**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ**

**ФАКУЛЬТЕТ МЕХАТРОНІКИ ТА КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ**

**КАФЕДРА КОМП'ЮТЕРНИХ НАУК**

# *Дипломна магістерська робота*

на тему

## *«Розробка програмного забезпечення для комп'ютерної гри "Шахи" з використанням платформи Angular»*

Виконав: студент групи МгІТ-2-20

спеціальності 122 Комп'ютерні науки
освітньої програми Комп'ютерні науки
(шифр і назва спеціальності)

Студент Снєсарь А. Р.
(прізвище та ініціали)

Керівник к.т.н., доц. Тетяна Демківська
(прізвище та ініціали)

Рецензент д.т.н., проф. Віктор Чупринка
(прізвище та ініціали)

Київ 2021

# АНОТАЦІЯ

**Снєсарь А.Р. Розробка програмного забезпечення для комп'ютерної гри "Шахи" з використанням платформи Angular.**

Дипломна магістерська робота за спеціальністю 122- «Комп’ютерні науки та технології» – Київський національний університет технологій та дизайну, Київ, 2021 рік.

Досліджено основні інструменти, які використовуються для побудови вебдодатків з допомогою платформи Angular, зокрема методи вводу-виводу інформації, способи передачі даних між компонентами додатку та принципи побудови модульної архітектури.

В ході даної роботи було розроблено вебдодаток гра «Шахи» з використанням бібліотек Angular Material та Bootstrap.

Ключові слова: вебдодаток, платформа, фреймворк, Angular, бібліотека, Material, івенти, шахи, гра, івенти.

# A N N O T A T I O N

**Snesar A.R. Development of software for the computer game "Chess" using the Angular platform.**

Master's thesis in the specialty 122 - "Computer Science and Technology" - Kyiv National University of Technology and Design, Kyiv, 2021.

The main tools used to build web applications using the Angular platform are studied, including input-output methods, methods of data transfer between application components and the principles of building a modular architecture.

In the course of this work, a web application (in this case a game) was developed using the Angular Material and Bootstrap libraries.

Keywords: web application, platform, framework, Angular, library, Material, events, chess, game, events.

# ЗМІСТ

# ВСТУП

В умовах сьогодення цифрова трансформація стає основою всіх сфер життєдіяльності суспільства. В умовах глобалізації, інформаційні технології — це одночасно величезний ринок та індустрія, а також платформа ефективності й конкурентоспроможності всіх інших ринків та індустрій. Рівень інтеграції інформаційних технологій в повсякденне життя звичайних людей значно виріс, та продовжує нарощувати темпи розповсюдження. Інформаційні технології перетворилися на життєво важливий інструмент взаємодії на усіх рівнях, від користувача домашнього комп'ютера до країн чи міжнародних організацій.

Інтенсивний розвиток інтернету, удосконалення мов програмування та технологій, розширення сфери взаємодії браузерів – усе це приводить до все більшої необхідності підтримки програмних реалізацій на різних платформах. Одним з лідерів цього сегменту, можна вважати браузери. Завдяки динамічному, якісному розвитку, все більші функції перекладаються саме на них. Більшість розповсюджених програмних продуктів вже мають свої аналоги у вигляді інтернет сторінок (більшість пакетів MS Office, програми для читання файлів, редагування, розрахункові програми, ігри та інші).

Важливою сферою людського життя, була й залишається (і все ще можна спостерігати за ростом динаміки розвитку) сфера розваг, зокрема, ігри. Роль ігор важко переоцінити; вони поєднують в собі декілька функцій, серед них виховна, розважальна, пізнавальна та інші. Хотілося б звернути увагу на ігри саме в середовищі інтерактивних інформаційних систем (комп'ютери, смартфони, планшети, ноутбуки, консолі і тому подібне). Можна стверджувати, що саме цей сегмент IT ринку розвивається найшвидше. Комп'ютерні ігри створили свою, величезну спільноту, так звану субкультуру, з власними звичаями та правилами. Відомі компанії стають об'єктами постійної уваги, а їх продукти стають класикою, та знаходять шанувальників в усіх кутках земного шару. Протягом останніх років показник, що характеризує рівень попиту на комп'ютерні ігри, постійно зростає. Це дозволяє зробити висновок про те, що розробка

кросплатформенного розважального програмного продукту актуальна і доцільна тема дослідження.

# Розділ 1. Теоретична частина. Основні принципи функціонування вебтехнологій

## 1.1 Постановка задачі

Метою даного дослідження є розробка Angular додатку та ознайомлення з мовами розробки високого рівня, що містить в собі аналіз проблем, які вони вирішують, в першу чергу сучасних тенденцій принципів побудови додатків.

Основною метою дослідницького проекту є розробка вебдодатку (гри) для вебраузера на базі платформи Angular з використанням бібліотек Angular Material та Bootstrap.

Додаток повинен представляти функціонал повноцінної гри:

- Відповідність архітектури повному ігровому циклу;
- Вплив на змінні стану гри повинен викликати динамічне перемальовування (рендерінг) нового стану;
- Динамічна зміна налаштувань та їх збереження навіть після завершення ігрової сесії;
- Зручний і зрозумілий інтерфейс користувача;
- Назви та короткий опис інструментів та підходів, застосованих у додатку.

Крім сучасних методів програмування, існує декілька поширених інструментів для запобігання виникнення синтаксичних помилок в коді, основні види аналізу коду це статичний та динамічний, які також широко використовуються в розробці ігрових проектів.

## 1.2 Аналіз останніх досліджень і публікацій

Особливість поставленого завдання привела до обробки інформації, починаючи з культурного феномену соціуму, створеного в вигаданій ігровій реальності, і закінчуючи роботами, що обговорюють проблематику вебдизайну.

Культурний феномен соціуму, створеного ігровою, вигаданою реальністю (і Інтернету технічної реалізації) схиляє до перегляду не “постіндустріального” обговорення (Д. Белл, Е. Тоффлер), а “мережного”, тому головним об’єктом для дипломного дослідження є роботи, в яких постає проблема соціальних та культурних переваг сучасності, та постмодерну, як часткової ситуації (М. Постер, Н. Б. Маньковська, Ж. Дельоз, Ж. Деррида, Ж. Бодрийяр, Р. Барт, Ю. Кристєва, П. Вірилiо, М.Кастельс, М. Хардт, Ж.-Ф. Ліотар, О. О. Мамалуй, У. Бек, М. Фуко, З. Бауман, В. В. Гусаченко, Ф. Гваттарі, та ін.).

Можна відстежити значні здобутки в області концептуалізації ігор, особливо їх теоретичної частини. Доведеться звернутися до величезної за обсягом кількості літератури, від філософів давньої Греції (Геракліт, Платон) до Ліотара та Бодрийяра, поступово проглядаючи зміщення гри з метафізичних та онтологічних аспектів – до естетичного та антропологічного.

Описи ігрових реалізацій, контрольованих, зазвичай, комп’ютерною мережею, представлені в великій кількості наукових робіт Д.Дойча та Д.Шапіро. Важко оминути письменників-фантастів, дуже часто їх концепції та уявлення служать натхненням для науковців та інженерів (В. Пелевін, У. Гібсон, Мерсі Шеллі Дж. Нун,). Майже одночасно виникла інша течія, яка описує соціокультурну модерну в термінах ігрового. Тема цифровізації соціальної реальності була розглянута в роботах Ж. Бодрийяра, котрий переходить до дещо філософських концепцій, та вводить новий відтінок терміну гіперреальності. Також, цю проблему розглядав Д. Іванов; він спрямував ідею дослідження на соціологічний аналіз створених нині віртуальних соціальних інститутів, тим самим вказавши на точки взаємовпливу між соціумами. Феномен цифрової повсякденності також більше звертається до “антропологічного” концепту – перетворення матеріального, а також статус суб'єкта чи об’єкта у вигаданій реальності.

Варто відзначити величезний вплив та значимість цих наукових розробок. Дослідження соціальної, вигаданої, навіть «віртуальної» реальності мають,

зазвичай, партикулярний характер; вони не роблять підсумку досвіду суб'єкта, навіть обговорюючи культурні практики. Комунікації з використанням інтернет технологій також є наслідком єдиних трансформаційних соціальних процесів

Вебдизайн, в свою чергу, є не тільки інструментом моделювання простору культури, а й принципово новою мовою, що впливає на мислення і культуру користувачів. У вирішенні даної проблеми було виявлено, синтезовано і використано кілька підходів: семантико-семіотичний напрямок в культурології, культультурологічні аспекти вивчення вебдизайну в мистецтвознавстві та теорії дизайну. Використовуючи класичний семіотичний підхід, в тому числі методологію Ю. Лотмана і Ю. Степанова, проаналізовано інтернет культуру як семіотичну систему. Ґрунтуючись на теоріях інформаційно-семіотичного напрямку, можна запропонувати вивчати проектні властивості вебдизайну, як суміжно тему до його загальних властивостей. З метою вивчення вебдизайну як об'єкту, що складно піддається структуризації; та динамічної мови інтернет культури були застосовані некласичні напрямки, що обґрунтовують теорію про провідну роль мови в формуванні мислення. Згідно з А. Вежбицькой використаний синтез ідей класичного мовознавства характеризується гіпотезою відносності Сепіра-Уорфа. Проаналізовано культурологічні аспекти в розумінні вебдизайну в дослідженнях мистецтвознавців: Д. Бородаєва, Е. Панофского. Використаний трирівневий метод Е. Панофского, було доповнено для аналізу сайтів як креолізованних текстів. Вивчення тенденцій формоутворення в сучасному дизайні, запропоноване А. Лаврентьєвим, використано для вивчення проектних властивостей вебдизайну. Для соціокультурного аналізу аудиторії сайтів варто використовувати дані практикоорієнтованих праць вебдизайнерів.

Також, аналіз робіт щодо даної теми привів до висновку, щодо значної розбіжності між практикою та теорією. Більшість праць, що переважають у сучасному науковому обігу, стосуються теоретичних та загально практичних тем. Серед основоположних досліджень можна виділити роботи В. Даниленка, В. Ляхова, С. Сєрова, В. Плишевського, В. Глазичева, В. Сеньковського, Є.

Черневич. Вони досить конкретно намагаються продемонструвати характеристику якісних змін у дослідженнях процесу дизайну загалом, та надають відповідно матеріально-теоретичну базу. Особливості обраного ракурсу розгляду питання наштовхує на звернення до довідників традиційної галузі графічного дизайну, в тому числі типографії чи мистецтва книги. Варто зазначити, що мультимедіа зазвичай класифікують не як результат цифрової революції, а скоріше цифровим представленням ідей; присутні у мистецтві століттями, вони спонукають до розгляду основних напрямків вивчення проблемних питань. Як наслідок, проводячи аналіз сучасного стану технологій та практик розробки вебдодатків, варто враховувати попередній досвіт презентації та подання інформації.

## 1.3 Історія розвитку вебтехнологій

### Де народилися вебтехнології

Всесвітня павутина була винайдена у 1989 році Тімом Бернерсом-Лі, під час його роботи в Європейській лабораторії з ядерних досліджень. Спочатку, інтернет був створений для задоволення власних потреб вченого – автоматичний обмін інформацією з іншими вченими інститутів та університетів по всьому світі.

Зараз ЦЕРН не ізольований; навпаки – він є центром величезної спільноти, що включає в себе більш ніж 17000 вчених з різних країн. Звісно, вони проводять певний час на сайті ЦЕРНу, але в більшості все ж працюють в лабораторіях власних університетів. Тому безпечні комунікації є досить великим пріоритетом. Основна ідея WWW: об’єднати комп’ютерні технології, мережу передачі данних та гіпертекст у просту та потужну глобальну систему.

### Як почався інтернет

Тім Бернерс-Лі написав першу пропозицію щодо Всесвітньої павутини у березні 1989 року, а свою другу пропозицію у травні 1990 року. Найперша пропозиція щодо Всесвітньої павутини була написана Тімом Бернерсом-Лі в березні 1989 року. Друга була написана ним же разом з інженером Робертом Каййо дещо пізніше: в травні 1990 року. Вона була оформлена як пропозиція

щодо оптимізації управління інформацією. Це дало змогу окреслити основні концепції та визначити важливі терміни, що знаходяться в основі поняття Інтернет. У документі описується "гіпертекстовий проєкт" під назвою "WorldWideWeb", в якому "павутину" "гіпертекстових документів" можна було переглядати "браузерами". До кінця 1990 року Тімом Бернерсом-Лі був створений перший вебсервер і браузер в ЦЕРНі, для демонстрації своїх ідей. Він розробив код для свого вебсервера на комп'ютері NeXT. Щоб запобігти випадковому вимкненню, на комп'ютері була червоною фарбою написана від руки етикетка: "Цей апарат є сервером. Не вимикайте!!"**.**

Дизайн WWW дозволяв легкий доступ до наявної інформації у вигляді ранньої вебсторінки, пов'язаної з інформацією, корисною для вчених CERN (наприклад, телефонної книги CERN та посібників із використання центральних комп'ютерів CERN). Система пошуку спиралася на ключові слова – у перші роки не було пошукових систем. Оригінальний веббраузер Бернерса-Лі, що працював на комп'ютерах NeXT, показував його бачення та мав багато функцій сучасних веббраузерів. Він включав можливість змінювати сторінки безпосередньо в браузері – перша можливість редагування в Інтернеті.

**Інтернет розширюється**

Лише деякі користувачі мали доступ до комп'ютерної платформи NeXT, на якій працював перший вебпереглядач, але незабаром розпочалася розробка простішого вебпереглядача «лінійного режиму», який міг працювати у будь-якій системі. Його написала Нікола Пеллоу під час студентської роботи в ЦЕРН. У 1991 році Бернерс-Лі випустила своє програмне забезпечення WWW. Він включав браузер "лінійний режим", програмне забезпечення вебсервера та бібліотеку для розробників. У березні 1991 року програмне забезпечення стало доступним для колег, які використовують комп'ютери CERN. Кілька місяців потому, у серпні 1991 року, він оголосив про програмне забезпечення WWW у групах новин Інтернету та зацікавленість проєктом поширився по всьому світі.

