де всі компоненти можуть бути додані за допомогою менеджера пакетів Nuget.

Переваги:

- Високий рівень безпеки веб-додатка, тому використовується у великих проектах
- Кроссплатформеність на відміну від asp.net mvc
- Велика швидкість. За рахунок компіляції коду середовищем CLR

Недоліки:

- Велика ціна розробки (якщо порівнювати із розробленням на php, python, node)

Python - динамічний та високорівнева мова програмування загального призначення. Основні особливості: має простий синтаксис, динамічну типізацію, автоматичне управління пам'яттю. На цій МП можна використовувати наступні підходи програмування: імперативне, структурне, функціональне, об'єктно-орієнтоване. Одним з найпопулярніших фреймворків на цій МП є Django:

Django - opensource-фреймворк для розробки веб-додатків. Як і більшість веб-фреймворків використовує шаблон проектування MVC (Model-View-Controller). Істотною особливістю фреймовка є те, що веб-сайт або веб-додаток рекомендується будувати на основі декількох програм, що є незалежними.

Переваги:

- Швидка розробка проектів
- Простий синтаксис
- Підтримка проекту співтовариством
- Дешева розробка

Недоліки:

- не підходить для написання великих проектів
- поступається безпекою веб-додаткам написаних мовою c#, java

Java - популярна мова програмування, що використовує об’єктно-орієнтовану парадигму програмування. В основному використовується для написання веб-додатків великих за розмірів. Синтаксис цієї мови схожий на синтаксис сі-подібних мов: C, C++, C#. Основними плюсами використання цієї мови програмування є:

- безпека
- велика кількість готових рішень
- кроссплатформенність

Недоліки:

Основним мінусом є те, що час запуску програми більше у порівнянні з іншими МП, такі як C,C++. Це залежить від того, що Java використовує

java-машину, що запускає, написану нами програму, перетворює код в байт-код, а потім вже в команди операційної системи. Оскільки розроблювана програма є веб-додатком, що буде запускатися дуже рідко, і час запуску не є дуже важливим фактором, то цей недолік в даному випадку не є великим мінусом.

Деякі Java веб-фреймворки:

1. Spring Framework - великий та популярний веб-фреймворк на Java. Має рішення для основних проблем, що виникають у створенні веб-додатків, використовуючи мову Java. Наприклад, у даному фреймворку є модулі, що допомагають у деяких зазначених нище процесах створення та роботи з веб-додатком:
    1. Робота з базами даних
    2. Керування транзакціями
    3. Використання схеми розділення логіки додатку, даних та інтерфейсу: MVC (Model-View-Controller)
    4. Авторизація та аутентифікація користувачів
2. JSP (Java Server Pages) - менш популярний веб-фреймворк у порівнянні з Spring та старіший, але все ще використовується у розробці. Дана технологія дозволяє створювати динамічні веб-сторінки. У гіпертекстові сторінки вставляється java код. Приймаючи запити від клієнта сервер формує потрібну сторінку та відправляє вже готовий html-документ.
3. Javalin - невеликий java веб-фреймворк, що займає лише декілька тисяч рядків коду, тому легко знайти цікавий розробнику клас. Але при цьому javalin має всі основні модулі для створення веб-додатків. Достатньо легкий у розумінні, а, отже, ідеально підходить для навчальних проектів.

Інформація про МП java та деякі її фреймворки були взяті з ресурсів [6, 7, 8]

Оскільки існує дуже багато технологій для серверної частини веб-розробки, то потрібно вибирати найбільш зручні та підходящі для відпові-

дного проекту. Оскільки розроблюваний веб-сайт не є великим та важким проектом, то майже всі розглянуті мови програмування та фреймворки підходять до поставленої задачі. Проаналізовані технології достатньо є популярними, тому знайти дешевого хостинг провайдера буде легко. Але це не стосується ASP.NET CORE. Ціна хостингів, які підтримують згадана технологію, вище у порівнянні з іншими розглянутими технологіями. Серед оглянутих МП та фреймворків буде вибрано Java та фреймворк Javalin. Java – використовується переважно для розробки великих проектів, але на цій МП також можна розробляти швидко та безпечно для користувача. Фреймворк javalin був вибраний оскільки він легкий, займає всього лиш декілька тисяч рядків коду, легкий для розуміння, що також дуже важливо для швидкості розробки та має весь необхідний функціонал для розробки.

2.2 Вибір СУБД

Оскільки бюджет розробляємого веб-сайту дуже малий, то вибір має проводитися тільки з безкоштовних СУБД.

Postgresql - професійна безкоштовна СУБД з відкритим кодом, яка має дуже багатий функціонал та при цьому високу продуктивність, стабільність до високонавантажених систем. Postgresql має велике співтовариство, яке прислухається до мінусів даної СУБД від розробників та активно працює над ними, що робить цю СУБД швидко розвиваючою та швидко набираючою популярність. Також postgresql має дуже багато доповнень, не зважаючи на велику кількість функцій. Доповнення додають варіативність та додаткові можливості збереження даних інформаційної системи та управління цими даними. Також в даній системі управління базами даних можна створювати розширення, ще більше розширюючи базовий функціонал за рахунок своїх, власноруч створених, процедур. Дана СУБД намагається підтримувати стандарт ACID (Atomicity, Consistency, Isolation, Durability), що робить дану систему дуже стабільною. Недоліків у даної СУБД майже не має. За мінус до да-

ної системи можна віднести складність адміністрування, але це виходить вже із того, що вона має великий функціонал. Підбиваючи підсумки, можна відмітити наступні плюси та мінуси розглянутої СУБД.

Переваги:

1. Активне велике співтовариство
2. Об’єктність
3. Велика кількість доповнень
4. Можливість створення розширень

Недоліки:

1. Складне адміністрування

Mysql - популярна СУБД з відкритим кодом, що має достатньо великий функціонал, але не всі команди з мови SQL реалізовані. Система, що має можливості до масштабування, має непогану продуктивність. Працює за клієнт-серверною архітектурою на відміну від SQlite. Існує багато розширень, що дозволяє ще зручніше працювати з даної системою. Mysql входить до набору багатьох серверів та різноманітних збірок, наприклад: denwer, XAMPP, WAMP, … При цьому майже завжди має GUI (Graphical User Interface), що дуже зручно для початкових розробників. Є підтримка механізму реплікації - синхронізація декількох об’єктів (таблиці, схеми і т.д.), тобто, це процес копіювання даних з одного джерела до іншого, роблячи ці джерела однаковими за структурою та за даними. Оскільки розглянута СУБД була в один час найпопулярніша, то існує дуже багато готових рішень та навчальних матеріалів, що полегшує роботу програміста, системного адміністратора, користувача.

Переваги:

1. Легкість в адмініструванні
2. Достатньо великий функціонал
3. Велика кількість готових рішень і різноманітних матеріалів
4. Безкоштовна

Недоліки:

1. Неповна реалізація можливостей мови SQL

2. Нестабільна при великих навантаженнях

SQLite - СУБД, що має можливість легкого підключення та вбудовування у програми. Ця система не схожа на дві вище розглянуті: вона не використовує архітектуру клієнт-сервер, не являється іншою програмою в іншому потоці в операційній системі. SQLite - є бібліотекою і через програмний інтерфейс (API - application programming interface) використовується СУБД. Система використовується у додатках, які мають одного користувача, наприклад: ігри, мобільні додатки, веб-додатки, які носять інформаційний характер і наповнення контенту виконується рідко та не багатьма користувачами. Область використання розглянутої СУБД залежить від обмеження запису: можна роботи запис тільки в одному потоці, в одну одиницю часу - лише один запис. Тому дана СУБД не використовується у великих проектах, які зберігають багато даних. Швидкість SQLite - висока, оскільки програми звертаються безпосередньо до файлів, де зберігаються дані.

Переваги:

1. Швидкість
2. Переносимість бази даних

Недоліки:

1. Немає системи користувачів
2. Обмеження на запис до БД

Отже, виходячи з розміру проекту, планів та можливостей його масштабування була вибрана СУБД PostgreSQL. Дана СУБД має високу швидкість читання даних, використовується у всіх видах проектах, у тому числі і малих, до якого і відноситься розроблюваний веб-сайт. Дана СУБД має зручний інтерфейс PgAdmin для роботи з нею, що є невеликим, але приємним плюсом, на відміну від SQLite, яка не має офіційного зручного користувацького інтерфейсу. Для аналізу СУБД використовувалася література [9, 10]

2.3 Вибір інструментів розробки

Оскільки буде виконуватися на МП java, то для розробки веб-сайту необхідно обов'язково вибрати інтегроване середовище розробки (IDE).

IDE (Integrated Development Environment) - це комплексна програма, що містить набір інструментів, які використовуються програмістами для зручної розробки додатків. В загальному випадку середовище містить наступні компоненти:

- Текстовий редактор коду, що підсвічує синтаксис МП на якій розроблюється додаток.
- Компілятор, інтерпретатор
- Інструменти для відлагодження коду.

Деякі IDE, що використовуються при розробці на МП java:

IntelliJ IDEA - має дві версії: безкоштовну - Community edition та платну версію - Ultimate edition. Для розробки веб-сайтів використовується платна версію програми. Вона підтримує розробки не тільки на мові програмування, що використовується для серверної частини веб-сайту, але у даній IDE можна писати на javascript та typescript. Також є підтримка основних java фреймворків: Spring Framework, Vaadin, Play …. Політика компанії, що створює розглянуту IDE підтримує студентів, вчителів, open-source проекти та надає платну версію безкоштовно.

Eclipse - безкоштовне середовище, що має відкритий код, написана на Java. Раніше мала дуже велику популярність, але з кожним роком вона падає. Eclipse - достатньо повільна IDE, що робить розробку менш швидкою та менш приємною. Підключення плагінів до середовища також є не дуже приємний процесом. Деякі розширення погано взаємодіють між собою, хоча в документації до цих плагінів дані проблеми не вказуються.