**Інтернет стає глобальним**

Завдяки зусиллям Поля Кунца та Луїзи Аддіс, перший вебсервер у США з’явився у мережі у грудні 1991 року, знову в лабораторії фізики частинок: Стенфордському лінійному прискорювальному центрі (SLAC) у Каліфорнії. На цьому етапі існувало лише два види браузерів. Однією з них була оригінальна версія розробки, яка була складною, але доступною лише на машинах NeXT. Іншим був браузер "лінійного режиму", який було легко встановити та запустити на будь-якій платформі, але обмежений у потужності та зручності користування. Було зрозуміло, що невелика команда в ЦЕРН не може виконати всю роботу, необхідну для подальшого розвитку системи, тому Бернерс-Лі закликав через Інтернет приєднатися інших розробників. Кілька осіб написали браузери, переважно для X-Window Систем. Помітними серед них були MIDAS Тоні Джонсона з SLAC, Віола Пей Вей від технічного видавця O'Reilly Books та Erwise - від фінських студентів з Гельсінського технологічного університету. На початку 1993 року Національний центр додатків суперкомп'ютерів (NCSA) при університеті штату Іллінойс випустив першу версію своєї «Мозаїки».

## 1.4 Основні сфери впливу вебтехнологій

Технології впливають майже на всі аспекти життя 21 століття, починаючи від ефективності та безпеки транспорту, закінчуючи доступом до продуктів харчування та охорони здоров’я, соціалізації та продуктивності. Потужність Інтернету дозволила глобальним спільнотам формуватись, а ідеям та ресурсам стати більш динамічними. Однак надмірне використання деяких технологій було пов'язане зі зниженням психічного здоров'я, посиленням соціального поділу та занепокоєння щодо конфіденційності.

Ми сприймаємо технології як належне щодня - навіть якщо вони миттєво повідомляють нам останні новини, роблять наше капучино або зв’язують нас з людиною в іншій частині світу. Пандемія коронавірусу зробила нас ще більш залежними від технологій і водночас допомогла подолати виклики року.

## Продвинута комунікація

Однин із засобів, що набув масового поширення за останні кілька років є відео дзвінки. Концепція існувала досить давно, але революція високошвидкісного широкосмугового доступу за доступними цінами привела до того, що тепер легко надсилати та отримувати обсяги даних, необхідних для відео дзвінка.

Відео дзвінки провели останнє десятиліття, повільно проникаючи у повсякденне життя. Врешті, тривала пандемія значно прискорила темпи інтеграції та забезпечила майбутнє відеозвінків як повсякденний спосіб підтримувати зв’язок.

Вони привели до змін не тільки в суспільному житті. Останні роки більшість людей працює із дому, і особисті зустрічі замінили на відео конференції.

## Послаблення приватності

Доступ до всього в Інтернеті значно підвищує рівень зручності, проте також впливає і на рівень безпеки. Кожен рух в Інтернеті залишає цифровий слід і цим, при бажанні, можуть скористатись зловмисники. За безпеку на певній вебсторінці зазвичай відповідають її розробники; також, варто пам'ятати про загрози зі сторони середовища передачі інформації чи машини клієнта.

Звичайно, як і у всьому іншому, технології надають інструменти, що здатні захистити певну частину комплексної системи. Проте варто зауважити, що рівень захисту вимірюється найбільш вразливим сегментом. У 2021 році це є ще більш важливим, адже все більше людей користуються вебтехнологіями не лише для особистих потреб, але й отримують доступ до спільних робочих мереж із власного будинку, де не можна покластися на закриту безпеку фізичного офісу.

## Інтерактивна торгівля

Технології не оминули і фізичні покупки. Завдяки безконтактним карткам та розрахункам по телефону нам не потрібно турбуватися про передачу готівки або введення PIN-коду.

Система торгових точок (POS) стала величезним благом для бізнесу, незалежно від його розміру. POS дає змогу не тільки приймати платежі в електронному вигляді, але й автоматично керувати рівнями запасів, створювати електронні квитанції, керувати схемами лояльності, керувати продажами тощо.

Все більший ринок завойовує онлайн торгівля; більше не потрібно виходити з дому, щоб робити покупки – достатньо планшета, ноутбука чи смартфону, аби отримати доступ до віртуального магазину з будь-якої точки світу, де ми можна придбати практично будь-що.

Технології також демократизували роздрібну торгівлю. Фізична присутність відійшла на другий план; щоб відкрити власний магазин тепер достатньо лише комп'ютера та ідеї.

**Миттєвий доступ до інформації**

Доступ до будь-якої інформації тепер коштує декількох секунд часу та пару кліків. Не так давно потрібно було відвідати бібліотеку, щоб дізнатися більш детальну інформацію про предмет, якщо така взагалі була доступна. Тепер, завдяки досягненням технологій, можна знайти сотні тисяч вебсторінок, присвячених практично будь-чому, починаючи від «в'язання гачком» (Google дає 129 000 000 результатів) до «римської історії» (1 360 000 000 результатів).

Сучасні додатки роблять більшість звичних раніше речей застарілими. Наприклад, GPS – тепер достатньо завантажити відкрити вебсторінку і вибрати найкращий маршрут, який буде укомплектований вказівками та супутниковими знімками; що декілька десятків років назад здавалося неможливим. Існують програми для підприємств, які автоматично маршутують транспортні засоби поряд з інформацією про дорожній рух, погоду, техніку безпеки та правову інформацію. Технологія додатків також спростила навчання, зустрічі, відпочинок та майже все, про що тільки можна задуматись.

Не варто забувати і про фактичні пристрої, на яких працюють усі ці програми. Зростання кількості смартфонів було експоненційним за останнє десятиліття, і щоденні пошуки в Інтернеті на мобільних пристроях зараз

перевершують кількість тих, що знаходяться на портативних або настільних комп’ютерах. Щороку безперебійно вдосконалюються портативні пристрої.

## Віртуальне соціальне життя

Значно вплинуло на суспільне життя впровадження соціальних медіа. Ця індустрія швидко розвивається; зараз її першопрохідці, такі як MySpace та оригінальна ітерація Facebook, вважаються уже застарілими. Такі послуги, як Snapchat, TikTok, Instagram та інші, тепер дають нам уявлення про поведінку та звички інших людей у режимі реального часу, незалежно від того, чи вони мають кількох послідовників, чи знаменитості з мільйонами підписників.

Бізнес також не залишається осторонь, і кмітливий менеджер у соціальних мережах вважається не менш важливим співробітником ніж завідувач виробництвом, особливо беручи до уваги його можливість створити або зламати репутацію бренду.

Курс соціальних медіа за останні кілька років був дещо нерівним, але суспільство ніколи раніше не мало змоги до глобальної комунікації у такому масштабі.

Звичайні користувачі ще можуть відмовитися від участі в соціальних мережах, але підприємства, які не брали участі в акції, незабаром відставатимуть від конкурентів. Цифровий маркетинг – надзвичайно важливий аспект будь-якої компанії, яка має присутність в Інтернеті.

## Розумне відстеження здоров'я

Ще однією тенденцією в технологіях стало зростання фітнес-пристроїв. Люди досить давно використовують техніку, що дає змогу оцінити стан організму. Але з новими технологіями це можна робити все більш точно, з миттєвим зворотним зв’язком та рекомендаціями від пристроїв, які можуть контролювати сон, роботу чи тренування. Можливість слідкувати за пульсом та кров’яним тиском, відстежувати та контролювати плани вправ, стежити за режимом сну – все це зараз сприймається як звична функція.

## 1.5 Наявні технології

Сфера розробки вебдодатків не надто поступається швидкістю розвитку іншим сферам вебтехнологій. Постійно з'являються нові інструменти та засоби, призначені спростити життя розробникам чи оптимізувати існуючі процеси. Основним стеком технологій для створення клієнтських вебдодатків є html, css та js. Декілька років назад їх було достатньо; зараз їх використання в чистому вигляді вважається не раціональним для додатків середнього та великого розміру. Сучасні інструменти значно пришвидшують швидкість розробки ціною швидкодії (якщо порівнювати ідеальний додаток на «чистому» стекі та фреймворках), але таке грубе прирівнювання подібне порівнянню написання програм на Ассемблері та C++.

### HTML

HTML (Hypertext Markup Language) – це код, який використовується для структурування і відображення вебсторінки і її контенту.

Прийнято вважати, що перша версія HTML була написана в 1993 році Тімом Бернерсом-Лі. Після того було створено багато різних версій HTML. Найбільш широко використовуваною версією протягом 2000-х років був HTML 4.01, який став офіційним стандартом у грудні 1999 року. Зараз неофіційним стандартом є HTML 5ої версії. Завдяки принципу зворотної сумісності веба можлива робота з більш старими версіями.

Інша версія гіпертекстової розмітки – XHTML – є результатом перепису HTML як мови XML.

В HTML можна розмістити контент всередині будь-якої кількості таблиць, параграфів чи сегментів. Підтримується використання маркованих списків, зображень, відео, таблиць даних тощо.

HTML не вважається мовою програмування в загальному розумінні. Це мова розмітки, яка використовується, щоб повідомити браузеру, як саме відобразити ту чи іншу вебсторінки. HTML складається з ряду визначених елементів. Вони використовуються, щоб вкладати або обертати різні частини

контенту. Це дає змогу змінювати порядок відображення контенту або змушує його діяти специфічним способом.

## CSS

CSS означає каскадні таблиці стилів. Це мова опису презентації вебсторінок, включаючи кольори, макет та шрифти, що робить вебсторінки презентабельними для користувачів.

CSS призначений для створення таблиць стилів. Він не залежить від HTML і може використовуватися з будь-якою мовою розмітки на основі XML.

Все більшої популярності набувають препроцесори. Препроцесори CSS - це мови сценаріїв, які розширюють можливості CSS за замовчуванням. Вони дозволяють використовувати логіку у CSS коді, таку як змінні, вкладення, успадкування, міксини, функції, математичні операції та інші. Найбільш популярні SASS та LESS. Препроцесори дають більший рівень абстракції та дозволяють писати більш лаконічні та зрозумілі конструкції. Але в результаті, під час збирання проєкту, код написаний в препроцесорі компілюється в CSS.

## Javascript

JavaScript – це сценарій або мова програмування, що дозволяє реалізовувати складні функції на вебсторінках (або в іншому середовищі компіляції). Своєчасне оновлення вмісту, інтерактивні карти, анімовані 2D/3D графіка, прокрутка музичних творів, тощо – все це результат роботи JavaScript.

JavaScript підтримує подієві, функціональні та імперативні стилі програмування. Він має інтерфейси прикладного програмування (API) для роботи з текстом, датами, регулярними виразами, стандартними структурами даних та об’єктною моделлю документа (DOM).

Основний стандарт JavaScript (ECMAScript) не передбачає засоби введення/виведення інформації (такі як графічні засоби чи мережеві сховища). В реальних за стосунках такі засоби зазвичай надає веббраузер (чи інша система).

Спочатку двигуни JavaScript використовувалися лише у веббраузерах, але тепер вони є основними компонентами інших програмних систем, часто серверів та різноманітних програм.

У сучасних реаліях JavaScript в чистому вигляді використовується все рідше. Йому на зміну прийшли фреймворки. Фреймворк часто є багатошаровою структурою, яка вказує на те, які програми можна або потрібно будувати і як вони будуть взаємопов'язані. Деякі фреймворки комп’ютерної системи також містять актуальні програми, визначають інтерфейси програмування або пропонують інструменти програмування для використання фреймворків. Може визначати структуру для набору функцій у системі та їх взаємозв’язку; шарів операційної системи; шарів підсистеми додатків; може визначати як стандартизувати комунікацію на певному рівні мережі; і так далі.

Найбільш популярні фреймворки для JavaScript: React, Vue та Angular.

## React

React інколи називають «неповноцінним» фреймворком або бібліотекою. Це пов’язано з тим, що в своєму чистому вигляді він пропонує лише контроль за відображенням (динамічний ирендер сторінок при зміні значень певних змінних). Всі інші процеси (роутинг, повідомлення, стан додатку, інжекції та інше) лягають на плечі розробників. Насправді ж під React’ом зазвичай мають на увазі повноцінну систему: більшість необхідних функцій для будь-якого додатку існують у вигляді бібліотек, сумісних (або ж взагалі написаних лише для) React. Він був створений та підтримується Facebook.

Загалом, фреймворк визначають як такий, що має велику швидкість розгортання та низький поріг входження (достатньо знати JavaScript та HTML, щоб зрозуміти його основні концепції). З іншого боку, це приводить до дещо більших проблем відносно підтримки проєкту та входження нових розробників до

існуючої команди (звісно, якщо такі проблеми не були передбачені та відносно них не прийняли відповідних заходів).

### Vue.js

Vue.js дуже схожий на React, але він пропонує трохи більше готових рішень «з коробки». Vue.js пропонує значну свободу в виборі бібліотек, технологій, принципів розробки та архітектурних підходів. Створений та підтримується спільнотою ентузіастів (на відміну від React та Angular не має за собою великої корпорації).

Характеризується гнучкістю та досить низьким розміром оптимізованого додатку. Має подібні до React'у переваги та недоліки.

### Angular

Angular – найбільший фреймворк з наявних. В базовій збірці він одразу пропонує більшість рішень для додатків. Загальна архітектура, принцип організації та взаємодії компонентів, робота з стилями та шаблонами – все це уже визначено до написання безпосередньо коду програми. Зазвичай він надає лише один підхід для вирішення певної задачі.

В результаті, переваги та недоліки протилежні до React’а та Vue.js’а. Він має набагато вищий поріг входу, вимагаючи від розробника знання API та принципів Angular. Розгортання додатку, створення базової реалізації, підготовка моделей – все це потребує більше часу, ніж в інших фреймворках. З протилежної сторони, єдиний підхід до побудови проєктів спрощує подальшу підтримку та входження нових розробників до існуючої команди.

## 1.6 JavaScript фреймворк з визначеною модульною архітектурою Angular

Angular – фреймворк JavaScript, який допомагає розробникам створювати додатки. Він надає безліч інструментів, які просто роблять реалізацію великих завдань сучасних додатків, таких як прив’язка даних, маршрутизація, анімація та інше.

Angular також представляє ряд конвенцій про підходи до розробки додатків. Це може бути дуже корисно для великих команд, які повинні працювати разом у одному проекті; або ж для поодиноких розробників, що поки не дуже розуміються на побудові архітектури для складних додатків. Це один з небагатьох фреймворків JavaScript, який забезпечує майже вичерпний посібник «стайлгайду» з великою кількістю оглядних прикладів того, як писати код, використовуючи цей фреймворк.