NetBeans - кросплатформне середовище, що може запускатися на будь-яких системах, що містять JVM (Java Virtual Machine). Існує декілька різних пакетів для встановлення різних МП, фреймворків, в тому числі і для

веб-розробки. При необхідності можна встановити всі розширення, що не будуть конфліктувати між собою на відміну від плагінів Eclipse.

Для зручної верстки веб-сайту можна використовувати додаткове програмне забезпечення, що спеціалізується саме на цьому завданні:

- Sublime text 3. Це є простим редактором і дуже зручним для мови розмітки HTML та для CSS. За рахунок хорошого та зручного дизайну, що пришвидшує написання коду та за рахунок плагінів, які можна додатково встановити. Один із найпопулярніших та зручних плагінів для HTML/CSS у sublime text 3 можна відмітити Emmet.
- Notepad++. Є зручним текстовим редактором, що підтримує багато форматів та підсвічує синтаксис багатьох мов програмування, а також мов розмітки. Є менш популярним ніж sublime text 3, але не менш функціональний

Для розробки будемо використовувати платну версію IDE IntelliJ IDEA, оскільки по-перше це середовище має усі необхідні інструменти для створення веб-сайтів, які достатньо швидко працюють, а по-друге можна оформити платну версію середовища для студентів безоплатно.

3. Програмна реалізація

3.1 Проектування бази даних для зберігання зміни рейтингу

Основний вигляд даних для яких буде створюватися база даних має наступний вигляд (Рисунок):

РЕЙТИНГ - ЛИСТ НА 01.04.2020 Сумська
Примітка: w -жін.; w-неакт. FIDE; -неакт. Укр;

Прізвище та ім'я	Звання	Парт	Доб	УКР	Д/народ.	ФЕД	#IDF	Д/дії	Прім.
Ахмедзянов Павло		0	0	1920 0	12.07.94	СУМ	20195	01.01.19	н
Ащаулова Тетяна		0	0	1924 0	12.01.02	СУМ	14157136	01.07.21	w
Бабовал Анатолій	кмс	0	0	2188 0	10.07.76	СУМ	20002	01.04.20	
Байдаков Микола		0	0	0	21.05.00	СУМ	20244	01.01.21	
Балицький Роман	кмс	0	0	2125 0	21.03.87	СУМ	14131072	01.03.22	
Балицький Юрій	кмс	0	0	2151 0	17.03.83	СУМ	20004	01.03.22	
Безродний Олександр		0	0	0	21.03.56	СУМ	20249	01.04.21	
Белеванцев Анатолій		0	0	1915 0	28.05.38	СУМ	20145	01.07.16	н
Бірюков Віктор		0	0	0	24.08.46	СУМ	20251	01.07.21	
Богданов В.		0	0	2019 0	.....	СУМ	20005	01.07.09	н
Болдирєв Антон		0	0	2036 1680	22.06.75	СУМ	14177870	01.10.21	
Бражник Борис		0	0	0	11.07.45	СУМ	20252	01.07.21	
Брушко Олександр	кмс	0	0	2091 0	24.03.63	СУМ	20104	01.04.14	н
Буц Віктор		0	0	1815	15.03.44	СУМ	20227	01.07.21	

Рисунок 3.1 – приклад рейтинг-листу

Оскільки мета проекту реалізувати показ зміни саме українського рейтингу шахістів, тому були взяті дані не з всіх стовпців вище зазначеного документу, а саме:

- прізвище,
- ім’я,
- звання,
- партії,
- український рейтинг,
- дата народження шахіста,
- федерація,
- ідентифікатор в український (або в міжнародній) федерації,

- дійсність шахіста,
- стать шахіста.

Із зазначених стовпців створили базу даних, що буде містити відповідні поля.

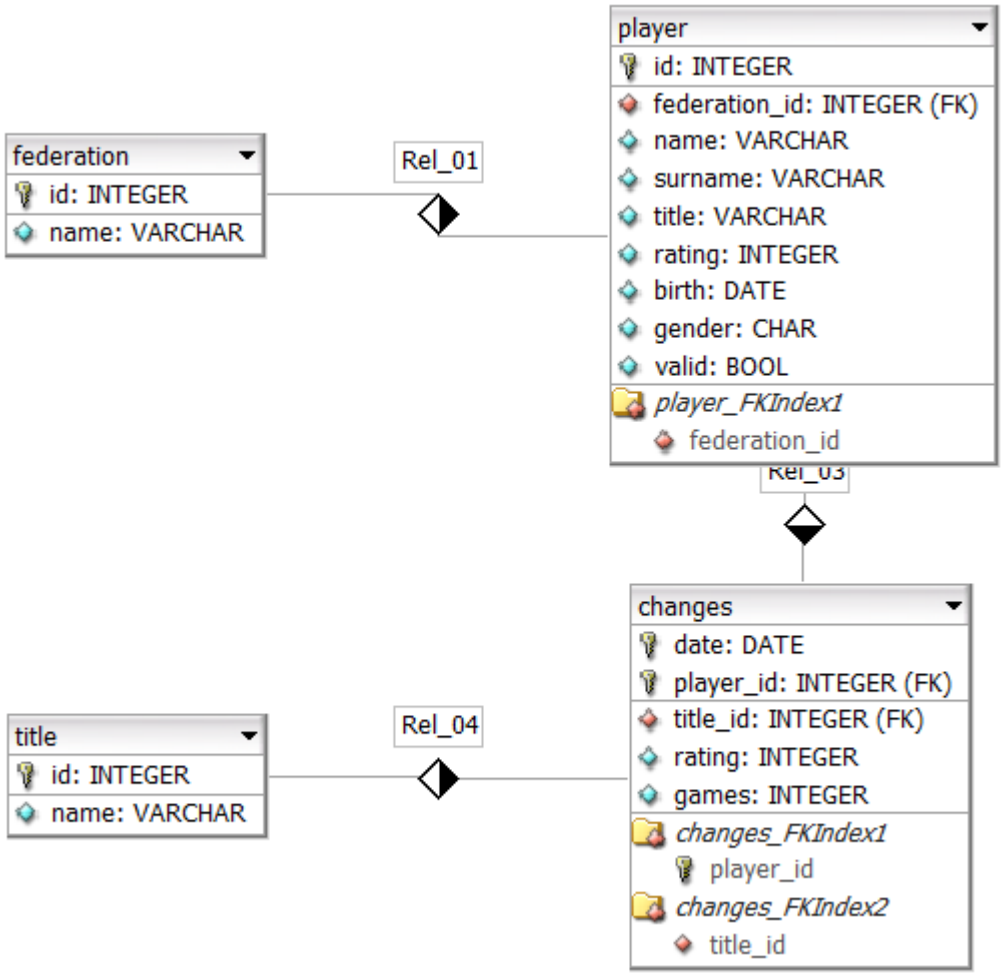


Рисунок 3.2 – ERD діаграма БД зміни рейтингу шахістів

Характеристику ERD діаграми у вигляді таблиці, що зображена нижче:

Таблиця 3.1 – характеристика БД

Таблиця	Поле	Тип данних	Ключі	Обмеження
federation	id	INTEGER	PK	Не пустий
	name	VARCHAR		
title	id	INTEGER	PK	Не пустий
	name	VARCHAR		
player	id	INTEGER	PK	Не пустий

	federa-tion_id	INTEGER	FK – federation	Не пустий
	name	VARCHAR		
	surname	VARCHAR		
	title	VARCHAR		
	rating	INTEGER		
	birth	DATE		
	gender	CHAR		
	valid	BOOL		
changes rating	date	DATE	PK	Не пустий
	player_id	INTEGER	PFK - player	Не пустий
	title_id	INTEGER	FK - title	Не пустий
	rating	INTEGER		
	games	INTEGER		
title	id_title	INTEGER	PK	Не пустий
	name	VARCHAR		

У таблиці Player поля rating та title повинні бути видалені з цієї сутності за 3 нормальної формою. Але в даному випадку, оскільки дані в даній таблиці будуть оновлюватися рідко та щоб отримати значення зазначених полів потрібно буде використовувати SQL конструкцію JOIN з таблицею changes, із-за якої швидкість отримання даних сильно впаде. Тому було зроблене рішення залишити зазначені поля.

3.2 Парсинг даних

Дані парсились зі сторінки кваліфікаційної комісії сайту шахової федерації України (ФШУ).

Було декілька варіантів як можна були брати дані на вказаному вище сайті:

1. Дані усіх федерацій за конкретний період
2. Дані кожної федерації за конкретний період.

Був вибраний другий варіант, тому що за всі періоди, які вказані на сайті, дані існують на кожну федерацію.