Головні особливості:

**Генерація коду**

Angular перетворює шаблони на код, оптимізований для сучасних віртуальних машин JavaScript, надаючи переваги рукописного коду та продуктивність фреймворку.

**Шаблони HTML**

Дозволяють створювати представлення інтерфейсу за допомогою простого та потужного синтаксису шаблонів. Включає в себе декілька спеціалізованих директив, що дозволяють значно пришвидшити та оптимізувати роботу з DOM.

**Angular CLI**

Надає інструменти командного рядка: дає змогу створити повноцінний проєкт за одну команду, додати компоненти, директиви, сервіси та інше. Дозволяє одразу включати тести до створених модулів, а також дає змогу миттєво розгорнути додаток.

**Інтеграція з IDE**

Типовий підхід до вирішення більшості задач дає високу інтегрованість з існуючими редакторами коду, що включає: розумне завершення коду, миттєві помилки, відгуки та інше.

**Розбиття коду**

Сторінки швидко завантажуються за допомогою компонентного маршрутизатора, який забезпечує автоматичне розбиття коду; користувачі

завантажують лише код, необхідний для відображення запиту, який вони бажають отримати.

### Тестування

Новий проєкт одразу включає в себе Jasmine та Protractor, що дає змогу легко створювати юніт, інтеграційні та e2e тести.

### Анімації

Angular дозволяє створювати високопродуктивні, складні, послідовні та часові шкали анімації з дуже лаконічним кодом. Він надає власне API для таких анімацій, в той же час маючи повну підтримку нативних css анімацій.

### Універсальність

Зібраний додаток може бути запущений на будь-якому з популярних серверів (включаючи, але не обмежуючись технологіями Node.js, .NET, PHP). Дозволяє майже миттєве відображення; надає інструменти для SEO оптимізації та серверного рендеру.

## Висновки

Була розглянута коротка історія виникнення інтернету, охарактеризовані чинники, що сприяли створенню системи передачі інформації та її подальшої глобалізації. Було проведене ознайомлення з останніми публікаціями, що стосуються розвитку інтернету, його впливу на соціум; та комп’ютерні ігри, як частковий випадок розвитку соціальних інститутів.

Проведено дослідження сучасних інструментів розробки, підходів та методологій. Виокремлено та розглянуто найбільш поширені засоби створення вебдодатків. Проведено коротке порівняння найпопулярніших JavaScript фреймворків. Обґрунтовано вибір платформи Angular та проаналізовано її основні особливості.

# Розділ 2. Алгоритмічне забезпечення. Підходи та методи розробки використані в ході створення додатку

## 2.1 Цикл ігрового процесу та інтерфейс

Майже всі ігри мають схожий алгоритм запуску, роботи головного меню та входу в ігровий процес. Все починається за запуску гри, після чого користувач потрапляє у головне меню гри. В цьому розділі інтерфейсу може бути декілька кнопок які ведуть до наступних розділів:

- Створення нової гри;
- Загрузка збереженої гри;
- Налаштування;
- Графічні;
- Керування;
- Локалізація;
- Основні;
- Інформація про розробника;
- Вихід з програмного додатку.

Всі ці розділи, в залежності від необхідності, зустрічаються у більшості ігор. В моєму випадку реалізований зручний інтерфейс, наявна вкладка гри, налаштувань та деталей. Представлена інформація про розробника та надані посилання на основне шахове ком'юніті.

Після створення нової гри користувач потрапляє у середовище де буде

відбуватися ігровий процес. Ігровий процес, на відміну від інтерфейсу, в усіх іграх працює за різними алгоритмами і правилами.

## 2.2 Загальна структура програми

Додаток включає в себе декілька модулів, що відповідають за певну частину логіки програми; та ядро, що їх об'єднує:

- Ігровий двигун – містить правила переміщення кожної з фігур, можливі ігрові нюанси та правила їх обробки;
- Ігрова дошка – містить частину логіки, що відповідає за побудову дошки, розміщення фігур та опис методів взаємодії з ними;
- Сервіс налаштувань – зберігає інформацію щодо обраних користувачем опцій та надає цю інформацію іншим модулям. Також містить API для зміни збережених опцій;
- Користувацький інтерфейс – відповідає за графічне представлення наявної інформації та надає користувачеві можливість впливати на стан програми;
- Ядро – відповідає за взаємодію існуючих модулів. Ним виступає безпосередньо Angular; надає інструменти для передачі інформації між різними компонентами та сервісами. Надає необхідні засоби для побудови та оптимізації безпосередньо додатку.

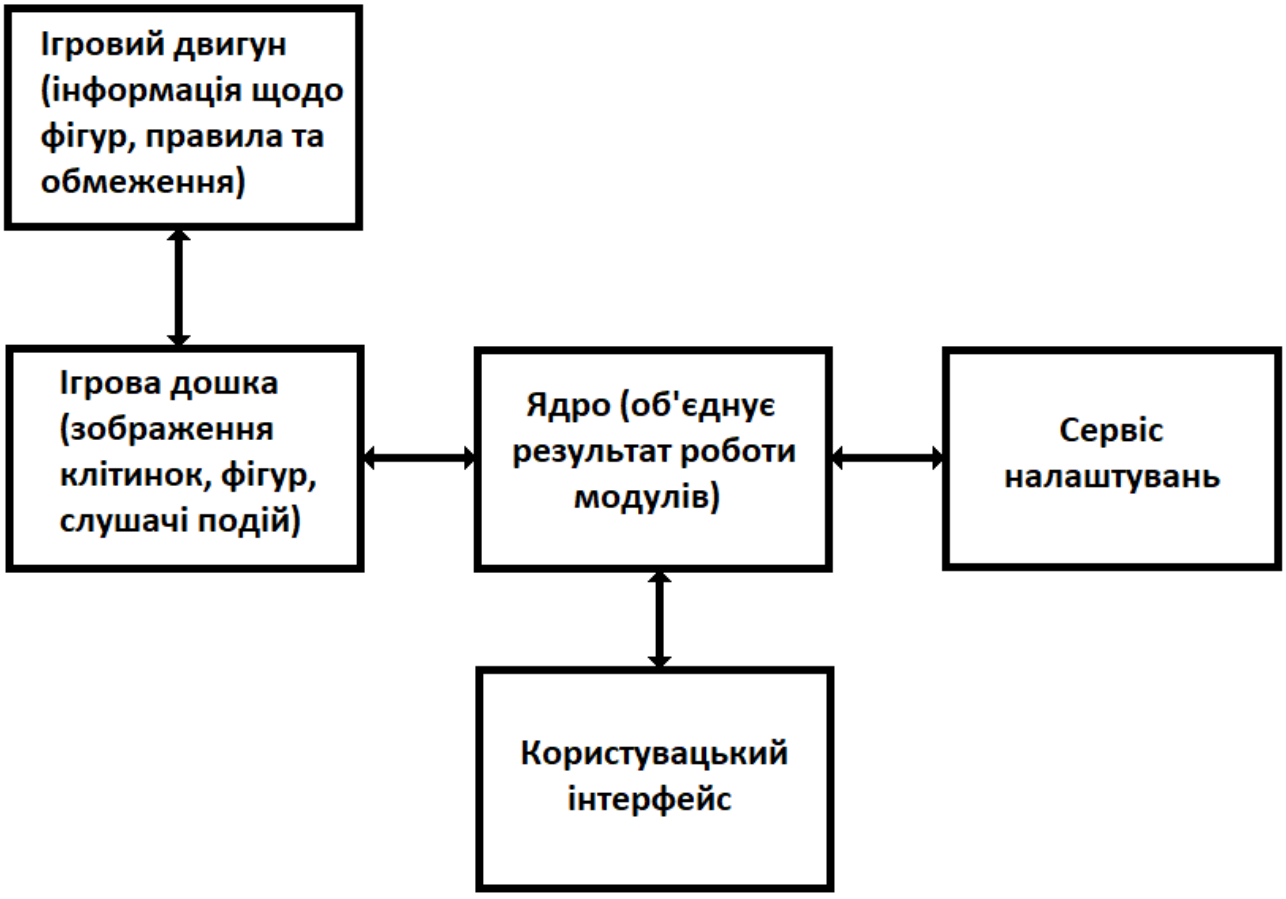



Рис. 1. Загальна структура взаємодії модулів додатку

Розглянемо їх більш детально в наступних розділах.

## 2.3 Ігровий двигун

Двигун зберігає стан гри, і на кожну спробу її оновлення спочатку перевіряє, чи таке оновлення задовольняє умовам гри. Двигун валідує існуючий стан дошки, порівнює надані дані з об'єктом фігури. Якщо такий хід може бути зроблений – перевіряє, чи не є цей хід особливим випадком (на кшталт шаху чи рокіровки) і потім перераховує новий стан відповідно до переданих та обрахованих даних. Має можливість ініціювати гру з певної позиції, або ж пропустити певну перевірку (що може бути корисним в деяких умовах, наприклад при навчанні або відтворенні партії).