На сайті ФШУ зберігаються дані шахістів від поточної дати до січня 2002 року. Оскільки сайт ФШУ часто змінювався, структура сторінок також змінювалось, а тому зібрати усі дані важко. Тому було зроблене рішення парсити дані від поточної дати до жовтня 2014 року. За цей період сторінки з рейтингом кожної федерації змінювалась 2 рази: 01.10.2016, 01.07.2014. Зміна структури сторінки показана на рисунки нижче:

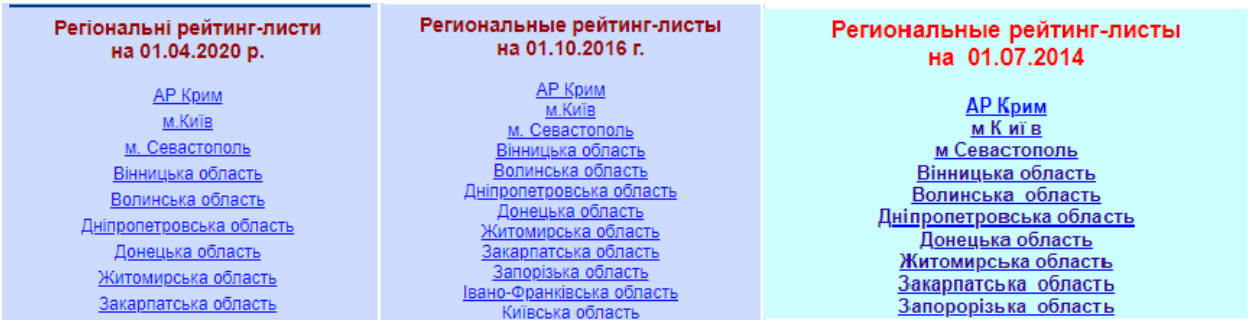


Рисунок 3.3 – зміна структури сторінок з рейтингами федерацій

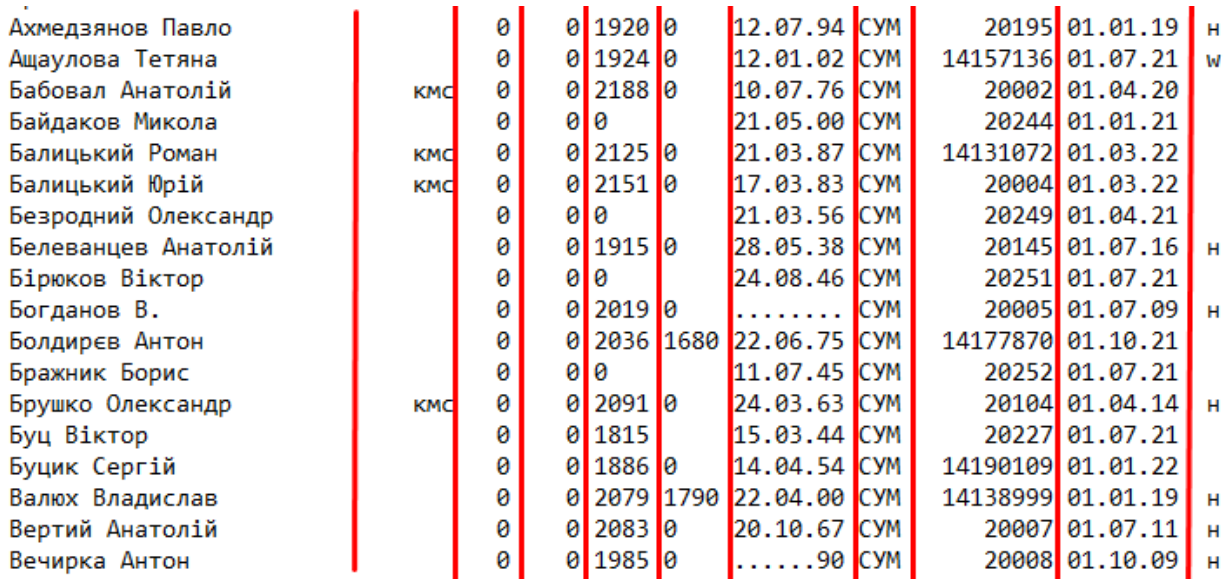
Результатом парсингу даних, отримали 41 дат, кожна з яких має 27 файлів, що містять рейтинг шахістів відповідної федерації. Для рішення різноманітних проблем з парсингом даних використовувалася література [13]

3.3 Обробка отриманих даних та створення записів у БД

Алгоритм заповнення БД даними наступний:

1. Опрацювання можливості зчитування рейтингів з текстових файлів та занесення до створеного класу «Player».

Для парсингу даних про шахіста був вибраний метод взяття даних за пробілами, оскільки дані у рейтинг-листу розташовані майже завжди однаково (кількість пробілів між стовпцями однаковий). На рисунку нижче наведено приклад:



```
Ахмедзянов Павло          0  0 1920 0    12.07.94 СУМ    20195 01.01.19 н
Ащаулова Тетяна           0  0 1924 0    12.01.02 СУМ 14157136 01.07.21 w
Бабовал Анатолій    кмс   0  0 2188 0    10.07.76 СУМ    20002 01.04.20
Байдаков Микола           0  0 0       21.05.00 СУМ    20244 01.01.21
Балицький Роман     кмс   0  0 2125 0    21.03.87 СУМ 14131072 01.03.22
Балицький Юрій      кмс   0  0 2151 0    17.03.83 СУМ    20004 01.03.22
Безродний Олександр       0  0 0       21.03.56 СУМ    20249 01.04.21
Белеванцев Анатолій       0  0 1915 0    28.05.38 СУМ    20145 01.07.16 н
Бірюков Віктор            0  0 0       24.08.46 СУМ    20251 01.07.21
Богданов В.               0  0 2019 0    ........ СУМ    20005 01.07.09 н
Болдирєв Антон            0  0 2036 1680 22.06.75 СУМ 14177870 01.10.21
Бражник Борис             0  0 0       11.07.45 СУМ    20252 01.07.21
Брушко Олександр    кмс   0  0 2091 0    24.03.63 СУМ    20104 01.04.14 н
Буц Віктор                0  0 1815    15.03.44 СУМ    20227 01.07.21
Буцик Сергій              0  0 1886 0    14.04.54 СУМ 14190109 01.01.22
Валюх Владислав           0  0 2079 1790 22.04.00 СУМ 14138999 01.01.19 н
Вертий Анатолій           0  0 2083 0    20.10.67 СУМ    20007 01.07.11 н
Вечирка Антон             0  0 1985 0    ......90 СУМ    20008 01.10.09 н
```

Рисунок 3.4 – розмежування рейтинг-листа

Використані відрізки за кількістю символів наведені в таблиці нижче:

Таблиця 3.2 – відрізки символів відповідних стовпців

№	Стовпець	Кількість символів у відрізку. (Включно)
1	Прізвище та Ім’я	0 – 25
2	Звання	26 – 32
3	Кількість зіграних ігор	33 – 36
4	Число зміни рейтингу	37 – 41
5	Український рейтинг	42 – 46
6	Міжнародний рейтинг	47 – 51
7	Дата народження	52 – 59
8	Федерація	60 – 64
9	Український або міжнародний ідентифікатор	65 – 75

10	Дата дійсності рейтингу шахіста	76 – 83
11	Примітки	85 – кінець рядка

Деякі стовпці в рейтинг-листах мали інакший вигляд. Також для більш зручного використання даних були зроблені маніпуляції із початковими даними стовпця наведені в таблиці нижче:

Таблиця 3.3 – важливі особливості стовпців рейтинг-листу

№ (номер стовпця. Таблиця -)	Особливість стовпця та його зчитування
1	Ім’я шахіста може складатися більше ніж 2 слова. Прізвище шахіста завжди має 1 слово
2	При зчитуванні даних може бути 2 звання: український та міжнародний. Використовується українське звання
7,10	Може бути декілька варіантів представлення дати: 1. DD.MM.YYYY (вказано день, місяць, рік) 2. …….. (пропущено в різних варіаціях день, місяць. Пропущену цифра замінена на крапку) 3. __.__.__ , де _ це пробіл (пропущено в різних варіаціях день, місяць. Пропущену цифра замінена на пробіл)

Клас «Player», призначення якого зберігання взятих із рейтинг-листу даних шахіста. Структура класу наведена у таблиці нижче:

Таблиця 3.4 – структура класу Player

Поле	Пояснення
private String name	Ім’я шахіста. Може складатися більше ніж 2 слова
private String surname	Прізвище шахіста. У прізвищі завжди 1 слово
private int title	Ідентифікатор звання шахіста. При зчитуванні даних може бути 2 звання: український та міжнародний. Використовується українське звання
private int games	Кількість зіграних ігор
private int ratingModified	Змінений рейтинг
private int ratingUkr	Український рейтинг
private int ratingFide	Міжнародний рейтинг
private String birth	Дата народження.
private String federation	Федерація шахіста
private int idFederation	Ідентифікатор федерації
private int id	Ідентифікатор шахіста
private String dateValid	Дата дійсності
private char gender	Стать шахіста
private boolean valid	Дійсність
private static Map<String, Integer> titles	Хеш-таблиця, що містить відповідність ідентифікатора назві звання

private static Map<String, Integer> regions	Хеш-таблиця, що містить відповідність ідентифікатора федерації назві федерації
Методи	
getTitles, getRegions, getName, getSurname, getTitle, getGames, getRatingModified, getRatingUkr, getRatingFide, getBirth, getFederation, getIdFederation, getId, getValidDate, getGender, isValid.	Створені гетери для всіх полів класу
private boolean isInt(String strNum)	Функція визначення чи є рядок числом
private String getBirthByString()	Функція перетворення дати народження у потрібний формат
private String getValidDateByString()	Функція перетворення дати дійсності у потрібний формат
public static void intializeTitlesRegions()	Функція для ініціалізації полей tiles, regions

2. Створення запитів для створення записів у БД таблиці «title».
3. Створення запитів для створення записів у БД таблиці «federation».
4. Створення запитів для створення записів у БД таблиці «player».
5. Створення таблиці аутентифікації.

Оскільки було виявлено, що ідентифікатор шахіста може змінюватися (національний ідентифікатор може змінитися на міжнародний, але людина одна і та ж сама), то було вирішено створити таблицю, яка зіставляє для ша-

хістів національний ідентифікатор та міжнародний. Ідентифікатори були записані лише для шахістів, які є в національному рейтингу на квітень 2020 року. Кожному українському ідентифікатору відповідає міжнародний або український ідентифікатор, якщо у шахіста не має міжнародного. Ці данні при створенні запитів будуть занесені до хеш-таблиці, що зробить швидкий доступ до потрібного ідентифікатора. Приклад на рисунку нижче

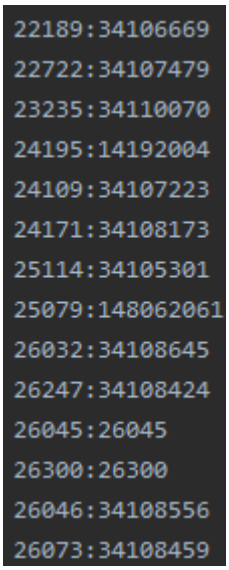


Рисунок 3.5 – приклад фрагменту таблиці аутентифікації гравців

6. Зчитуємо данні шахістів конкретної дати та заміняємо ідентифікатор на той, який знаходиться у таблиці ідентифікації, яка була створена на попередньому кроці.
7. Використовуємо змінені дані для створення запитів внесення даних до таблиці «changes» в БД.
8. За допомогою PgAdmin виконуємо створені запити на попередньому кроці.

Результат виконаних дій можна побачити на рисунку, що наведений нижче:

	date [PK] date	player_id [PK] integer	title_id integer	rating integer	games integer
1	2015-01-01	14106680	2	2207	0
2	2015-01-01	14115743	2	2127	0
3	2015-01-01	14134098	1	2061	0
4	2015-01-01	14109069	3	2423	0
5	2015-01-01	12206	1	1833	0
6	2015-01-01	14129485	2	2078	0
7	2015-01-01	14106663	3	2376	0
8	2015-01-01	14122910	1	1994	0
9	2015-01-01	12003	2	2071	0
10	2015-01-01	14103036	1	2080	0

Рисунок 3.6 – дані в таблиці «changes»

3.4 Серверна частина

Архітектура веб-сайту складається з 3 основних рівнів:

1. Рівень для роботи з базою даних (Використовувався підхід DAO – data access object)
2. Бізнес-логіка (мають сутності та класи, що керують існуючими сутностями)
3. Зв’язок із інтерфейсом додатку (Controllers – обробляють запити, що надходять від користувачів)

Між собою рівні мають наступний взаємозв’язок: рівень бізнес-логіки може використовувати рівень для роботи з базою даних, а рівень інтерфейсу – бізнес-логіку. Таке розмежування дозволяє більш зручно масштабувати додатки, краще орієнтуватися в коді, тобто, позитивно впливає на підтримку коду в подальшому. Вказані взаємозв’язки між рівнями зображені на рисунку нижче:

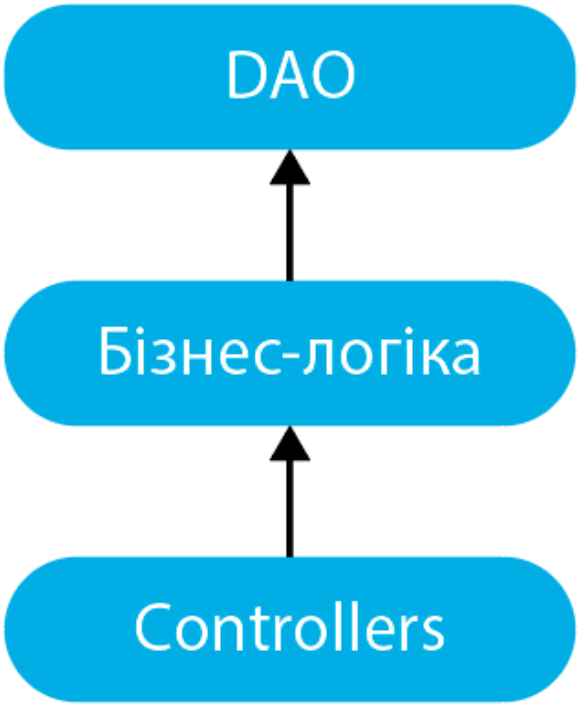


Рисунок 3.7 – архітектура серверної частини веб-сайту

Для створення вищезазначеної архітектури серверної частини веб-сайту використовувалася література [11, 12]

Серверна частина отримує запит від користувача на рівні Controller та віддає відповідь у форматі json.

Були реалізовані наступні запити, що зазначені в таблиці нижче:

Таблиця 3.5 – реалізовані запити

Метод	Запит	Пояснення
post	api/players	Запит повертає всіх гравців. Запит приймає 2 параметри: limit, offset. Ці параметри впливають на кількість повернених гравців.
post	api/players/federation	Запит повертає гравців конкретної федерації.

		Запит повертає всіх гравців. Також запит приймає 3 параметри: limit, offset, idFederation. Limit, offset - впливають на кількість повернених гравців, idFederation – ідентифікатор федерації
post	api/players/surname	Запит повертає гравців, які мають схоже прізвище. Запит повинен мати 3 параметри: limit, offset (впливають на кількість гравців, що повертаються), surname – прізвище гравця
post	api/posts	Повертає всі пости. Запит повинен мати 3 параметри: limit, offset (впливають на кількість гравців, що повертаються)
post	api/posts/id	Запит повертає пост за вказаним ідентифікатор, що передається при запиті у тілі запиту (параметр id)
post	api/posts/title	Запит повертає пости за схожою назвою. Запит має 3 параметри: limit, offset (впливають на кількість постів, що повертаються), title – назва поста
post	api/posts/create	Запит створює пост у БД. Запит повинен всі необхідні параметри для створення поста: title, description, text, image

post	api/posts/count	Запит повертає загальну кількість постів у базі даних. Запит використовується при створенні пагінації на сайті
get	/images	Запит повертає відповідну до переданого ідентифікатору картинку до поста.