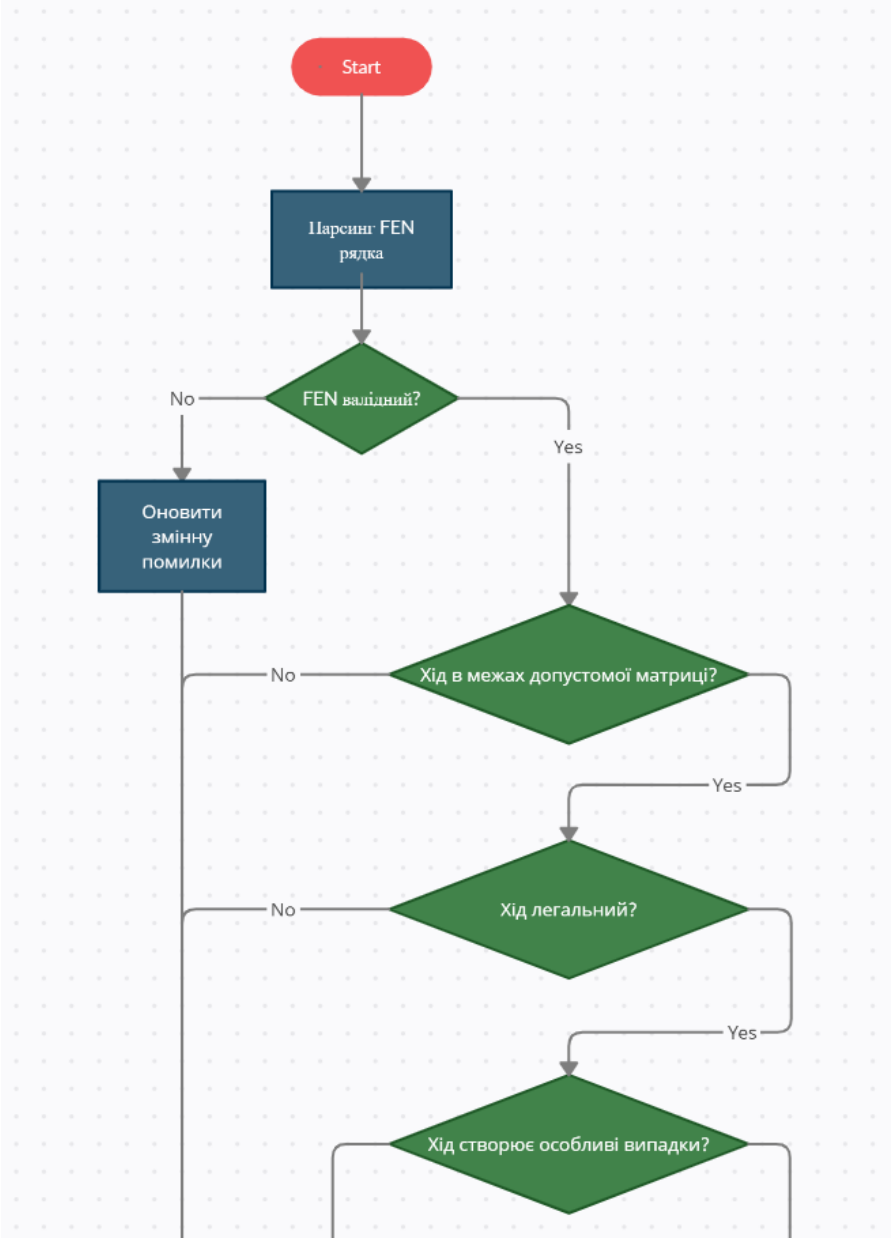



Рис. 2 Алгоритм роботи ігрового двигуна, частина 1

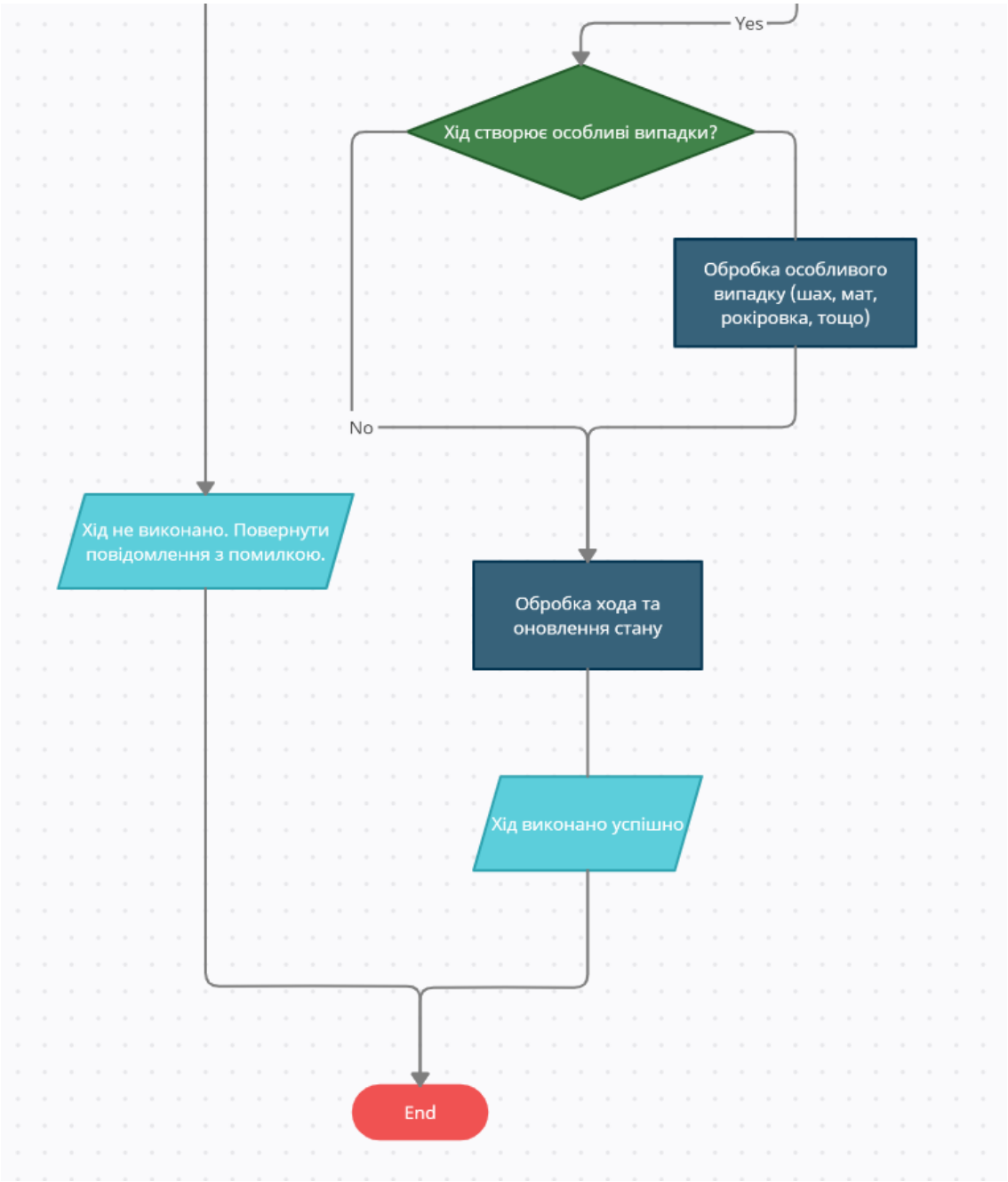



Рис. 3. Алгоритм роботи ігрового двигуна, частина 2

## 2.4 Ігрова дошка

Дошка відповідає за відображення безпосередньо дошки та фігур на ній. Вона прослуховує дії користувача та намагається змінити стан гри відповідно до них. Тобто дошка не здатна виявити, був такий хід правильним чи ні, лише передає інформацію щодо початкової клітинки ходу та кінцевої, а ігровий двигун вирішує, чи може такий хід бути зробленим.

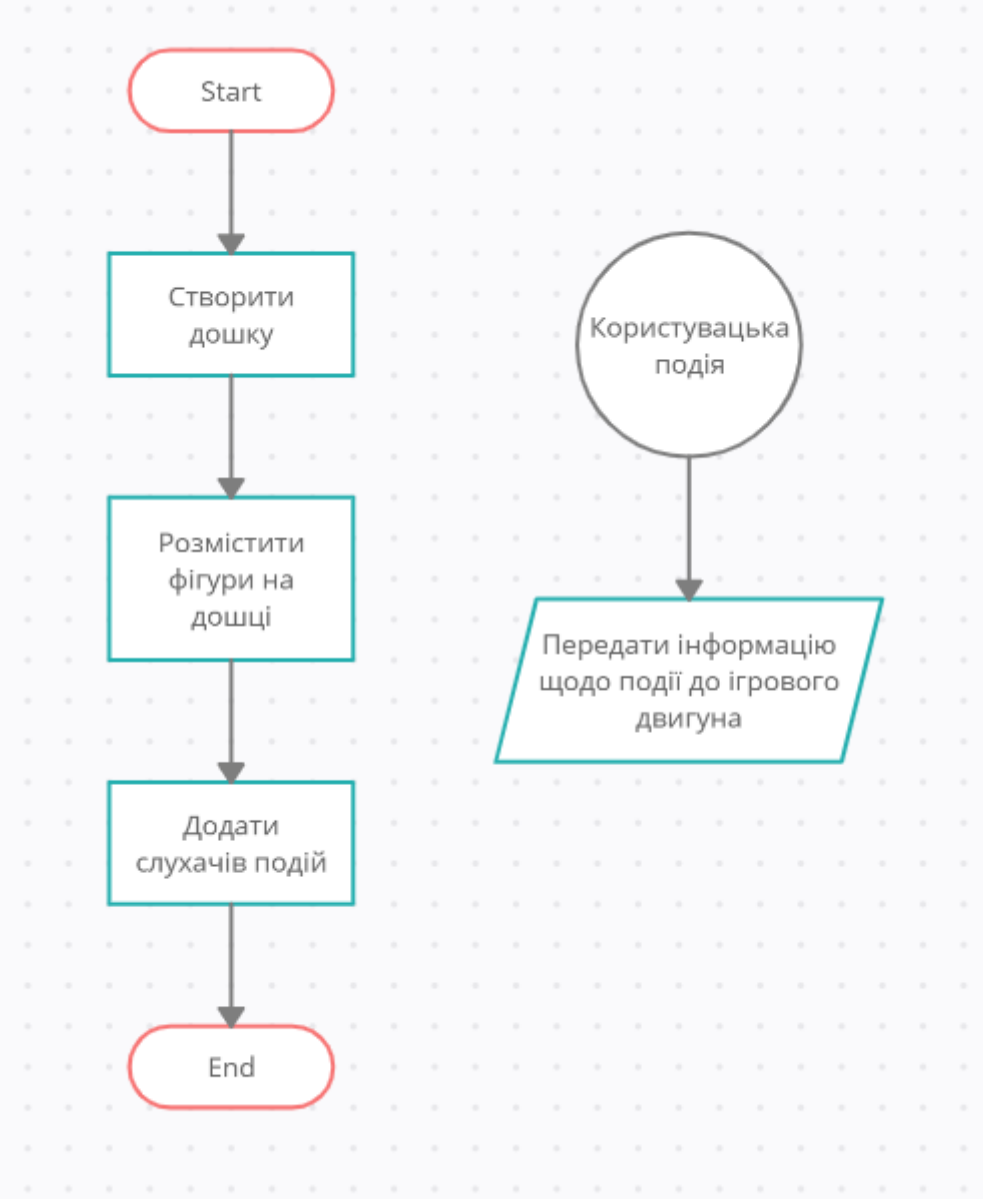



Рис. 4 Алгоритм роботи ігрової дошки

## 2.5 Сервіс налаштувань

Сервіс налаштувань відповідає за збереження та зміну користувацькх налаштувань. Коли будь-яка з опцій буде змінена, новий об’єкт опцій зберігається в браузерному сховищі (LocalStorage), що дає змогу відновити вибір користувача навіть після закінчення сесії. Під час запису нової опції вона спочатку валідується, потім об’єднується зі значеннями за замовченням та збереженими опціями. Це дозволяє завжди оновлювати налаштування з наступними пріоритетами (в порядку зростання): за замовчуванням, збережені опції, оновлені опції. Приблизно такий самий алгоритм використовується при запиті іншими компонентами налаштувань користувача. Збережений об’єкт спочатку

об’єднується зі значеннями за замовчуванням. Це дозволяє уникнути проблем відсутності або не валідності певних значень.

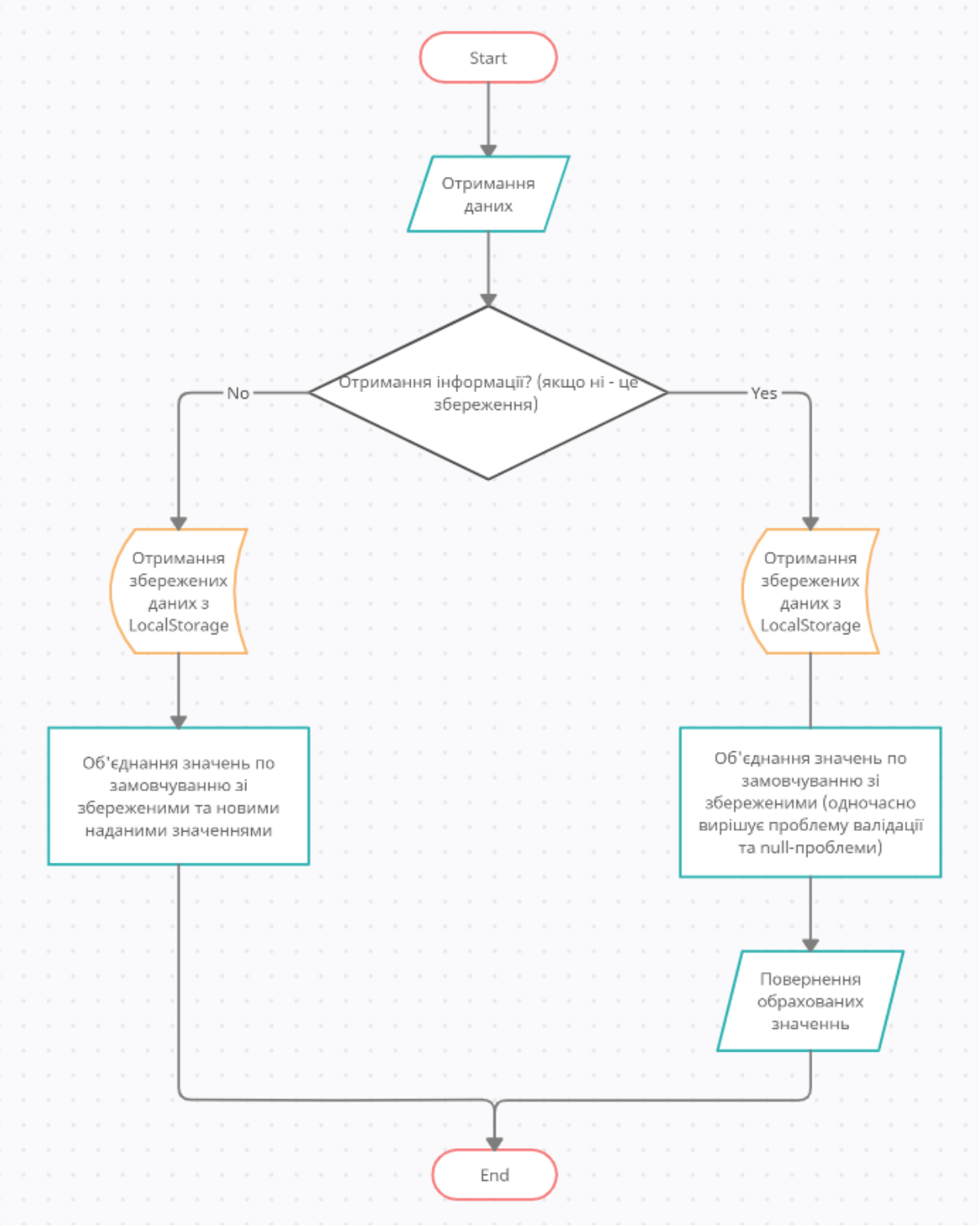



Рис. 5. Алгоритм роботи сервісу налаштувань

## 2.6 Користувацький інтерфейс

Користувацький інтерфейс відповідає за відображення стану програми на екрані користувача. Він дає змогу взаємодіяти з іншими модулями програми.

Особливість роботи Angular дозволяє автоматично змінювати зміст вебсторінки при динамічній зміні значень стану додатку. Це стосується лише простих змінних; складні структури, такі як об’єкти, не змінюють посилань на адреса пам’яті при своїх мутаціях, тому фреймворк не здатний відслідкувати та динамічно оновити контент у випадку таких змін. В такому разі зміст сторінки варто оновити програмно (або ж не використовувати складні структури для збереженню станів, а розбивати їх на більш прості).

В даному додатку, інтерфейс містить в собі панель навігації з посиланнями на основні сторінки веб-сайту; футер з інформацією про розробника та дату створення та основну частину – контент. В залежності від url контент динамічно перемальовується Angular’ом; це створює враження єдності веб-сайту, без необхідності перевантаження сторінки при кожній зміні наявного роуту. Додатки з таким підходом до обробки користувацьких івентів називають SPA (Single Page Application), тобто одно сторінковий додаток. Оскільки він справляє враження єдиного, цілісного програмного продукту це позитивно впливає на користувацький досвід.

## 2.7 Центральний сервіс взаємодії модулів

Центральний сервіс взаємодії модулів (далі – ядро) поєднує в собі перераховані вище модулі, налаштовує зв’язки між ними та відповідає за передачу інформації. Angular підтримує багато способів становлення зв’язків між компонентами:

- Прив'язка елемента DOM до значення компонента [одностороння]. У подвійних фігурних дужках вказується зміна яка ставиться у відповідник в шаблоні. Під час запуску програми запис на кшталт {{name}} буде автоматично замінятися відповідними значеннями, визначеними в компоненті. Якщо в процесі роботи додатка властивість name в компоненті змінить своє значення, то також зміниться значення в розмітці html, яка прив'язана до цих властивостей;

- Прив'язка властивості елемента DOM до значення компонента [одностороння]. Головна особливість – прив'язка відноситься не до атрибуту, а саме до властивості елемента в javascript, який представляє даний елемент html.

Тобто html-елемент <input> в javascript представлений інтерфейсом HTMLInputElement, у якого є властивість value;

- Прив'язка атрибуту елемента до значення компонента [одностороння]. Іноді виникає необхідність виконати прив'язку не до властивості, а саме до атрибуту html елемента. Хоча властивості і атрибути html елементів можуть перетинатися, але таке буває не завжди. У цьому випадку ми можемо використати вираз [attr.назва_атрибуту] = "значення". Після префікса attr через крапку йде назва атрибута. Зазвичай подібна прив'язка застосовується до атрибутів елементів aria, svg і table. Наприклад, атрибут colspan, який об'єднує стовпці таблиці, не має відповідної властивості. І в цьому випадку ми можемо застосовувати прив'язку до атрибутів;
- Прив'язка методу компонента до події в DOM (генерація події в DOM викликає метод на компоненті) [одностороння]. Дозволяє створювати нові події з середини компонента за допомогою спеціальних змінних. Виклик функції emit на такій змінній викличе відповідну подію, якщо вона була прив’язана за допомогою круглих дужок в шаблоні;
- Двостороння прив'язка, коли елемент DOM прив'язаний до значення на компоненті, при цьому зміни на одному кінці прив'язки відразу приводять до змін на іншому кінці. Як правило, двостороння прив'язка застосовується при роботі з елементами введення, наприклад, елементами типу input. Тобто, зміна в шаблоні оновить відповідну змінну в компоненті, і навпаки.