3.5 Демонстрація роботи веб-сайту

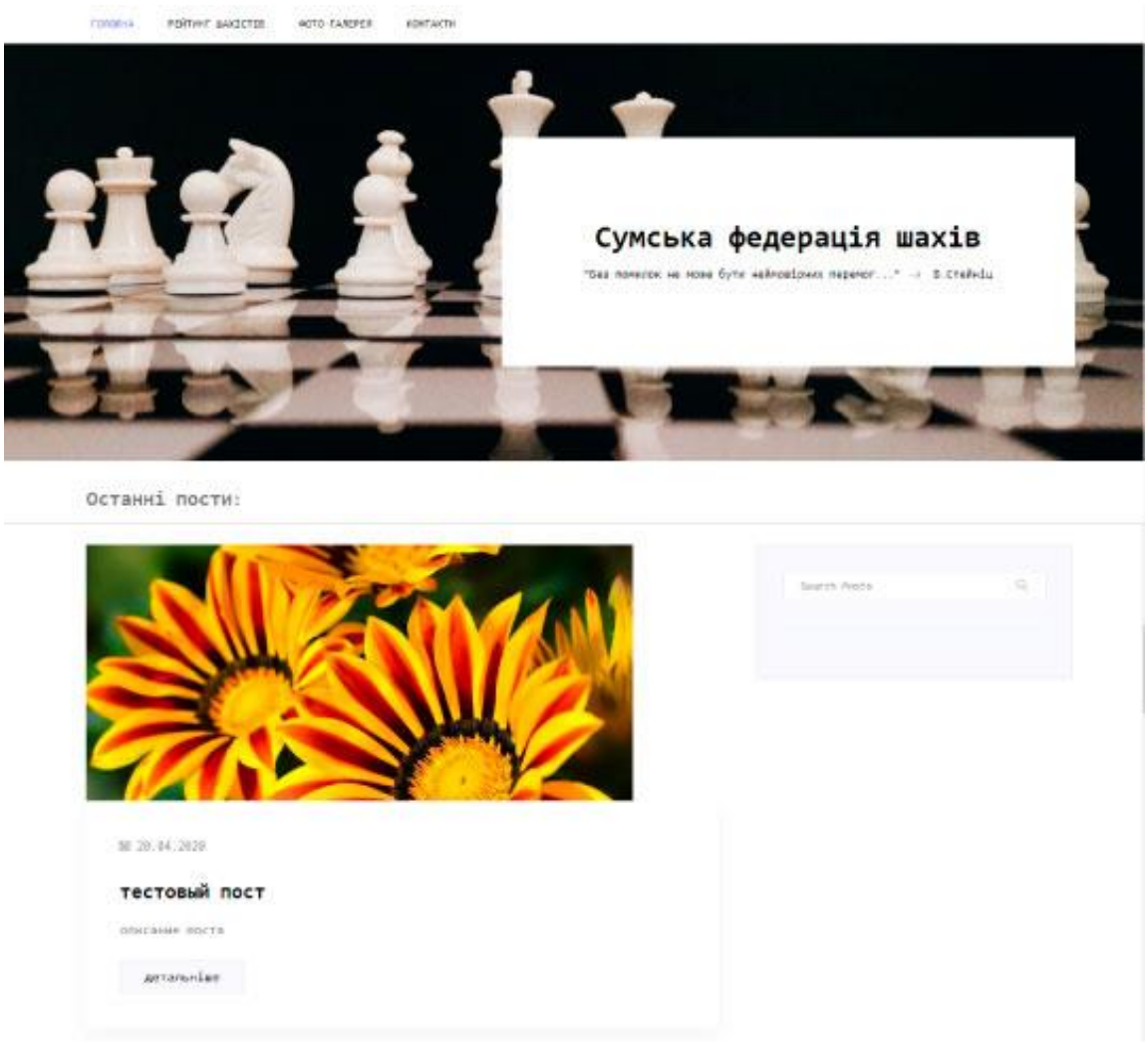


Рисунок 3.8 – основна сторінка

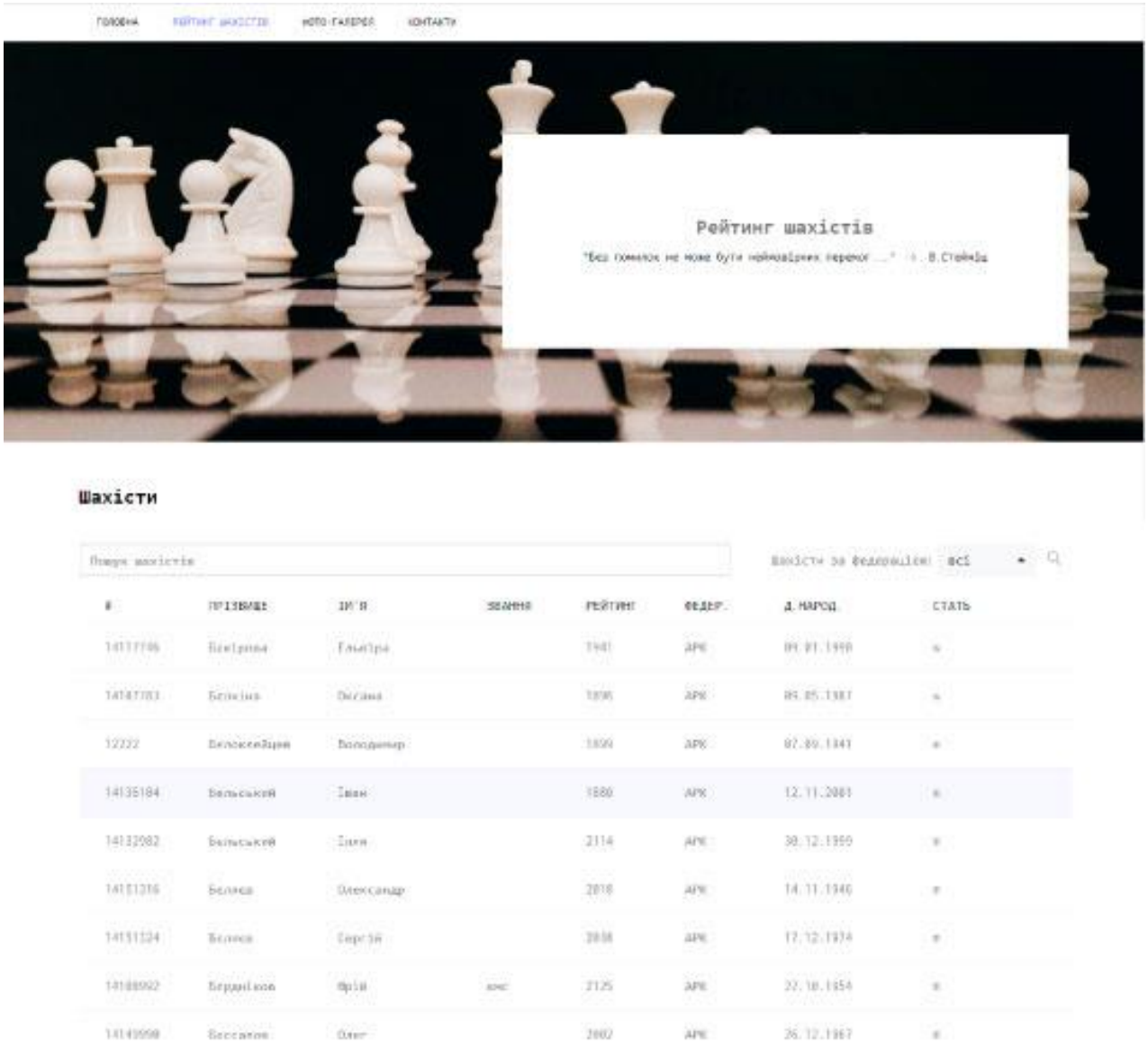


Рисунок 3.9 – сторінка з рейтингами шахістів

Рисунок 3.10 – зміна рейтингу шахіста

Висновки

Виконавши роботу, ми зробили огляд подібних веб-сайтів, де їх розглянули з боку дизайну, швидкодії та структури, тобто наповнення. Огляд існуючих рішень допоміг у визначенні розділів та функціоналу, який повинен обов’язково реалізований на веб-сайті шахової федерації.

Також був виконаний огляд інструментів для реалізації веб-сайту. Було прийняте рішення використовувати веб-фреймворк для розробки, оскільки деякі модулі веб-сайту не є стандартними завданнями.

Було зібрано дані про зміну рейтингів шахістів за допомогою selenium web driver та створено інтерфейс для відображення даних на веб-сайті. Додатково на сайті було реалізовано функціонал блогу.

Список літератури

1. Вікіпедія. Framework [Електронний ресурс] : [Веб-сайт] - Режим доступу: https://en.wikipedia.org/wiki/Framework
2. Вікіпедія. CMS [Електронний ресурс] : [Веб-сайт]. - Режим доступу: https://en.wikipedia.org/wiki/Content_management_system
3. Використовування фреймворку та CMS [Електронний ресурс] : [Веб-сайт]. - Режим доступу: https://qna.habr.com/q/384731
4. Документація з Bootstrap 4 [Електронний ресурс] : [Веб-сайт]. - Режим доступу: https://getbootstrap.com/docs/4.1/getting-started/introduction/
5. Огляд Material Design [Електронний ресурс] : [Веб-сайт]. - Режим доступу: https://material.io/design
6. Документація з фреймворку Javalin [Електронний ресурс] : [Веб-сайт]. - Режим доступу: https://javalin.io/documentation
7. Огляд JSP [Електронний ресурс] : [Веб-сайт]. - Режим доступу: https://metanit.com/java/javaee/3.1.php
8. Вікіпедія java [Електронний ресурс] : [Веб-сайт]. - Режим доступу: https://uk.wikipedia.org/wiki/Java
9. Порівняльна характеристика СУБД [Електронний ресурс] : [Веб-сайт]. - Режим доступу: https://devacademy.ru/article/sqlite-vs-mysql-vs-postgresql/
10. PostgreSQL vs MySQL [Електронний ресурс] : [Веб-сайт]. - Режим доступу: https://habr.com/ru/company/mailru/blog/248845/
11. Трехтировое приложение [Електронний ресурс] : [відео]. – Режим доступу: https://www.youtube.com/watch?v=Usv6zV3w4uY&t=410s
12. Разбор кода DAO Layer на Java [Електронний ресурс] : [відео]. – Режим доступу: https://www.youtube.com/watch?v=flMFICpPfcI&t=1538s
13. stackoverflow.com [Електронний ресурс] : [Веб-сайт]. – Режим доступу: https://stackoverflow.com

Додаток 1. Практична реалізація БД

```
CREATE TABLE title (
  id SERIAL  NOT NULL ,
  name VARCHAR      ,
PRIMARY KEY(id));

CREATE TABLE post (
  id SERIAL  NOT NULL ,
  title VARCHAR    ,
  description VARCHAR    ,
  text TEXT    ,
  image VARCHAR    ,
  date DATE      ,
PRIMARY KEY(id));

CREATE TABLE user_ (
  id SERIAL  NOT NULL ,
  login VARCHAR    ,
  password_ VARCHAR      ,
PRIMARY KEY(id));

CREATE TABLE federation (
  id SERIAL  NOT NULL ,
  name VARCHAR      ,
PRIMARY KEY(id));

CREATE TABLE player (
  id SERIAL  NOT NULL ,
```

```
  federation_id INTEGER   NOT NULL ,
  name VARCHAR    ,
  surname VARCHAR    ,
  title VARCHAR    ,
  rating INTEGER    ,
  birth DATE    ,
  gender CHAR    ,
  valid BOOL      ,
PRIMARY KEY(id)  ,
  FOREIGN KEY(federation_id)
    REFERENCES federation(id));

CREATE INDEX player_FKIndex1 ON player (federation_id);
CREATE INDEX IFK_Rel_01 ON player (federation_id);

CREATE TABLE changes (
  date SERIAL  NOT NULL ,
  player_id INTEGER   NOT NULL ,
  title_id INTEGER   NOT NULL ,
  rating INTEGER    ,
  games INTEGER      ,
PRIMARY KEY(date, player_id)    ,
  FOREIGN KEY(player_id)
    REFERENCES player(id),
  FOREIGN KEY(title_id)
    REFERENCES title(id));
CREATE INDEX changes_FKIndex1 ON changes (player_id);
CREATE INDEX changes_FKIndex2 ON changes (title_id);
CREATE INDEX IFK_Rel_03 ON changes (player_id);
CREATE INDEX IFK_Rel_04 ON changes (title_id);
```

Додаток 2. Лістинг програми

Application.java

```
package project.chess;

import io.javalin.Javalin;
import org.apache.log4j.Logger;
import project.chess.controller.MainController;
import project.chess.model.ModelException;
import project.chess.model.util.PropertyDB;

public class Application {
   private static Logger logger = Logger.getLogger(Application.class);

   public static void main(String[] args){

       logger.info("Try to read db properties");
       PropertyDB propertiesDB = new PropertyDB("config/db.properties");

       logger.info("START");
       Javalin app =Javalin.create();
       app.start(7000);

       app.config.addStaticFiles("pages");

       MainController mainController = new MainController(app);
       mainController.startApp();

   }
}
```

BaseController.java

```
package project.chess.controller;

/**Base interface for all the controllers of project*/
public interface BaseController {
   void execute();
}
```

MainController.java

```
package project.chess.controller;

import io.javalin.Javalin;
import org.apache.log4j.Logger;

import java.util.ArrayList;
import java.util.List;

/*** MainClass for controllers: create and execute them */
public class MainController {
   private static Logger logger = Logger.getLogger(MainController.class);
   private Javalin app;
   private List<BaseController> controllers = new ArrayList<>();

   public MainController(Javalin app){
       this.app = app;
```

```
    }

    public void startApp(){
        app.get("/", context -> {
            logger.info("render: index.html");
            context.render("pages/index.html");
        });

        try{
            createControllers();
            executeControllers();
        }catch (Exception ex){
            logger.error("Cannot create and execute controller", ex);
        }
    }

    public void createControllers(){
        controllers.add(new PlayerController(app));
        controllers.add(new FederationController(app));
        controllers.add(new PostController(app));
        controllers.add(new LoginController(app));
    }

    public void executeControllers(){
        for (BaseController controller: controllers)
            controller.execute();
    }
}
```

PostController.java

```
package project.chess.controller;

import io.javalin.Javalin;
import io.javalin.http.UploadedFile;
import org.apache.log4j.Logger;
import project.chess.model.PostManager;
import project.chess.model.PostManagerImpl;
import project.chess.model.entity.Post;
import project.chess.model.util.RequestUtil;
import project.chess.model.util.Validation;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.time.LocalDate;
import java.util.List;

public class PostController implements BaseController {
    private static Logger logger = Logger.getLogger(PostController.class);
    private Javalin app;

    public PostController(Javalin app){
        this.app = app;
    }

    @Override
    public void execute() {
        app.post("api/posts", context -> {
            String offsetStr = context.formParam("offset");
            String limitStr = context.formParam("limit");
            int limit, offset;
```

```
            if (!Validation.isUnsignedNumber(new
String[]{limitStr,offsetStr})){
                context.result("false");
                return;
            }
            limit = Integer.parseInt(limitStr);
            offset = Integer.parseInt(offsetStr);

            PostManager postManager = new PostManagerImpl();
            List<Post> posts = postManager.getPosts(limit, offset);
            context.json(posts);
        });

        app.get("api/posts", context -> {
            String offsetStr = context.queryParam("offset");
            String limitStr = context.queryParam("limit");
            int limit, offset;
            if (!Validation.isUnsignedNumber(new
String[]{limitStr,offsetStr})){
                context.result("false");
                return;
            }
            limit = Integer.parseInt(limitStr);
            offset = Integer.parseInt(offsetStr);

            PostManager postManager = new PostManagerImpl();
            List<Post> posts = postManager.getPosts(limit, offset);
            context.json(posts);
        });

        app.post("api/posts/id", context -> {
            String postId = context.formParam("id");
            int id;
            if (!Validation.isUnsignedNumber(postId)){
                context.result("false");
                return;
            }
            id = Integer.parseInt(postId);

            PostManager postManager = new PostManagerImpl();
            Post post = postManager.getPost(id);
            if(post == null){
                context.result("{}");
                return;
            }
            System.out.println(postId);
            System.out.println(post);
            context.json(post);
        });

        app.post("api/posts/title", context -> {
            String offsetStr = context.formParam("offset");
            String limitStr = context.formParam("limit");
            String title = context.formParam("title");
            int limit, offset;
            if (!Validation.isUnsignedNumber(new
String[]{limitStr,offsetStr})){
                context.result("false");
                return;
            }
            limit = Integer.parseInt(limitStr);
            offset = Integer.parseInt(offsetStr);
```

```
            PostManager postManager = new PostManagerImpl();
            List<Post> posts = postManager.getPosts(title, limit, offset);
            context.json(posts);
        });

        app.post("api/posts/create", context -> {
            if (RequestUtil.getSessionCurrentUser(context) == null){
                context.result("User is not authorized");
            }
            //TODO сделать контроллер для создания поста
            String title  = context.formParam("title");
            String description = context.formParam("description");
            String text = context.formParam("text");
            LocalDate dateCreation = LocalDate.now();
            UploadedFile image = context.uploadedFile("image");

            PostManager postManager = new PostManagerImpl();
            postManager.savePost(title, description, text, image,
dateCreation);
        });

        app.post("api/posts/count", context -> {
            PostManager postManager = new PostManagerImpl();
            context.json(postManager.countPosts());
        });

        app.get("/images", ctx  -> {
            String id = ctx.queryParam("id");
            id = id.replaceAll("\"","");
            System.out.println(id);
            try{
               FileInputStream image = new FileInputStream(new File("images/"
+ id));
                ctx.result(image);
            }catch (FileNotFoundException e){
                ctx.status(404);
            }

        });
    }
}
```

PlayerController.java

```
package project.chess.controller;

import io.javalin.Javalin;
import org.apache.log4j.Logger;
import project.chess.model.RatingsManager;
import project.chess.model.RatingsManagerImpl;
import project.chess.model.util.Validation;

public class PlayerController implements BaseController {
   private Logger logger = Logger.getLogger(PlayerController.class);
   private Javalin app;

   public PlayerController(Javalin app){
       this.app = app;
   }

   @Override
```

```
   public void execute() {
       app.post("api/players", context -> {
           String offsetStr = context.formParam("offset");
           String limitStr = context.formParam("limit");
           int limit, offset;
           if (!Validation.isUnsignedNumber(new String[]{limitStr,
offsetStr})){
               context.result("false");
               return;
           }

           limit = Integer.parseInt(limitStr);
           offset = Integer.parseInt(offsetStr);

           RatingsManager ratings = new RatingsManagerImpl();
           context.json(ratings.getAllPlayers(limit, offset));
       });

       app.post("api/changesPlayer", context -> {
           int id;
           String idPlayer = context.formParam("id");
           logger.debug("id player = " + idPlayer);

           if (!Validation.isUnsignedNumber(idPlayer)){
               context.result("[]");
           }

           id = Integer.parseInt(idPlayer);
           RatingsManager ratingsManager = new RatingsManagerImpl();
           context.json(ratingsManager.getChangesPlayer(id));
       });

       app.post("api/players/federation", context -> {
           String offsetStr = context.formParam("offset");
           String limitStr = context.formParam("limit");
           String idFederationStr = context.formParam("idFederation");
           logger.debug("offsetStr = " + offsetStr + ", limitStr = " +
limitStr + ", id federation = " + idFederationStr);
           int limit, offset, idFederation;
           if (!Validation.isUnsignedNumber(new
String[]{limitStr,offsetStr,idFederationStr})){
               logger.warn("Limit or offset or id federation isn't unsigned
number");
               context.result("false");
               return;
           }
           limit = Integer.parseInt(limitStr);
           offset = Integer.parseInt(offsetStr);
           idFederation = Integer.parseInt(idFederationStr);

           RatingsManager ratingsManager = new RatingsManagerImpl();
           if(idFederation == 0){
               context.json(ratingsManager.getAllPlayers(limit, offset));
           }else {
               context.json(ratingsManager.getAllPlayers(idFederation, limit,
offset));
           }
       });

       app.post("api/players/surname", context -> {
           String surname = context.formParam("surname");
```

```
            String offsetStr = context.formParam("offset");
            String limitStr = context.formParam("limit");
            int limit, offset;
            if (!Validation.isUnsignedNumber(new
String[]{limitStr,offsetStr})){
                context.result("false");
                return;
            }
            limit = Integer.parseInt(limitStr);
            offset = Integer.parseInt(offsetStr);

            RatingsManager ratingsManager = new RatingsManagerImpl();
            context.json(ratingsManager.getPlayers(surname.trim(), limit,
offset));
        });
    }
}
```

LoginController.java

```
package project.chess.controller;

import io.javalin.Javalin;
import org.apache.log4j.Logger;
import project.chess.model.UserManager;
import project.chess.model.UserManagerImpl;
import project.chess.model.util.RequestUtil;

import java.util.HashMap;
import java.util.Map;

public class LoginController implements BaseController {
    private Logger logger = Logger.getLogger(LoginController.class);
    private static Map<String, Object> model = new HashMap<>();
    private Javalin app;
    private UserManager userManager;

    public LoginController(Javalin app){
        this.app = app;
        this.userManager = new UserManagerImpl();
    }

    @Override
    public void execute() {
        app.post("user/status", context -> {
            if (RequestUtil.getSessionCurrentUser(context) != null){
                context.json("true");
            }else{
                context.json("false");
            }
        });

        app.post("login", ctx ->  {
            logger.info("logging...");
            model.put("currentUser", RequestUtil.getSessionCurrentUser(ctx));

            if (!userManager.authenticate(ctx.formParam("login"),
ctx.formParam("password"))) {
                model.put("authenticationFailed", true);
                logger.info("login or password isn't correct");
                ctx.result("false");
                return;
```

```
            } else {
                logger.info("user is successful authenticate");
                ctx.sessionAttribute("currentUser", ctx.formParam("login"));
                model.put("authenticationSucceeded", true);
                model.put("currentUser", ctx.formParam("login"));
                //ctx.redirect("admin");
                ctx.result("true");
                return;
            }
        });

        app.post("logout", context ->  {
            context.sessionAttribute("currentUser", null);
            context.sessionAttribute("loggedOut", "true");
            context.redirect("/");
        });

        app.get("admin", context -> {
            String currentUser = RequestUtil.getSessionCurrentUser(context);
            if (currentUser == null){
                context.result("User is not authorized");
                return;
            }
            logger.info("user is admin: " + currentUser);
            context.render("pages/admin.html");
        });

        app.get("login", context -> {
            context.render("pages/login.html");
        });
    }

}
```

FederationController.java

```
package project.chess.controller;

import io.javalin.Javalin;
import org.apache.log4j.Logger;
import project.chess.model.RatingsManager;
import project.chess.model.RatingsManagerImpl;

public class FederationController implements BaseController{
   private static Logger logger =
Logger.getLogger(FederationController.class);
   private Javalin app;

   public FederationController(Javalin app){this.app = app;}

   public void execute() {
        app.post("api/federations", context -> {
            RatingsManager ratingsManager = new RatingsManagerImpl();
            context.json(ratingsManager.getAllFederations());
        });
    }
}
```