Також Angular має гарно розвинену API інжекцій, що дозволяє підключати сервіси у необхідні компоненти. Інстанс сервісу, переданий у конструктор компонента, буде створений під час генерації компонента, та знищений разом з ним. Це дуже потужний інструмент, що дозволяє взаємодіяти з даними на різних рівнях додатку, при цьому використовуючи оптимальну кількість пам’яті. Можлива інжекція абстрактних класів сервісів, коли компонент знає лише інтерфейс сервісу, але не його тип. Такий підхід дозволяє будувати дуже гнучкі додатки, з високим рівнем повторно використаного коду.

Всі перераховані вище інструменти були використані під час побудови додатку, окрім двосторонньої прив’язки – це досить специфічний тип об’єднання даних, що прийшов з AngularJS. В даному випадку в ньому не було необхідності.

## Висновки

В результаті проведеного дослідження було зібрано та структуровано інформацію щодо сучасних практик та методологій розробки вебдодатків. Було розглянуто основні алгоритми та принципи функціонування використаних технологій та сервісів. Наведено блок-схеми алгоритмів головних модулів та класів.

Досліджено основні способи відображення інформації та прив’язки візуальних даних до стану змінних всередині Angular компонентів.

# Розділ 3. Програмне забезпечення. Створення гри «Шахи» з можливістю гри для одного або двох гравців

## 3.1 Опис програми та основні функції

Для розробки вебдодатку була обрана платформа Angular, тому що вона має визначену сталу структуру, велику кількість готових прототипів вирішень типових завдань розробки та широкий вибір бібліотек з вільною ліцензією. В якості бібліотеки компонентів була обрана Angular Material – вона містить більшість необхідних елементів, стилізованих за стайлгайдом Google (ця ж бібліотека використовується майже у всіх вебдодатках Google). Для пришвидшення створення користувацького інтерфейсу була додана бібліотека Bootstrap. Вона містить власний набір стилізованих компонентів та сітку для швидкого розміщення блоків програми. В даному вебдодаткі була використана лише сітка; можливий подальший розвиток програми, завдяки вибору бібліотек та БЕМ підходу до опису компонентів, приведення сторінок до адаптивного вигляду має бути досить простим завданням.

Angular дозволяє без великих затрат часу будувати оптимізовані вебдодатки, з гарною підтримкою в більшості браузерів. Гарна оптимізація досить важлива для програмних продуктів, адже основний підхід веброзробки – клієнт-сервер, має на увазі створення так званих тонких «клієнтів», тобто водночас низьке навантаження на машину клієнта та висока швидкість відгуку. Використані підходи та інструменти дозволяють створити такий додаток. Можливий подальший розвиток – підключення додатку до серверного API, що дало б змогу зберігати прогрес на віддаленій машині, а також грати з іншими гравцями онлайн в режимі реального часу.

Основна мета шахової партії – поставити «мат» супротивнику, тобто зробити такий хід, щоб водночас створити загрозу ворожому королю та відрізати йому шлях до відступу.

Гра буде закінчена коли користувач зможе зробити такий хід. Також користувач має змогу налаштувати, за фігури якого кольору він бажає грати; складність шахового бота в грі на одного гравця; зовнішній вигляд фігур та дошки.

## 3.2 Інтерфейс додатку

### Головна сторінка

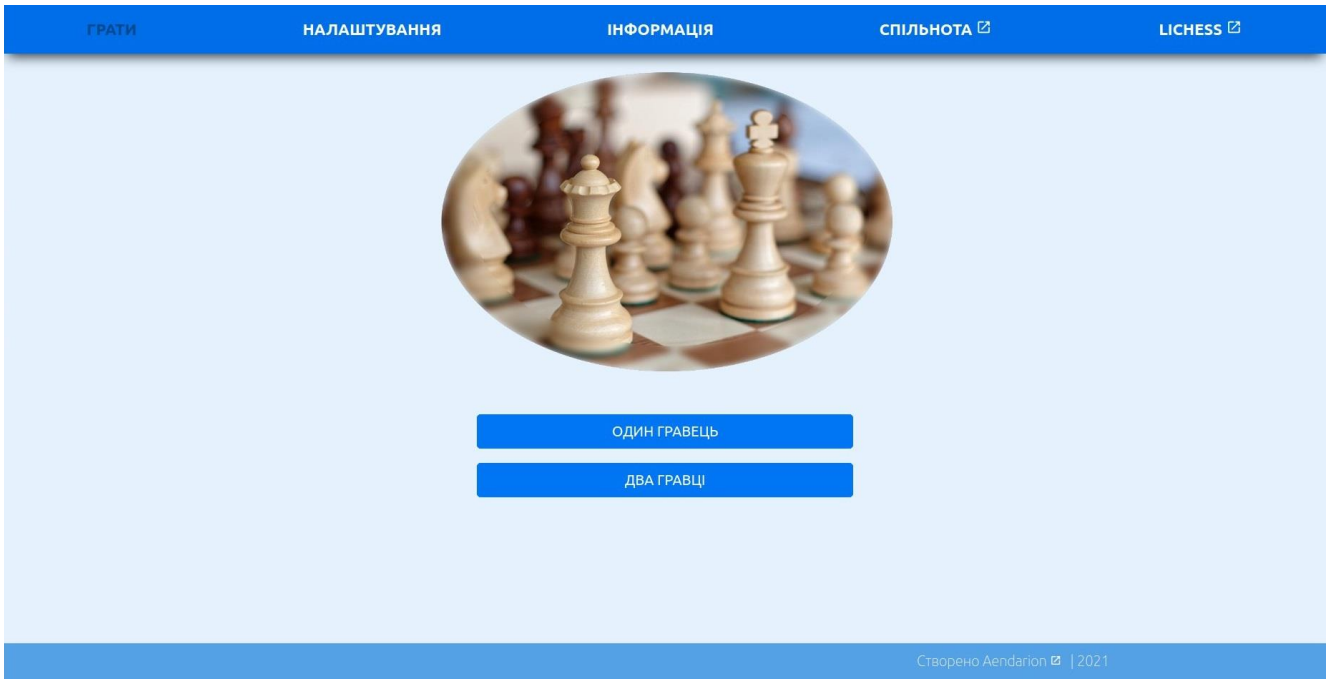



Рис. 6. Зображення головної сторінки сайту

При переході на сторінку вебсайту користувач одразу потрапляє на сторінку вибору режиму гри. Вона представлена хедером та футером, що характерний для всього сайту; а також, двома кнопками «один гравець» та «два гравці». При наведенні на одну з кнопок змінюється зображення відповідно до режиму гри, на який було наведено. Обидві кнопки ведуть на сторінку гри; «один гравець» - користувач буде грати проти шахового боту, «два гравці» - проти іншого гравця, тобто ходи будуть робитись на одному екрані по черзі.

## Сторінка гри

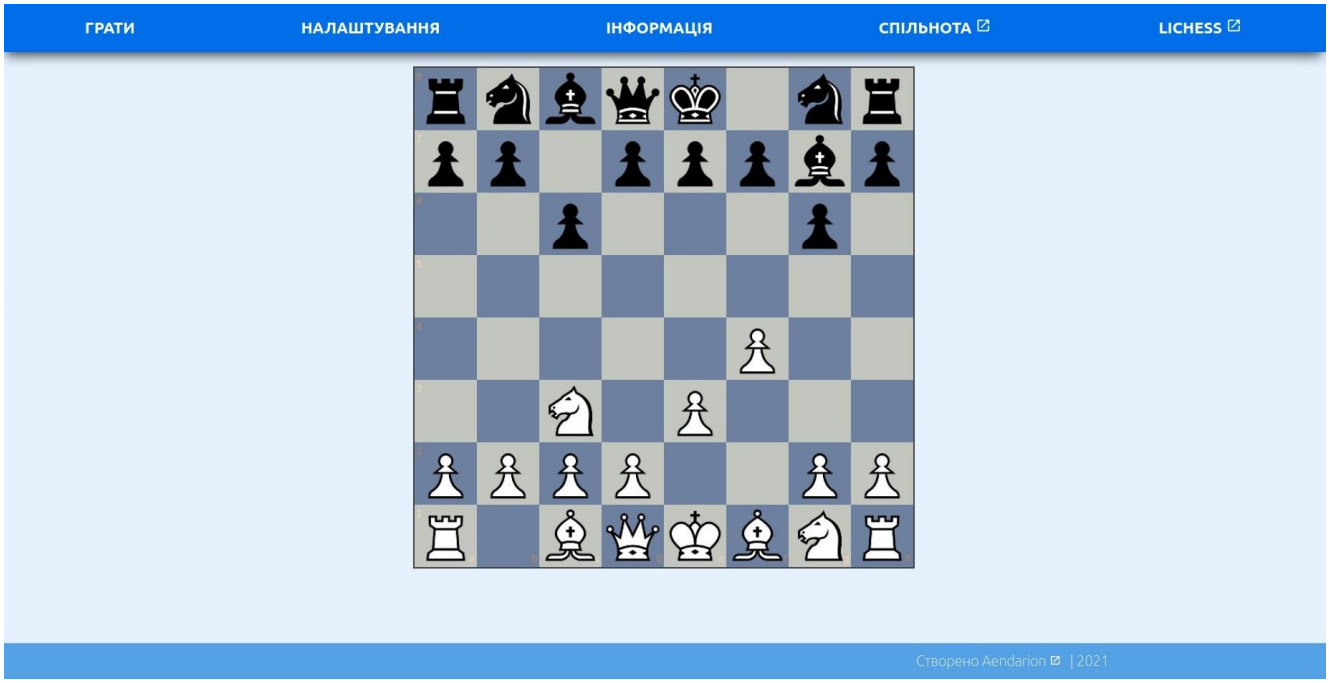



Рис. 7. Зображення сторінки гри

Сторінка гри представлена шаховою дошкою та фігурами на них. Управління фігурами відбувається за допомогою інтуїтивно зрозумілого способу перетягування (drag&drop). Якщо дія користувача не є допустимою – фігура просто повернеться на попереднє місце.

## Сторінка налаштувань

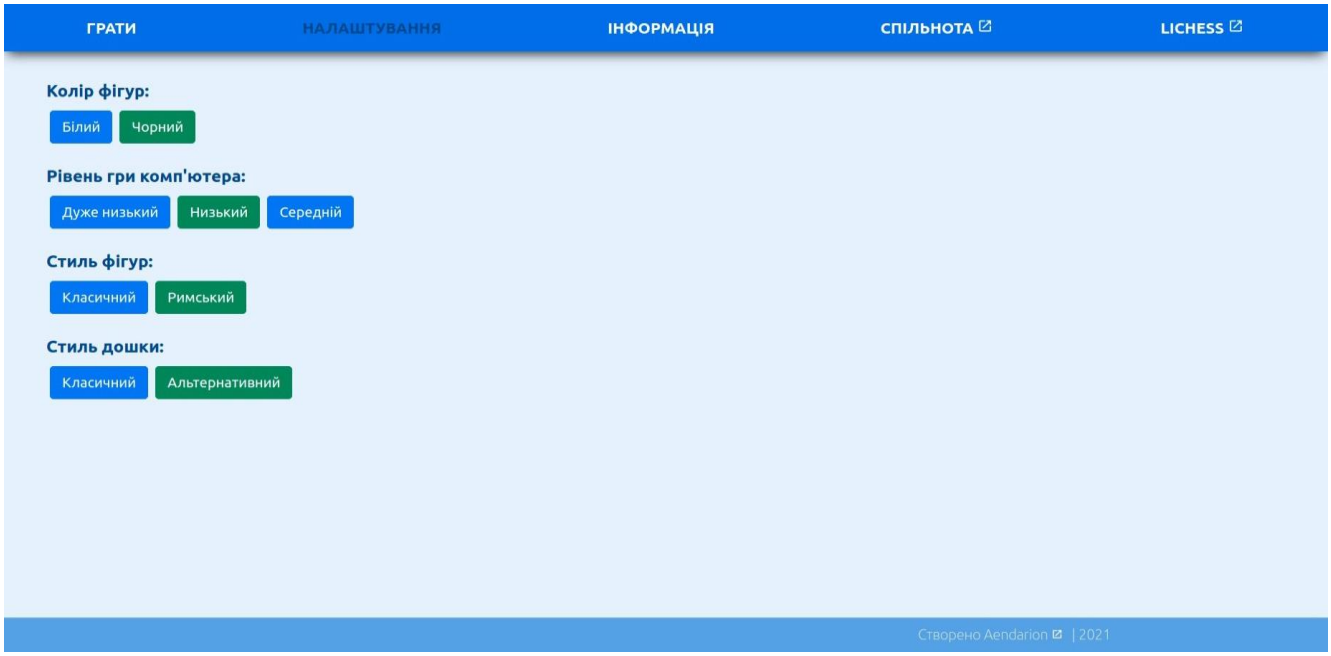



Рис. 8. Зображення сторінки налаштувань

Сторінка налаштувань включає в себе декілька перемикачів, що відображають наявні опції та обрані серед них. Клік по опції миттєву змінює наявні налаштування та оновлює їх стан в локальному сховищі. У випадку необхідності, дуже просто додати нові опції.

**Сторінка інформації**


Рис. 9. Зображення інформаційної сторінки

Інформаційна сторінка містить короткі деталі щодо реалізованого вебсайту, використаних підходів та технологій. Для лаконічної подачі інформації був використаний компонент з бібліотеки Angular Material – «акордеон». Він дозволяє розгортати деталі заголовку по кліку. Завдяки цьому сторінка, що містить багато текстового контенту, візуально займає не багато місця.

## 3.3 Використані програмні засоби

У проекті були використані наступні програмні засоби:

- Webstorm — Integrated Drive Electronics (Інтегроване середовище розробки);
- Figma — редактор для макетів (створення та редагування онлайн);
- Git — розподілена система контролю версій;
- Photoshop — мультифункціональний графічний редактор;

- Веббраузери (для тестування та запуску): Chromium, Google Chrome, Opera, Firefox.

## 3.4 Використанні компоненти та класи

У проекті були використані наступні класи:

OnePlayerComponent. Названий клас являє собою об'єкт, що контролює основну частину процесу гри для одного гравця. Вперше, примірник (інстанс) цього класу створюється ядром, при переході на відповідний url; що зумовлено вибором відповідного режиму. Клас контролює більшість динамічних елементів розташованих в зоні активності. Він відповідає за змінні стану гри, валідує переміщення фігур та контролює реакції на користувацькі події шахового бота.