ChangePlayer.java

```
package project.chess.model.entity.subEntity;

import com.fasterxml.jackson.annotation.JsonProperty;
```

```
import java.time.LocalDate;

public class ChangePlayer {
   @JsonProperty("date")
   private LocalDate date;
   @JsonProperty("playerId")
   private int playerId;
   @JsonProperty("title")
   private String title;
   @JsonProperty("rating")
   private int rating;
   @JsonProperty("games")
   private int games;

   public ChangePlayer(LocalDate date, int playerId, String title, int
rating, int games) {
       this.date = date;
       this.playerId = playerId;
       this.title = title;
       this.rating = rating;
       this.games = games;
   }

   public int getPlayerId() {
       return playerId;
   }

   public int getRating() {
       return rating;
   }

   public LocalDate getDate() {
       return date;
   }

   public String getTitle() {
       return title;
   }

   public int getGames() {
       return games;
   }
}

Federation.java
package project.chess.model.entity;

import com.fasterxml.jackson.annotation.JsonProperty;

public class Federation {
   @JsonProperty("id")
   private int id;
   @JsonProperty("name")
   private String name;

   public Federation(int id, String name) {
       this.id = id;
       this.name = name;
   }
```

```
}
Player.java
package project.chess.model.entity;

import com.fasterxml.jackson.annotation.JsonProperty;
import org.apache.log4j.Logger;
import project.chess.dao.DAOException;
import project.chess.dao.DaoFactory;
import project.chess.dao.PlayerDao;

import java.time.LocalDate;
import java.util.ArrayList;
import java.util.List;

public class Player {
   private static Logger logger = Logger.getLogger(Player.class);

   @JsonProperty("id")
   private int id;
   @JsonProperty("name")
   private String name;
   @JsonProperty("surname")
   private String surname;
   @JsonProperty("title")
   private String title;
   @JsonProperty("currentRating")
   private int currentRating;
   @JsonProperty("federation")
   private String federation;
   @JsonProperty("birth")
   private LocalDate birth;
   @JsonProperty("gender")
   private char gender;
   @JsonProperty("valid")
   private boolean valid;

   private DaoFactory daoFactory;
   private PlayerDao playerDao;

   public Player(int id, String name, String surname, String title, int
currentRating, String federation, LocalDate birth, char gender, boolean
valid) {
       this.id = id;
       this.name = name;
       this.surname = surname;
       this.title = title;
       this.currentRating = currentRating;
       this.federation = federation;
       this.birth = birth;
       this.gender = gender;
       this.valid = valid;
   }

   @Override
   public String toString() {
       return "Player{" +
               "id=" + id +
               ", name='" + name + '\'' +
               ", surname='" + surname + '\'' +
               ", title='" + title + '\'' +
               ", currentRating=" + currentRating +
```

```
                ", federation='" + federation + '\'' +
                ", birth=" + birth +
                ", gender=" + gender +
                ", valid=" + valid +
                '}';
    }

    public int getId() {
        return id;
    }

    public String getName() {
        return name;
    }

    public String getSurname() {
        return surname;
    }

    public String getTitle() {
        return title;
    }

    public int getCurrentRating() {
        return currentRating;
    }

    public String getFederation() {
        return federation;
    }

    public LocalDate getBirth() {
        return birth;
    }

    public char getGender() {
        return gender;
    }

    public boolean isValid() {
        return valid;
    }
}
```

PlayerRating.java

```
package project.chess.model.entity;

import com.fasterxml.jackson.annotation.JsonProperty;
import project.chess.model.entity.subEntity.ChangePlayer;

import java.time.LocalDate;
import java.util.List;

public class PlayerRating {
    @JsonProperty("id")
    private int id;
    @JsonProperty("minRating")
    private int minRating;
    @JsonProperty("maxRating")
    private int maxRating;
    @JsonProperty("dates")
```

```
   private List<LocalDate> dates;
   @JsonProperty("ratings")
   private int[] ratings;
   @JsonProperty("titles")
   private String[] titles;
   @JsonProperty("games")
   private int[] games;

   public PlayerRating(int id, int minRating, int maxRating, List<LocalDate>
dates, int[] ratings, String[] titles, int[] games) {
       this.id = id;
       this.minRating = minRating;
       this.maxRating = maxRating;
       this.dates = dates;
       this.ratings = ratings;
       this.titles = titles;
       this.games = games;
   }
}
Post.java
package project.chess.model.entity;

import com.fasterxml.jackson.annotation.JsonProperty;

import java.time.LocalDate;

public class Post {
   @JsonProperty("id")
   private int id;
   @JsonProperty("title")
   private String title;
   @JsonProperty("description")
   private String description;
   @JsonProperty("text")
   private String text;
   @JsonProperty("image")
   private String image;
   @JsonProperty("dateCreation")
   private LocalDate dateCreation;

   public Post(int id, String title, String description, String text, String
image, LocalDate dateCreation) {
       this.id = id;
       this.title = title;
       this.description = description;
       this.text = text;
       this.image = image;
       this.dateCreation = dateCreation;
   }

   public int getId() {
       return id;
   }

   public String getImage() {
       return image;
   }

   public String getTitle() {
       return title;
   }
```

```
    public String getDescription() {
        return description;
    }

    public String getText() {
        return text;
    }

    public LocalDate getDateCreation() {
        return dateCreation;
    }
}
User.java
package project.chess.model.entity;

public class User {
    private int id;
    private String login;
    private String password;

    public User(int id, String login, String password) {
        this.id = id;
        this.login = login;
        this.password = password;
    }


    public int getId() {
        return id;
    }

    public String getLogin() {
        return login;
    }

    public String getPassword() {
        return password;
    }
}
PropertyDB.java
package project.chess.model.util;

import org.apache.log4j.Logger;
import project.chess.model.ModelException;

import java.io.FileReader;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

public class PropertyDB {
    private Logger logger = Logger.getLogger(PropertyDB.class);
    public static Map<String, String> dbConfig = new HashMap<>();

    public PropertyDB(String pathFile){
        StringBuilder str = new StringBuilder();
        try (FileReader fileReader = new FileReader(pathFile)) {
            logger.info("Reading db properties");
```

```
            int count;
            while ((count = fileReader.read()) != -1) {
                str.append((char) count);
            }
        } catch (IOException e) {
            //TODO нужно ли пробрасывать исключение?
            logger.error("Cannot read db properties", e);
            //throw new ModelException("Cannot read db properties", e);
        }

        String[] properties = str.toString().split("\n");
        String[] buffer;
        for (String property : properties) {
            buffer = property.split("=");
            dbConfig.put(buffer[0].trim(), buffer[1].trim());
        }
    }
}
```

RandomString.java

```
package project.chess.model.util;

import java.util.Random;

/**
* Generate random string.
* @author Anton Kudryavtsev
*/
public class RandomString {
   private int length;

   /**
    * Constructor.
    * @param length length of string
    */
   public RandomString(int length) {
       this.length = length;
   }

   /**
    * Generate random string.
    * @return random string
    */
   public String generate() {
       int leftLimit = 97; // letter 'a'
       int rightLimit = 122; // letter 'z'
       int targetStringLength = length;
       Random random = new Random();
       StringBuilder buffer = new StringBuilder(targetStringLength);
       for (int i = 0; i < targetStringLength; i++) {
           int randomLimitedInt = leftLimit + (int)
                   (random.nextFloat() * (rightLimit - leftLimit + 1));
           buffer.append((char) randomLimitedInt);
       }
       return buffer.toString();
   }
}
```

RequestUtil.java

```
package project.chess.model.util;
import io.javalin.http.Context;

/**
* <p> Класс для работы данными сесии</p>
```

```
* */
public class RequestUtil {
    /**
     * Функция получения текущего пользователя
     * @param ctx объект context
     * @return возвращает текущего пользователя
     * */
    public static String getSessionCurrentUser(Context ctx) {
        return (String) ctx.sessionAttribute("currentUser");
    }

    /**
     * Функция удаления аттрибута выхода пользователя
     * @param ctx объект context
     * @return возвращает результат удаления
     * */
    public static boolean removeSessionAttrLoggedOut(Context ctx) {
        String loggedOut = ctx.sessionAttribute("loggedOut");
        ctx.sessionAttribute("loggedOut", null);
        return loggedOut != null;
    }

    /**
     * Функция для удаления перенаправления пользователя
     * @param ctx объект context
     * @return возвращает значение перенаправления
     * */
    public static String removeSessionAttrLoginRedirect(Context ctx) {
        String loginRedirect = ctx.sessionAttribute("loginRedirect");
        ctx.sessionAttribute("loginRedirect", null);
        return loginRedirect;
    }
}
```

StoreImage.java

```
package project.chess.model.util;

import com.google.common.io.Files;
import io.javalin.http.UploadedFile;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import project.chess.model.ModelException;

import java.io.File;
import java.io.IOException;
import java.io.InputStream;

public class StoreImage {
    private Logger logger = LoggerFactory.getLogger(StoreImage.class);
    private UploadedFile uploadedFile;

    public StoreImage(UploadedFile uploadedFile) {
        this.uploadedFile = uploadedFile;
    }

    public String store() throws ModelException {
        String path = "unnamed";

        try {
            InputStream initialStream = uploadedFile.getContent();
            byte[] buffer = new byte[initialStream.available()];
            initialStream.read(buffer);
```

```
            path = generatePath();
            File targetFile = new File("images/" + path);
            Files.write(buffer, targetFile);
            logger.info("Storing image executing successfully. Image path - "
+ path);
        } catch (IOException e) {
            logger.error("Some problem with storing image - " +
e.getMessage());
            throw new ModelException("Some problem with storing image - " +
e.getMessage(), e);
        }

        return path;
    }

    private static String generatePath() {
        return new RandomString(100).generate();
    }
}
```