Клас включає в себе реалізацію алгоритму шахового бота. Спочатку визначається глибина прорахування в матриці ходів (визначається в налаштуваннях), будується дерево рішень та відбувається певна кількість ітерацій для пошуку оптимального рішення (відповідно визначених раніше).

TwoPlayersComponent. Названий клас являє собою об'єкт, що дублює логіку змінних та валідації попереднього класу, проте не містить алгоритму роботи шахового бота (вона не потрібна для гри за двох учасників). Також, містить дещо модифікований алгоритм роботи з активною стороною — оскільки колір фігур змінюється покроково, в класі покращені умови переміщення фігур певного кольору.

OptionsService. Названий клас являє собою об'єкт, що зберігає користувацькі налаштування та синхронізує їх з файловою системою (не буквально, з використанням API LocalStorage). Вперше, примірник (інстанс) цього класу створюється ядром, при побудові одного з керуючих класів, якщо він інжектований в його конструктор. В класі реалізовані сетери та гетери, що містять логіку збереження та редагування з сайд ефектами та об’єкт options; це дає змогу перевизначати та зчитувати значення в одному й тому ж методі. Такий підхід значно покращує модель взаємодії з налаштуваннями. Для роботи з LocalStorage використовується відповідний клас, за замовчуванням визначений в Angular

(такий підхід дає змогу зберігати файли локально без додаткових бібліотек та інжекцій).

Також в програмі наявні класи: NotFoundPageComponent, GameModeComponent, AboutComponent та інші. Вони відповідають за представлення графічної інформації, зазвичай зберігають в собі декілька статусних змінних та не реалізують специфічної логіки.

Компоненти з бібліотеки Angular Material, використані у додатку:

- Buttons – реалізує стилі кнопки, містить прості попередньо визначені правила відображення, рендеренгу та реакції на користувацькі події.
- Accordion – реалізує логіку згортання та розгортання переданої інформації. Використовується для лаконічної подачі текстової інформації. Окрім типових функцій містить логіку створення та видалення власних слухачів користувацьких подій.

Компоненти інтерфейсу:

- Button – реалізовує широкі можливості створення кнопок на веб-сторінці.
- Image – призначений для відображення графічних зображень в об'єкті інтерфейсу.
- P – використовується для відображення тексту; пропоноє можливість значного редагування стилів контенту.
- Div – універсальний тег для групування та позиціонування будь-яких елементів.

## Висновки

В результаті проведеного дослідження був розроблений вебдодаток (гра) на базі JavaScript фреймворку Angular з використанням бібліотек Angular Material та Bootstrap і методології БЕМ. Основний функціонал додатку: проведення шахової партії для одного та двох гравців; відображення інформації щодо використаних підходів та компонентів; надання посилань до сайтів шахової спільноти.

Для розробки додатку була використана мова програмування TypeScript (JavaScript); в якості допоміжних технологій представлення візуальної інформації

були обрані html та scss. Розглянуто основні принципи створення циклу ігрового процесу та інтерфейсу.

# Висновки

Було проведено дослідження в напрямку історії розвитку вебтехнологій, розглянуті чинники, що впливали на прогресс створення систем передачі та обміну інформацією. Були розглянуті сучасні наукові публікації, що стосуються вищезгаданих тем.

Досліджено інструментів розробки програмного забезпечення, розглянуті найбільш поширені підходи та методології. Проведено огляд загальноприйнятих технологій та фреймворків для створення вебсайтів. Більш детально розглянута платформа для створення вебдодатків Angular.

Були перечислені Розглянуті основні алгоритми програми, принципи побудови інтерфейсу користувача та розглянуті основні програмні компоненти. Надано опис головним классам компонентів та розглянуто принципи їх функціонування. Виокремлено основні програмні компоненти та надано опис функціоналу та алгоритмів функціонування специфічних (таких, що не встановлені за замовчуванням) сторінок додатку.

Був розроблений вебдодаток (гра) використанням платформи Angular. Надано скріншоти роботи програми та короткий опис функціоналу наданих зображень сторінок. Розглянуто функціонал сторінок створеного додатку: режим гри для одного гравця, проти шахового бота; режим гри для двох гравців; сторінка налаштування режиму гри, ігрового поля та фігур; головну сторінку та сторінку з інформацією щодо використаних технологій.

Для розробки додатку була використана мова програмування TypeScript (надбудова над JavaScript); в якості технологій для відображення стану гри були обрані html та scss (препроцесор для css). В процесі створення додатку були використані загальноприйняті підходи до реалізації модульної архітектури та впроваджені принципи побудови об'єктно-орієнтованих програм: абстракція, поліморфізм, наслідування та інкапсуляція.

# Літературні джерела


1. Вплив сучасних інтернет-технологій [Електронний ресурс] // Центр досліджень соціальних комунікацій – 2012. – Режим доступу до ресурсу: http://nbuviap.gov.ua/index.php?option=com_content&view=article&id=1059:metodika-vidboru-internet-informatsiji-vpliv-suchasnikh-internet-tekhnologij&catid=127&Itemid=460.
2. Вплив можливостей інтернет-технологій на розвиток підприємств [Електронний ресурс] // Ефективна економіка – 2017. – Режим доступу до ресурсу: http://www.economy.nayka.com.ua/?op=1&z=2264.
3. Вплив Інтернет-технологій на формування лідерських якостей молоді [Електронний ресурс] // ARMG Publishing – 2008. – Режим доступу до ресурсу: https://armgpublishing.sumdu.edu.ua/ua/uajournals/uabel/volume-3-issue-3/article-9/.
4. Фреймворки в веб-разработке [Електронний ресурс] // Автоматизация бизнеса – 2011. – Режим доступу до ресурсу: https://web-creator.ru/articles/about_frameworks.
5. ТОП-10 фреймворков для веб-разработки в 2020 [Електронний ресурс] // Proglib – 2020. – Режим доступу до ресурсу: https://proglib.io/p/web-frameworks-2020.
6. Анализ шести веб-фреймворков: плюсы, минусы и особенности выбора [Електронний ресурс] // Habr – 2013. – Режим доступу до ресурсу: https://habr.com/ru/company/ruvds/blog/343894/.
7. Angular (фреймворк) [Електронний ресурс] // Wikipedia – 2014. – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Angular_(%D1%84%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA).
8. Введение в Angular [Електронний ресурс] // Webformyself – 2017. – Режим доступу до ресурсу: https://webformyself.com/vvedenie-v-angular-chto-eto-za-frejmvork-i-zachem-ego-ispolzovat/.

9. Что такое Angular [Електронний ресурс] // Metanit – 2015. – Режим доступу до ресурсу: https://metanit.com/web/angular2/1.1.php.
10. Ігровий цикл [Електронний ресурс] // Житомирська політехніка – 2017. – Режим доступу до ресурсу: https://learn.ztu.edu.ua/pluginfile.php/156654/mod_resource/content/1/%D0%9B%D0%B5%D0%BA%D1%86%D1%96%D1%8F_4.pdf.
11. Структура гри [Електронний ресурс] // Студопедія – 2019. – Режим доступу до ресурсу: https://studopedia.com.ua/1_427578_struktura-gri.html.
12. Структура програми [Електронний ресурс] // Step by step – 2014. – Режим доступу до ресурсу: https://step.org.ua/konspekt/pascal/tema5.
13. Основні принципи будови хорошого сайту [Електронний ресурс] // SEO Guide – 2013. – Режим доступу до ресурсу: https://www.taina.com.ua/osnovni-pryncypy-budovy-horoshoho-sajtu/.
14. Структура веб-сайтів, різновиди веб-сторінок [Електронний ресурс] // Stud MSK – 2015. – Режим доступу до ресурсу: http://stud-msk.ho.ua/inf/91.htm.
15. Адаптивний вебдизайн [Електронний ресурс] // Wikipedia – 2015. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%90%D0%B4%D0%B0%D0%BF%D1%82%D0%B8%D0%B2%D0%BD%D0%B8%D0%B9_%D0%B2%D0%B5%D0%B1%D0%B4%D0%B8%D0%B7%D0%B0%D0%B9%D0%BD.
16. Принципи проектування ефективних веб-сайтів [Електронний ресурс] // Молодий вчений – 2015. – Режим доступу до ресурсу: http://molodyvcheny.in.ua/files/journal/2015/9/66.pdf.
17. Історія розвитку Web-сервісів [Електронний ресурс] // Використання сервісів Web 2.0 в науковій діяльності – 2017. – Режим доступу до ресурсу: http://web20mikitchak.blogspot.com/p/blog-page_26.html.
18. Тенденції та перспективи розвитку сайтобудування, їх вплив на архівне копіювання веб-сайтів [Електронний ресурс] // Статті та повідомлення – 2018. – Режим доступу до ресурсу: https://archives.gov.ua/wp-content/uploads/09-18.pdf.

19. Актуальні тенденції розвитку технологій в галузі веб-програмування: аналітичний огляд [Електронний ресурс] // MDU – 2014. – Режим доступу до ресурсу: http://mdu.edu.ua/wp-content/uploads/gmit114.pdf.
20. Розвиток інтернет та веб технологій [Електронний ресурс] // Українська бібліотечна асоціація – 2014. – Режим доступу до ресурсу: https://www.slideshare.net/osadchasv/20-30842608.
21. 15 провідних тенденцій веб розробки у 2020 році [Електронний ресурс] // WEB4U – 2020. – Режим доступу до ресурсу: https://web4u.in.ua/blog/15-prov-dnih-tendenc-y-veb-rozrobki-u-2020-roc-31.
22. HTML [Електронний ресурс] // Wikipedia – 2013. – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/HTML.
23. Особенности архитектуры веб-сайта [Електронний ресурс] // BulgarPromo – 2018. – Режим доступу до ресурсу: https://bulgar-promo.ru/Arkhitektura_web-saita.
24. Модели жизненного цикла, принципы и методологии разработки программного обеспечения [Електронний ресурс] // Evergreen – 2019. – Режим доступу до ресурсу: https://evergreens.com.ua/ru/articles/software-development-metodologies.html.
25. Библиотека (программирование) [Електронний ресурс] // Wikipedia – 2014. – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5).
26. Етапи, методології, процеси та моделі SDLC [Електронний ресурс] // Sdlc Phases – 2020. – Режим доступу до ресурсу: https://uk.myservername.com/sdlc-phases.
27. Життєвий цикл програмного забезпечення [Електронний ресурс] // Wikipedia – 2017. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%96%D0%B8%D1%82%D1%82%D1%94%D

0%B2%D0%B8%D0%B9_%D1%86%D0%B8%D0%BA%D0%BB_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F.

28. Язык JavaScript [Електронний ресурс] // Современный учебник JavaScript – 2016. – Режим доступу до ресурсу: https://learn.javascript.ru/.
29. CSS [Електронний ресурс] // Wikipedia – 2015. – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/CSS.
30. Стадії циклу розробки ПЗ [Електронний ресурс] // qalight – 2018. – Режим доступу до ресурсу: https://qalight.ua/baza-znaniy/stadiyi-tsiklu-rozrobki-pz/.
31. CRUD [Електронний ресурс] // Wikipedia – 2014. – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/CRUD.
32. Основные понятия и особенности клиент-серверной архитектуры [Електронний ресурс] // TestMatick – 2018. – Режим доступу до ресурсу: https://testmatick.com/ru/osnovnye-ponyatiya-i-osobennosti-klient-servernoj-arhitektury/.

# ДОДАТОК 1

Код програми

## app.routing-module.ts:

```
import {RouterModule, Routes} from '@angular/router';
import {GameModeComponent} from './game-mode/game-mode.component';
import {NgModule} from '@angular/core';
import {BrowserAnimationsModule} from '@angular/platform-browser/animations';
import {NotFoundPageComponent} from './not-found-page/not-found-
page.component';
import {TwoPlayersComponent} from './two-players/two-players.component';
import {OptionsComponent} from './options/options.component';
import {AboutComponent} from './about/about.component';
import {OnePlayerComponent} from './one-player/one-player.component';

const routes: Routes = [
  {
    path: '',
    redirectTo: 'game-mode',
    pathMatch: 'full',
  },
  {
    path: 'game-mode',
    component: GameModeComponent,
  },
  {
    path: 'options',
    component: OptionsComponent,
  },
  {
    path: 'about',
    component: AboutComponent,
  },
  {
    path: 'one-player',
    component: OnePlayerComponent,
  },
  {
    path: 'two-players',
    component: TwoPlayersComponent,
  },
  {
    path: '**',
```

```
    component: NotFoundPageComponent,
    data: {menu: 'public', title: 'Page Not Found'},
  },
];

@NgModule({
  imports: [
    RouterModule.forRoot(routes, {
      initialNavigation: 'enabled',
      scrollPositionRestoration: 'enabled',
    }),
    BrowserAnimationsModule,
  ],

  exports: [RouterModule],
})
export class AppRoutingModule { }
```