Validation.java

```
package project.chess.model.util;

import org.apache.log4j.Logger;

public class Validation {
    private static Logger logger = Logger.getLogger(Validation.class);

    public static boolean isUnsignedNumber(String[] numbers){
        for (int i = 0; i < numbers.length; i++) {
            if (!isUnsignedNumber(numbers[i])){
                return false;
            }
        }

        return true;
    }

    public static boolean isUnsignedNumber(String str){
        int offset;
        if (str == null){
            logger.warn("string is null");
            return false;
        }

        try{
            offset = Integer.parseInt(str);
        }catch (NumberFormatException e){
            logger.warn("string is not a number");
            return false;
        }

        if (offset < 0){
            return false;
        }

        return true;
    }
}
```

ModelException.java

```
package project.chess.model;

public class ModelException extends Exception{

   public ModelException(){ super();}

   public ModelException(String message){super(message);}

   public ModelException(String message, Throwable cause){super(message,
cause);}
}
```

PostManager.java

```
package project.chess.model;

import io.javalin.http.UploadedFile;
import project.chess.model.entity.Post;

import java.time.LocalDate;
import java.util.List;

public interface PostManager {
   List<Post> getPosts(int limit, int offset);

   List<Post> getPosts(String title, int limit, int offset);

   Post getPost(int id);

   void savePost(String title, String description, String text, UploadedFile
image, LocalDate date) throws ModelException;

   int countPosts();



}
```

PostManagerImpl.java

```
package project.chess.model;

import io.javalin.http.UploadedFile;
import org.apache.log4j.Logger;
import project.chess.dao.DAOException;
import project.chess.dao.DaoFactory;
import project.chess.dao.PostDao;
import project.chess.model.entity.Post;
import project.chess.model.util.StoreImage;

import java.time.LocalDate;
import java.util.ArrayList;
import java.util.List;

public class PostManagerImpl implements PostManager{
   private static Logger logger = Logger.getLogger(PostManagerImpl.class);
   private DaoFactory daoFactory;
   private PostDao postDao;

   public PostManagerImpl(){
       daoFactory = DaoFactory.getInstance();
       postDao = daoFactory.getPostDao();
```

```
    }

    public List<Post> getPosts(int limit, int offset){
        List<Post> posts = new ArrayList<>();
        try{
            posts = postDao.getAll(limit, offset);
        }catch (DAOException e){
            logger.error("Posts not found", e);
        }

        return posts;
    }

    public List<Post> getPosts(String title, int limit, int offset){
        List<Post> posts = new ArrayList<>();
        try{
            posts = postDao.searchByTitle(title, limit, offset);
        }catch (DAOException e){
            logger.warn("Posts not found");
        }

        return posts;
    }

    public Post getPost(int id){
        Post post = null;
        try{
            post = postDao.getById(id);
        }catch (DAOException e){
            logger.warn("Post by id not found");
        }
        return post;
    }

    public int countPosts(){
        int count = -1;
        try{
            count = postDao.countPosts();
        }catch (DAOException e){
            logger.warn("Number of posts not get");
        }
        return count;
    }

    public void savePost(String title, String description, String text,
UploadedFile image, LocalDate date) throws ModelException {
        //TODO если картинка не загружена, то как пользователь об этом узнает?
эта функция ничего же не возвращает
        // надо, что-то придумать: 1)пробрасывать исключение 2)вместо возвра-
щаемого типа void сделать boolean/
        StoreImage storeImage = new StoreImage(image);
        String imageTitle = storeImage.store();
        if (imageTitle.equals("unnamed")){
            return;
        }
        try{
            postDao.create(new Post(0, title, description, text, imageTitle,
date));
        }catch (DAOException e){
            logger.warn("Cannot create post");
        }
```

```
    }
}
```

RatingsManager.java

```
package project.chess.model;

import project.chess.model.entity.PlayerRating;
import project.chess.model.entity.subEntity.ChangePlayer;
import project.chess.model.entity.Federation;
import project.chess.model.entity.Player;

import java.util.List;

public interface RatingsManager {

    List<Player> getAllPlayers();

    List<Player> getAllPlayers(int limit, int offset);

    List<Player> getAllPlayers(int idFederation, int limit, int offset);

    List<Federation> getAllFederations();

    PlayerRating getChangesPlayer(int id);

    List<Player> getPlayers(String surname, int limit, int offset);
}
```

RatingsManagerImpl.java

```
package project.chess.model;

import org.apache.log4j.Logger;
import project.chess.dao.DAOException;
import project.chess.dao.DaoFactory;
import project.chess.dao.FederationDao;
import project.chess.dao.PlayerDao;
import project.chess.model.entity.PlayerRating;
import project.chess.model.entity.subEntity.ChangePlayer;
import project.chess.model.entity.Federation;
import project.chess.model.entity.Player;

import java.time.LocalDate;
import java.util.ArrayList;
import java.util.List;

public class RatingsManagerImpl implements RatingsManager{
    private static Logger logger = Logger.getLogger(RatingsManagerImpl.class);
    private List<Player> players = new ArrayList<>();

    private DaoFactory daoFactory;
    private PlayerDao playerDao;
    private FederationDao federationDao;

    public RatingsManagerImpl(){
        logger.trace("Getting daoFactory instance");
        daoFactory = DaoFactory.getInstance();
        logger.trace("Getting playerDao");
        playerDao = daoFactory.getPlayerDao();
        logger.trace("Getting federationDao");
        federationDao = daoFactory.getFederationDao();
    }
```

```
    public List<Player> getAllPlayers(){
        return getAllPlayers(1000000, 0);
    }

    public List<Player> getAllPlayers(int limit, int offset){
        List<Player> players = new ArrayList<>();
        try{
            players = playerDao.getAll(limit, offset);
        }catch (DAOException e){
            //TODO нужно ли здесь добавлять stacktrace??? и какой должен быть
уровень логирования
            logger.warn("Players not found");
        }

        return players;
    }

    public List<Player> getAllPlayers(int idFederation, int limit, int
offset){
        List<Player> players = new ArrayList<>();
        try{
            players = playerDao.allByFederation(idFederation, limit, offset);
        }catch (DAOException e){
            logger.warn("Players of federation not found");
        }

        return players;
    }

    public List<Federation> getAllFederations(){
        List<Federation> federations = new ArrayList<>();
        try{
            federations = federationDao.getAll();
        } catch (DAOException e) {
            logger.warn("Federations not found");
        }
        return federations;
    }

    public PlayerRating getChangesPlayer(int id){
        List<ChangePlayer> changeList = new ArrayList<>();
        PlayerRating playerRating = null;

        try{
            logger.trace("getting changes rating of player");
            changeList = playerDao.getChangePlayerById(id);

            if (changeList.size() != 0){
                logger.trace("getting ratings, titles, dates array");
                int[] ratings = new int[changeList.size()];
                int[] games = new int[changeList.size()];
                List<LocalDate> dates = new ArrayList<>();
                String[] titles = new String[changeList.size()];

                for (int i = 0; i < changeList.size(); i++) {
                    ratings[i] = changeList.get(i).getRating();
                    games[i] = changeList.get(i).getGames();
                    dates.add(changeList.get(i).getDate());
                    titles[i] = changeList.get(i).getTitle();
                }
```

```
            logger.trace("find max and min rating");
            int[] sortedRating = sortRating(ratings);

            playerRating = new PlayerRating(
                    changeList.get(0).getPlayerId(),
                    sortedRating[0],
                    sortedRating[sortedRating.length-1],
                    dates,
                    ratings,
                    titles,
                    games
            );
        }
    } catch (DAOException e) {
        logger.warn("player's changes not found");
    }

    return playerRating;
}

private int[] sortRating(int[] ratingsPlayer){
    int[] ratings = new int[ratingsPlayer.length];

    logger.trace("Getting all the ratings of player");
    for (int i = 0; i < ratings.length; i++) {
        ratings[i] = ratingsPlayer[i];
    }

    if (ratings.length == 1){
        return ratings;
    }

    logger.trace("sorting ratings ASC. Selection sort");
    for (int min = 0; min < ratings.length-1; min++) {
        int least = min;
        for (int j = min+1; j < ratings.length; j++) {
            if (ratings[j] < ratings[least]){
                least = j;
            }
        }

        int tmp = ratings[min];
        ratings[min] = ratings[least];
        ratings[least] = tmp;
    }
    /*logger.debug("ratings sort: " + ratings.);
    for (int i = 0; i < ratings.length; i++) {
        System.out.println(ratings[i]);
    }*/
    return ratings;
}

public List<Player> getPlayers(String surname, int limit, int offset){
    List<Player> players = null;
    try{
        players = playerDao.allBySurname(surname.trim(), limit, offset);
    }catch (DAOException e){
        logger.warn("players by surname not found");
    }

    return players;
```

```
    }
}
```

UserManager.java

```
package project.chess.model;

public interface UserManager {
   boolean authenticate(String login, String password);
}
```

UserManagerImpl.java

```
package project.chess.model;

import org.apache.log4j.Logger;
import project.chess.dao.DAOException;
import project.chess.dao.DaoFactory;
import project.chess.dao.UserDao;
import project.chess.model.entity.User;

public class UserManagerImpl implements UserManager{
   private static Logger logger = Logger.getLogger(UserManagerImpl.class);
   private DaoFactory daoFactory;
   private UserDao userDao;

   public UserManagerImpl(){
       daoFactory = DaoFactory.getInstance();
       userDao = daoFactory.getUserDao();
   }

   public boolean authenticate(String login, String password){
       User user = null;
       try{
           user = userDao.getUserByLogin(login);
           if (user == null){
               return false;
           }

           if (password.equals(user.getPassword())){
               return true;
           }else {
               return false;
           }

       }catch (DAOException e){
           logger.error("Cannot get user during authentication", e);
       }

       return false;
   }
}
```