**one-player.component.ts:**

```
import {Component, OnInit} from '@angular/core';
import {OptionsService} from '../options-service/options.service';
import {ChessOptions} from '../options-service/symbols';

declare const ChessBoard: any;
declare const Chess: any;

@Component({
  selector: 'app-one-player',
  templateUrl: './one-player.component.html',
  styleUrls: ['./one-player.component.scss']
})
export class OnePlayerComponent extend AbstractGameComponent implements OnInit
{

  constructor(private chessOptions: OptionsService) {
  }

  options: ChessOptions = this.chessOptions.options;
  minimaxDepth: number = this.options.aiLevel;

  ngOnInit(): void {

    let board;
```

```
  const game = new Chess();

  // do not pick up pieces if the game is over
  // only pick up pieces for White
  const onDragStart = (source, piece, position, orientation) => {
    if (game.in_checkmate() === true || game.in_draw() === true || game.game_over()
=== true) {
      console.log('Game over!');
      return false;
    }
  };


  // uses the minimax algorithm with alpha beta pruning to caculate the best move
  const calculateBestMove = () => {

    const possibleNextMoves = game.moves();
    let bestMove = -9999;
    let bestMoveFound;

    for (const possibleNextMove of possibleNextMoves) {
      game.move(possibleNextMove);
      const value = minimax(this.minimaxDepth, -10000, 10000, false);
      game.undo();
      if (value >= bestMove) {
        bestMove = value;
        bestMoveFound = possibleNextMove;
      }
    }
    return bestMoveFound;
  };


  // minimax with alhpha-beta pruning and search depth d = 3 levels
  const minimax = (depth, alpha, beta, isMaximisingPlayer) => {
    if (depth === 0) {
      return -evaluateBoard(game.board());
    }

    const possibleNextMoves = game.moves();
    const numPossibleMoves = possibleNextMoves.length;

    let bestMove = -9999;
    if (isMaximisingPlayer) {
      for (let i = 0; i < numPossibleMoves; i++) {
```

```
        game.move(possibleNextMoves[i]);
        bestMove = Math.max(bestMove, minimax(depth - 1, alpha, beta,
!isMaximisingPlayer));
        game.undo();
        alpha = Math.max(alpha, bestMove);
        if (beta <= alpha) {
          return bestMove;
        }
      }

    } else {
      for (let i = 0; i < numPossibleMoves; i++) {
        game.move(possibleNextMoves[i]);
        bestMove = Math.min(bestMove, minimax(depth - 1, alpha, beta,
!isMaximisingPlayer));
        game.undo();
        beta = Math.min(beta, bestMove);
        if (beta <= alpha) {
          return bestMove;
        }
      }
    }

    return bestMove;
  };


  // the evaluation function for minimax
  const evaluateBoard = chessBoard => {
    let totalEvaluation = 0;
    for (let i = 0; i < 8; i++) {
      for (let j = 0; j < 8; j++) {
        totalEvaluation = totalEvaluation + getPieceValue(chessBoard[i][j], i, j);
      }
    }
    return totalEvaluation;
  };


  const reverseArray = array => array.slice().reverse();

  const whitePawnEval =
    [
      [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0],
      [5.0, 5.0, 5.0, 5.0, 5.0, 5.0, 5.0, 5.0],
```

```
    [1.0, 1.0, 2.0, 3.0, 3.0, 2.0, 1.0, 1.0],
    [0.5, 0.5, 1.0, 2.5, 2.5, 1.0, 0.5, 0.5],
    [0.0, 0.0, 0.0, 2.0, 2.0, 0.0, 0.0, 0.0],
    [0.5, -0.5, -1.0, 0.0, 0.0, -1.0, -0.5, 0.5],
    [0.5, 1.0, 1.0, -2.0, -2.0, 1.0, 1.0, 0.5],
    [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
  ];

const blackPawnEval = reverseArray(whitePawnEval);

const knightEval =
  [
    [-5.0, -4.0, -3.0, -3.0, -3.0, -3.0, -4.0, -5.0],
    [-4.0, -2.0, 0.0, 0.0, 0.0, 0.0, -2.0, -4.0],
    [-3.0, 0.0, 1.0, 1.5, 1.5, 1.0, 0.0, -3.0],
    [-3.0, 0.5, 1.5, 2.0, 2.0, 1.5, 0.5, -3.0],
    [-3.0, 0.0, 1.5, 2.0, 2.0, 1.5, 0.0, -3.0],
    [-3.0, 0.5, 1.0, 1.5, 1.5, 1.0, 0.5, -3.0],
    [-4.0, -2.0, 0.0, 0.5, 0.5, 0.0, -2.0, -4.0],
    [-5.0, -4.0, -3.0, -3.0, -3.0, -3.0, -4.0, -5.0]
  ];

const whiteBishopEval = [
  [-2.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -2.0],
  [-1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -1.0],
  [-1.0, 0.0, 0.5, 1.0, 1.0, 0.5, 0.0, -1.0],
  [-1.0, 0.5, 0.5, 1.0, 1.0, 0.5, 0.5, -1.0],
  [-1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 0.0, -1.0],
  [-1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, -1.0],
  [-1.0, 0.5, 0.0, 0.0, 0.0, 0.0, 0.5, -1.0],
  [-2.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -2.0]
];

const blackBishopEval = reverseArray(whiteBishopEval);

const whiteRookEval = [
  [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0],
  [0.5, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.5],
  [-0.5, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -0.5],
  [-0.5, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -0.5],
  [-0.5, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -0.5],
  [-0.5, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -0.5],
  [-0.5, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -0.5],
  [0.0, 0.0, 0.0, 0.5, 0.5, 0.0, 0.0, 0.0]
];
```

```
const blackRookEval = reverseArray(whiteRookEval);

const evalQueen = [
  [-2.0, -1.0, -1.0, -0.5, -0.5, -1.0, -1.0, -2.0],
  [-1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -1.0],
  [-1.0, 0.0, 0.5, 0.5, 0.5, 0.5, 0.0, -1.0],
  [-0.5, 0.0, 0.5, 0.5, 0.5, 0.5, 0.0, -0.5],
  [0.0, 0.0, 0.5, 0.5, 0.5, 0.5, 0.0, -0.5],
  [-1.0, 0.5, 0.5, 0.5, 0.5, 0.5, 0.0, -1.0],
  [-1.0, 0.0, 0.5, 0.0, 0.0, 0.0, 0.0, -1.0],
  [-2.0, -1.0, -1.0, -0.5, -0.5, -1.0, -1.0, -2.0]
];

const whiteKingEval = [

  [-3.0, -4.0, -4.0, -5.0, -5.0, -4.0, -4.0, -3.0],
  [-3.0, -4.0, -4.0, -5.0, -5.0, -4.0, -4.0, -3.0],
  [-3.0, -4.0, -4.0, -5.0, -5.0, -4.0, -4.0, -3.0],
  [-3.0, -4.0, -4.0, -5.0, -5.0, -4.0, -4.0, -3.0],
  [-2.0, -3.0, -3.0, -4.0, -4.0, -3.0, -3.0, -2.0],
  [-1.0, -2.0, -2.0, -2.0, -2.0, -2.0, -2.0, -1.0],
  [2.0, 2.0, 0.0, 0.0, 0.0, 0.0, 2.0, 2.0],
  [2.0, 3.0, 1.0, 0.0, 0.0, 1.0, 3.0, 2.0]
];

const blackKingEval = reverseArray(whiteKingEval);


const getPieceValue = (piece, x, y) => {
  if (piece === null) {
    return 0;
  }

  const absoluteValue = getAbsoluteValue(piece, piece.color === 'w', x, y);

  if (piece.color === 'w') {
    return absoluteValue;
  } else {
    return -absoluteValue;
  }
};


const getAbsoluteValue = (piece, isWhite, x, y) => {
```

```
  if (piece.type === 'p') {
    return 10 + (isWhite ? whitePawnEval[y][x] : blackPawnEval[y][x]);
  } else if (piece.type === 'r') {
    return 50 + (isWhite ? whiteRookEval[y][x] : blackRookEval[y][x]);
  } else if (piece.type === 'n') {
    return 30 + knightEval[y][x];
  } else if (piece.type === 'b') {
    return 30 + (isWhite ? whiteBishopEval[y][x] : blackBishopEval[y][x]);
  } else if (piece.type === 'q') {
    return 90 + evalQueen[y][x];
  } else if (piece.type === 'k') {
    return 900 + (isWhite ? whiteKingEval[y][x] : blackKingEval[y][x]);
  }
};


const makeAImove = () => {
  const bestMove = calculateBestMove();
  game.move(bestMove);
  board.position(game.fen());
};


const onDrop = (source, target) => {
  // see if the move is legal
  const move = game.move({
    from: source,
    to: target,
    promotion: 'q'
  });

  // illegal move
  if (move === null) {
    return 'snapback';
  }

  // make legal move for black AI player
  window.setTimeout(makeAImove, 250);
};


const onMouseoverSquare = (square, piece) => {
  // get list of possible moves for this square
  const moves = game.moves({
    square,
```

```
      verbose: true
    });

    // exit if there are no moves available for this square
    if (moves.length === 0) {
      return;
    }
  };


  const onMouseoutSquare = (square, piece) => {
  };


  // update the board position after the piece snap
  // for castling, en passant, pawn promotion
  const onSnapEnd = () => {
    board.position(game.fen());
  };

  // images has different extensions
  const ext = this.options.chessStyle === 'rome' ? 'png' : 'svg';

  const cfg = {
    draggable: true,
    position: 'start',
    orientation: this.options.firstMove ? 'white' : 'black',
    pieceTheme: `assets/img/chess-pieces/${this.options.chessStyle}/{piece}.${ext}`,
    onDragStart,
    onDrop,
    onMouseoutSquare,
    onMouseoverSquare,
    onSnapEnd
  };
  board = ChessBoard('myBoard', cfg);

  this.updateTheme();

  // move first if user chose black
  if (!this.options.firstMove) {
    window.setTimeout(makeAImove, 250);
  }
}

updateTheme(): void {
```

```
    if (this.options.boardStyle === 'alternative') {
      document.querySelectorAll('.white-1e1d7').forEach(elem => (elem as
HTMLElement).style.backgroundColor = '#727FA2');
      document.querySelectorAll('.black-3c85d').forEach(elem => (elem as
HTMLElement).style.backgroundColor = '#C3C6BE');
    }
  }

}
```

**options.component.ts:**

```
import { Component, OnInit } from '@angular/core';
import {OptionsService} from '../options-service/options.service';
import {AiLevel, ChessOptions} from '../options-service/symbols';

@Component({
  selector: 'app-options',
  templateUrl: './options.component.html',
  styleUrls: ['./options.component.scss']
})
export class OptionsComponent implements OnInit {

  constructor(private optionsService: OptionsService) { }

  options: ChessOptions;
  aiLevel = AiLevel;

  ngOnInit(): void {
    this.updateOptions();
  }

  updateOptions(): void {
    this.options = this.optionsService.options;
  }

  setFirstMove(firstMove: boolean): void {
    this.optionsService.options = {firstMove};
    this.updateOptions();
  }

  setDifficult(aiLevel: AiLevel): void {
    this.optionsService.options = {aiLevel};
    this.updateOptions();
  }
```

```
  setChessStyle(chessStyle: 'classic' | 'rome'): void {
    this.optionsService.options = {chessStyle};
    this.updateOptions();
  }

  setBoardStyle(boardStyle: 'classic' | 'alternative'): void {
    this.optionsService.options = {boardStyle};
    this.updateOptions();
  }
}
```

**options.service.ts:**

```
import { Injectable } from '@angular/core';
import {AiLevel, ChessOptions} from './symbols';


@Injectable({
  providedIn: 'root'
})
export class OptionsService {

  constructor() { }

  private storageKey = 'chessOptions';
  private defaultOptions = {
    firstMove: true,
    aiLevel: AiLevel.easy,
    chessStyle: 'classic',
    boardStyle: 'classic',
  };

  get options(): ChessOptions {
    const saved = JSON.parse(localStorage.getItem(this.storageKey));
    return Object.assign(this.defaultOptions, saved);
  }
  set options(newOptions: ChessOptions) {
    const saved = JSON.parse(localStorage.getItem(this.storageKey));
    const updated = Object.assign(this.defaultOptions, saved, newOptions);
    const jsonOptions = JSON.stringify(updated);
    localStorage.setItem(this.storageKey, jsonOptions);
  }

}
```

**two-players.component.ts:**

```
import { Component, OnInit } from '@angular/core';
import {ChessOptions} from '../options-service/symbols';
import {OptionsService} from '../options-service/options.service';


declare const ChessBoard: any;
declare const Chess: any;

@Component({
  selector: 'app-two-players',
  templateUrl: './two-players.component.html',
  styleUrls: ['./two-players.component.scss']
})
export class TwoPlayersComponent extend AbstractGameComponent implements
OnInit {

  constructor(private chessOptions: OptionsService) { }

  options: ChessOptions = this.chessOptions.options;

  ngOnInit(): void {
    let board = null;
    const game = new Chess();

    function onDragStart(source, piece, position, orientation): boolean {
      // do not pick up pieces if the game is over
      if (game.game_over()) { return false; }

      // only pick up pieces for the side to move
      if ((game.turn() === 'w' && piece.search(/^b/) !== -1) ||
        (game.turn() === 'b' && piece.search(/^w/) !== -1)) {
        return false;
      }
    }

    function onDrop(source, target): string {
      // see if the move is legal
      const move = game.move({
        from: source,
        to: target,
        promotion: 'q' // NOTE: always promote to a queen for example simplicity
      });
```

```
    // illegal move
    if (move === null) { return 'snapback'; }

    updateStatus();
  }

// update the board position after the piece snap
// for castling, en passant, pawn promotion
  function onSnapEnd(): void {
    board.position(game.fen());
  }

  function updateStatus(): void {
    let status = '';

    let moveColor = 'White';
    if (game.turn() === 'b') {
      moveColor = 'Black';
    }

    // checkmate?
    if (game.in_checkmate()) {
      status = 'Game over, ' + moveColor + ' is in checkmate.';
    }

    // draw?
    else if (game.in_draw()) {
      status = 'Game over, drawn position';
    }

    // game still on
    else {
      status = moveColor + ' to move';

      // check?
      if (game.in_check()) {
        status += ', ' + moveColor + ' is in check';
      }
    }

    // console.log(status, game.fen(), game.pgn());
  }

  // images has different extensions
```

```
    const ext = this.options.chessStyle === 'rome' ? 'png' : 'svg';

    const a = {
      chess24_board_theme: ['#9E7863', '#633526'],
      metro_board_theme: ['#EFEFEF', '#FFFFFF'],
      leipzig_board_theme: ['#FFFFFF', '#E1E1E1'],
      wikipedia_board_theme: ['#D18B47', '#FFCE9E'],
      dilena_board_theme: ['#FFE5B6', '#B16228'],
      uscf_board_theme: ['#C3C6BE', '#727FA2'],
      symbol_board_theme: ['#FFFFFF', '#58AC8A'],
    }

    const config = {
      draggable: true,
      position: 'start',
      pieceTheme: `assets/img/chess-pieces/${this.options.chessStyle}/{piece}.${ext}`,
      boardTheme: a.metro_board_theme,
      onDragStart,
      onDrop,
      onSnapEnd
    };
    board = ChessBoard('myBoard', config);

    updateStatus();
    this.updateTheme();
  }

  updateTheme(): void {
    if (this.options.boardStyle === 'alternative') {
      document.querySelectorAll('.white-1e1d7').forEach(elem => (elem as
HTMLElement).style.backgroundColor = '#727FA2');
      document.querySelectorAll('.black-3c85d').forEach(elem => (elem as
HTMLElement).style.backgroundColor = '#C3C6BE');
    }
  }
}
```

ДОДАТОК 2
Копії публікацій за темою магістерської роботи

ядром, коли користувач обирає відповідний режим. Всі елементи, що розташовуються на сторінці в грі на одного гравця, контролюються цим класом. Він відповідає за стани гри, валідацію ходів та ходи шахового бота. Бот керується простим алгоритмом, щоразу вибудовуючи дерево рішень з матриці ходів, а потім прораховуючи оптимальність ходу; глибина прорахування визначається налаштуваннями складності.

TwoPlayersComponent. Клас TwoPlayersComponent має схожу логіку до OnePlayerComponent, з певними відмінностями, що робить його простішим. Оскільки для двох гравців не потрібний шаховий бот, така логіка в ньому відсутня, а також зняте обмеження на вибір кольору гравця.

OptionsService. Клас OptionsService відповідає за збереження налаштувань користувача та роботу з LocalStorage. Примірник цього класу створюється при виклиці конструктора на одному з класів, в якому він інжектований. Клас має гетери та сетери для посилання на об'єкт options, що дозволяє в одному й тому ж методі отримувати значення та перезаписувати їх. Це значно спрощує модель налаштувань. Для збереження даних локально клас звертається до визначеного в Angular класу LocalStorage (в такому випадку додаткова інжекція не потрібна).

Додаткові класи, використані в додатку: AboutComponent, NotFoundPageComponent, GameModeComponent та інші. Вони класи відповідають за представлення певної інформації на сторінці та не містять в собі специфічної логіки.

Компоненти з бібліотек, використані у додатку:

- Buttons (Angular Material) - компонент кнопки, містить прості визначені стилі для відображення та користувацьких подій.
- Accordion (Angular Material) – компонент для лаконічного відображення змісту. Містить слухачів користувацьких подій для динамічного згортання та розгортання контенту.

Компоненти інтерфейсу:

- Button – тег віртуальної кнопки.
- Image – компонент для розташування зображень в об'єкті інтерфейсу.
- P – компонент тексту. Використовується для відображення тексту.
- Div – компонент змісту. Використовується для групування та позиціонування будь-яких елементів

В результаті даного дослідження був розроблений вебдодаток (гра) на базі JavaScript фреймворку Angular з використанням бібліотек Angular Material та Bootstrap і методології БЕМ.

Список використаних джерел

1. 10 лучших AR приложений в 2019 году [Електронний ресурс] // design glory. – 2019. – Режим доступу до ресурсу: https://design-glory.com/1975/10-luchshih-ar-prilozhenij-dlya-android-i-ios-v-2019-godu.

2. Fundamental concepts [Електронний ресурс] // ARCore – Режим доступу до ресурсу: https://developers.google.com/ar/discover/concepts#motion_tracking

програм, які вийшли за межі науково-дослідних лабораторій і фінансових відділів.

**Постановка завдання**

Метою даного дослідження є розробка Angular додатку та ознайомлення з мовами розробки високого рівня, що містить в собі аналіз проблем, які вони вирішують, в першу чергу сучасних тенденцій принципівпобудови додатків. Розглянуті можливі альтернативи та приклади використання архітектури в типовому додатку. Якість розробки визначається відповідністю розробленого програмного продукту до вимог, які були закладені на стадії проєктування системи. Всі вимоги до додатків, в тому числі і вебдодатків, поділяють на функціональні і не функціональні.

Функціональні вимоги визначають ту функціональність системи, яку розробники повинні побудувати, щоб користувачі змогли виконати свої завдання в рамках своїх бізнес-процесів. Нефункціональні вимоги являють собою опис характеристик додатки, важливих для користувача при роботі з системою.

**Основна частина**

Сучасні принципи побудови вебдодатків:

- Рендерінг на сервері здійснюється не заради SEO, а для продуктивності. Варто приймати до уваги додаткові запити для отримання скриптів, стилів і наступні запити до API. Також, варто звернути увагу на використання методу HTTP 2.0 Push.
- JavaScript дозволяє повністю приховати мережеву затримку. Використання даного підходу, як принципу дизайну, дозволяє звільнити програму майже від усіх індикаторів завантаження і повідомлень "зачекайте". PJAX та TurboLinks обмежують можливості по збільшенню суб'єктивної швидкості інтерфейсу.
- Сучасні технології розробки дозволяють точно керувати обміном даними з сервером. Обробка помилок, повторні запитах на користь клієнта, синхронізації даних у фоновому режимі й збереженні кеша в офлайні стали новим стандартом «де-факто» в розробці вебдодатків.
- Коли на сервері оновлюються дані, користувача варто повідомляти без затримки. Така форма підвищення продуктивності звільняють від необхідності здійснювати додаткові дії (натискати F5, оновлювати сторінку). Хоча вона і провокує нові проблеми, що потребують вирішення: управління повторним з'єднанням, відновлення стану.
- Відсутність керування історією навігації в браузері створює нові проблеми. Варто впевнитися, що додаток відповідає очікуваній

поведінці щодо навігації в історії. Гарною практикою вважається збереження власного кешу для швидкого відклику. Кнопка «Назад» повинна працювати швидко; користувачі не очікують занадто великої зміни даних.

- Сучасним підходом вважаєтьсяпосилання через pushповідомлення тільки даних, але й коду. Варто уникати помилок API і збільшувати продуктивність. Stateless DOM є гарним методом безболісного ре-рендерусторінки.

Всі ці підходи закладені в платформі Angular. Він представляє не тільки інструменти, але і шаблони дизайну для створення та обслуговування проєкту.

Правильнапідтримка Angular додатказвільняє відплутанини класів і методів, які,зазвичай, складно правити і ще складніше тестувати. Angular побудований на TypeScript, який, своєю чергою, покладається на ES6. Це означає зниження порогу входу, при цьому JSрозширюється функціямизі статичною типізацією, інтерфейсами, класами, просторами імен, декораторами й т.д.

Компоненти ізольовані. Платформа майже повністю прибирає жорсткий зв'язок між різними компонентами програми. Ін'єкція проходить подібно до NodeJS, що дозволяє легко замінювати компоненти.

Всі маніпуляції з DOM проходять там, де повинні. У Angular уявлення і логіка програми не пов'язані, що сильно спрощує розмітку та підвищує її потенціал до повторного використання.

Angular ретельно протестований і підтримує unit тести та наскрізне тестування за допомогою спеціалізованих інструментів, наприклад Jasmine і Protractor.

**Висновки**

Проведено дослідження сучасних принципів побудови вебдодатків, виокремлено сучасні підходи та методи побудови високорівневих абстракцій. Зроблено порівняння готових бібліотек та фреймворків; розглянуто основні платформи розробки програмних продуктів.

Проаналізувавши останні дослідження, було зроблено висновок, що веб залишається самим багатогранним середовищем передачі інформації. Додавання динаміки на наші вебсторінки зумовлює необхідність переконатися, що важливі принципи веба були збережені.

Нові унікальні можливості надає JavaScript. Використаннясучасних підходів забезпечує найкращий досвід роботи для користувачів найбільш вільної платформи з існуючих.

Розроблено вебзастосунок з використанням платформи Angular, що реалізовує клієнтську логіку для онлайн гри «Шахи». Додаток

УДК 519.246.8(075.8)

# ПРИНЦИПИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ З ВИКОРИСТАННЯМ ПЛАТФОРМИ ANGULAR

А.Р. Снєсарь, магістрант
*Київський національний університет технологій та дизайну*
Т.І. Демківська, кандидат технічних наук, доцент
*Київський національний університет технологій та дизайну*

Ключові слова: шахи, гра, вебдодаток, платформа, фреймворк, бібліотека, івенти, Angular, Material, івенти.

Основною метою дослідницького проекту є розробка вебдодатку (гри) для вебраузера на базі платформи Angular з використанням бібліотек Angular Material та Bootstrap.

Головними функціональними особливостями реалізованого додатку є:

– Додаток має отримати повноцінну функціональність гри:
– Зручний і зрозумілий інтерфейс користувача;
– Можливість зміни опцій та їх збереження навіть після закінчення ігрової сесії;
– Архітектура, що забезпечує повний ігровий цикл;
– Динамічний рендерінг об'єктів при зміні станів гри;
– Інформацію щодо використаних інструментів та підходів.

У проекті були використані наступні програмні засоби:

- Інтегроване середовище розробки (IDE) Webstorm;
- Онлайн редактор для створення та редагування макетів Figma;
- Система контролю версій Git;
- Растровий редактор зображень Photoshop;
- Веббраузери (для розробки та тестування): Chromium, Google Chrome, Opera, Firefox.

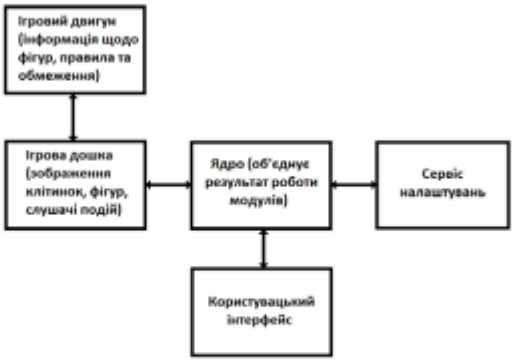



Рисунок 1 – Блок-схема

В процесі дослідження було створено та використано наступні класи:

OnePlayerComponent. Клас OnePlayerComponent представляє об'єкт, який є частиною ігрового процесу. Примірник цього класу викликається

СНЄСАРЬ А.С., ДЕМКІВСЬКА Т.І.

# РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ КОМП'ЮТЕРНОЇ ГРИ «ШАХИ» З ВИКОРИСТАННЯМ ПЛАТФОРМИ ANGULAR

SNESAR A.S., DEMKIVSKA T.I.

**DEVELOPMENT OF SOFTWARE FOR COMPUTER GAME "CHESS" USING THE ANGULAR PLATFORM**

*Advances in computer technology have defined the process of emergence of new various sign systems for writing algorithms of programming languages. The meaning of such a language is the simplification of program code.*

*High-level programming languages have been designed for platform independence in the essence of algorithms. Platform dependency is shifted to instrumental programs - translators that compile text written in a high-level language into elementary machine instructions (instructions). Therefore, for each platform, a platform-unique translator is developed for each high-level language. Angular is a striking example of combining modern programming patterns and a high level of abstraction.*

*It is now expanding into other areas such as mobile apps and Progressive Web Apps.Angular introduces Google's framework for building client-side applications. First of all, it is aimed at developing SPA solutions (Single Page Application). In this regard, Angular inherits from another AngularJS framework. At the same time Angular is not a new version of AngularJS, but a fundamentally new framework.*

*Angular provides functionality such as two-way binding to dynamically change data in one place in the interface when model data changes in another, templates, routing, and so on.*

## Вступ

Прогрес комп'ютерних технологій визначив процес появи нових різноманітних знакових систем для запису алгоритмів мов програмування. Сенс появи такої мови - спрощення програмного коду.

Попри на стрімке зростання числа мов програмування, програмне забезпечення істотно відставало в своєму розвитку від технічної бази комп'ютерів. У той час як в області виробництва апаратного забезпечення спостерігалось неухильне зростання продуктивності й зниження ціни, вартість програмного забезпечення продовжувала збільшуватися. Іноді витрати на програмне забезпечення великих нових обчислювальних систем доходили до 80% від їх загальної вартості.

Головним чинником, що сприяв розв'язанню цієї проблеми, стало створення нових мов програмування, так званого високого рівня, що характеризується складністю абстракцій при взаємодії з ними.

З появою цих мов програмісти отримали можливість більше часу приділяти вирішенню конкретної задачі, не відволікаючись на вельми тонкі питання організації самого процесу виконання завдання на машині. Крім того, поява цих мов ознаменувала перший крок на шляху створення

ДОДАТОК 3

Презентація дипломної магістерської роботи

## СФЕРИ ЗАСТОСУВАННЯ ВЕБТЕХНОЛОГІЙ

- Управління державою
- Банківська справа
- Медицина
- Розваги та послуги
- Військова справа
- Торгівля та облік
- Виховання та освіта
- Зв'язок та комунікації
- Мистецтво

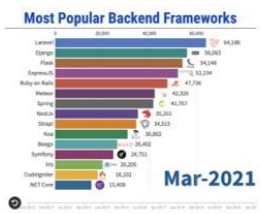
..та багато іншого

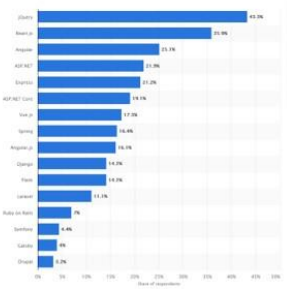
## ТЕХНОЛОГІЇ СТВОРЕННЯ ВЕБСАЙТІВ

Клієнтська частина: Vue.js, React, Angular, jQuery, Svelte, Foundation, Bootstrap, Ember, WordPress та інші

Серверна частина: Laravel, Django, Flask, Express.js, Ruby, Meteor, Spring, Nest.js та інші

Інструменти розробки та підтримки: Git, cybernetes, docker, Webstorm, VSCode, Figma, Photoshop та інші



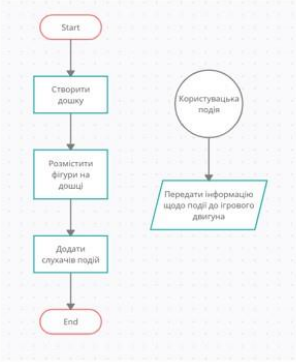
# ANGULAR ТА ЙОГО ОСОБЛИВОСТІ

- Single Page Application
- Підтримка PWA (Progressive Web Apps) та SSR (Server Side Rendering)
- Визначена робота з шаблонами
- Визначені можливі види компонентів
- Інтегрована навігація
- Інтегрована робота з формами
- Інтегрована робота з анімаціями (та підтримка нативних css анімацій)
- Інтегрована бібліотека RxJs (зручна робота з асинхронними операціями)
- Інтегрований TypeScript
- Підтримка бібліотек глобального стану додатку



# МОДУЛІ СТВОРЕНОГО ДОДАТКУ

- Ігровий двигун
- Ігрова дошка
- Сервіс налаштувань
- Користувацький інтерфейс
- Центральний сервіс взаємодії модулів

# СТОРІНКИ РОЗРОБЛЕНОГО ВЕБСАЙТУ

- Головна сторінка
- Сторінка гри для одного гравця
- Сторінка гри для двох гравців
- Сторінка налаштувань
- Інформаційна сторінка