

Міністерство освіти і науки України
Державний університет «Одеська політехніка»
Інститут комп'ютерних систем
Кафедра інформаційних систем

Баранюк Дмитро Андрійович,
студент групи АІ-202

КУРСОВА РОБОТА

Тема

Тема: веб-додаток фітнес тренування у домашніх умовах

Спеціальність:

122 Комп'ютерні науки

Освітня програма:

Комп'ютерні науки

Керівник:

Годовиченко Микола Анатолійович,
кандидат технічних наук, доцент

Одеса – 2022

ЗМІСТ

Анотація.....	4
Вступ.....	5
1 Аналіз задачі тренувань у домашніх умовах	6
1.1 Аналіз предметної області	6
1.2 Огляд додатків у сфері фітнес тренувань	6
1.2.1 Додаток Домашні тренування	7
1.2.2 Додаток Nike Training Club	8
1.2.3 Додаток Adidas Training	10
1.3 Вибір стеку технологій для розробки інформаційної системи	11
1.3.1 Вибір архітектурного шаблону	11
1.3.2 Вибір стеку технологій для клієнтської частини.....	13
1.3.3 Вибір стеку технологій для серверної частини.....	14
1.4 Висновки до першого розділу	15
2 Проектування інформаційної системи	16
2.1 Мета та задачі інформаційної системи.....	16
2.2 Типи користувачів в системі.....	16
2.3 Визначення функціональних вимог до інформаційної системи	19
2.4 Проектування структури додатку.....	21
2.5 Проектування сценаріїв	22
3 Реалізація інформаційної системи.....	24
3.1 Структура програмного проєкту	24
3.2 Реалізація основних алгоритмів	26
3.3 Розгортання додатку та інструкція користувача	27
Висновки.....	33

Перелік використаних джерел.....	34
Додаток А Програмний код.....	35

АНОТАЦІЯ

Дана робота присвячена розробці інформаційної системи домашніх тренувань. В рамках виконання даної роботи були проаналізовані існуючі додатки у сфері фітнесу, були сформовані вимоги до основного функціоналу інформаційної системи. Були обрані технології для проектування та реалізації данної системи, технології розробки клієнтської та серверної частини. Було проведено проектування розроблюваної інформаційної системи, були визначені функціональні вимоги, а також були розроблені макети клієнтської частини інформаційної системи. Згідно із проектною документацією була розроблена інформаційна система.

ABSTRACT

This work is devoted to the development of information system of home training. As part of this work, the existing applications in the field of fitness were analyzed, the requirements for the main functionality of the information system were formed. Technologies for design and implementation of this system, technologies for client and server development were selected. The development of the developed information system was carried out, the functional requirements were determined, and also the models of the client part of the information system were developed. According to the project documentation, an information system was developed.

ВСТУП

Останнім часом культ здорового образу життя набув жаленого розголосу. Майже усі країни, а саме країни розвиненого світу, намагаються прив'язати його своїм громадянам.

Правильне харчування, особиста гігієна, здоровий сон, фізичні вправи і багато чого іншого є неймовірно корисними для здоров'я людини. Під час карантину з'явилася проблема з фітнес складовою. Через карантинні обмеження встановлених представниками влади багатьох країн не можна було відвідувати спортивні зали, ігрові майданчики, спортивні секції та багато чого іншого. Люди майже цілодобово сиділи вдома і мали нерухомий стиль життя. Тому стали дуже популярними різного роду фітнес додатки, для легкого тренування в домашніх умовах та підтримки своєї фізичної складової.

Враховуючи дані обставини, актуальною є тема розробки інформаційної системи, яка дозволяє займатися фітнесом у домашніх умовах, і має багато різноманітних типів та видів тренувань для усіх охочих, від початківців до професійних спортсменів.

Метою кваліфікаційної роботи є розробка інформаційної системи тренувань у домашніх умовах.

Для досягнення цієї мети необхідно виконати наступні задачі:

- проаналізувати існуючі додатки для спортсменів та новачків, які дозволяють спланувати різноманітні тренування за видами (наприклад кардіо або нарощування м'язів) та рівнями складності;
- обрати технології для проектування та реалізації інформаційної системи для планування фітнес тренувань;
- визначити функціональні вимоги та здійснити проектування інформаційної системи;
- реалізувати спроектовану інформаційну систему
- переконатися у тому, що розроблена система задовольняє вимогам, що були визначені на етапі проектування.

1 АНАЛІЗ ЗАДАЧІ ТРЕНУВАНЬ У ДОМАШНІХ УМОВАХ

1.1 Аналіз предметної області

В останні роки наш світ кардинально змінився. Через карантинні обмеження багато сфер діяльності були на грані розвалу і фітнес став одним із них. Не можна було відвідувати спортклуби, секції, займатися на спортивних майданчиках з великою кількістю людей, потрібно було зберігати соціальну дистанцію, масковий режим і багато чого іншого. Тому кожна людина почала шукати альтернативу і рішенням стали звичайно ж фітнес додатки для тренувань на дому. Звичайно вони і раніше були досить популярними, аля після початку 2020 року на платформах типу Google Play Store та Apple Store цифри кількості завантажень виросли у декілька разів. Ними почали користуватися як звичайні люди, без раніше фізичної підготовки, так і професійні спортсмени.

Популярність кроссфіт платформ зростає вже не перший рік. Аналізуючи, як буде запускатися та інтегруватися черговий мобільний тренер, розробники намагаються конкурувати, пропонуючи додаткові функції та різні модернізації. Кращі додатки для фітнесу здатні не тільки контролювати дані, але й видавати слушні поради користувачеві. Система «підвантажує» інструкції, новини, статистику та інше. Програм для смарт годинників та фітнес браслетів десятки, але деякі екземпляри заслуговують на особливу увагу. Якщо б декілька років тому людство дізналося наскільки ця сфера розвинеться настільки швидко і сильно, воно б навіть не повірило у це.

1.2 Огляд додатків-аналогів у сфері фітнесу

В результаті аналізу ринку додатків для тренувань, було виявлено, що на даний момент є чимало відомих комерційних додатків, призначених конкретно для ринку. Тому, в якості додатків аналогів були обрані три популярних додатки з платформи Google Play Store з категорії «Фітнес». Такими додатками виявилися:

- Домашні тренування – додаток для тренувань у домашніх умовах;
- Nike Training Club – додаток для тренувань у домашніх та вуличних умовах;
- Adidas Training – додаток для тренувань у домашніх та вуличних умовах.

Проведемо огляд кожної програми-аналога і спробуємо дослідити основний функціонал кожного з них.

1.2.1 Додаток Домашні тренування[2]

Додаток Домашні тренування є, одним з найпопулярніших фітнес додатків (рис. 1.1). Він має особливу популярність в Україні. Його особливістю є велика різноманітність фітнес тренувань, які містять гімнастичну складову. Також слід виокремити наявність анімаційної інструкції до кожної вправи. Серед інших функцій програми слід згадати графік для відстеження власного прогресу та налаштування нагадувань про власні тренування.

В результаті дослідження роботи програми і особистого досвіду роботи з додатком, були виокремлені переваги і недоліки програми.

Переваги додатку Домашні тренування:

- велика кількість гімнастичних вправ на розтяжку м'язів для подальшого релаксу;
- анімовані інструкції до кожної вправи;
- зручна навігація;
- малий розмір додатку;
- не потребує особливих характеристик від пристрою.

Недоліки програми:

- супровід диктатора є неякісним на майже усіх мовах, судячи з відгуків користувачів різних країн;
- не має вибору тем для користувача, існує лише стандартна червона, яка на думку користувачів є досить агресивна.

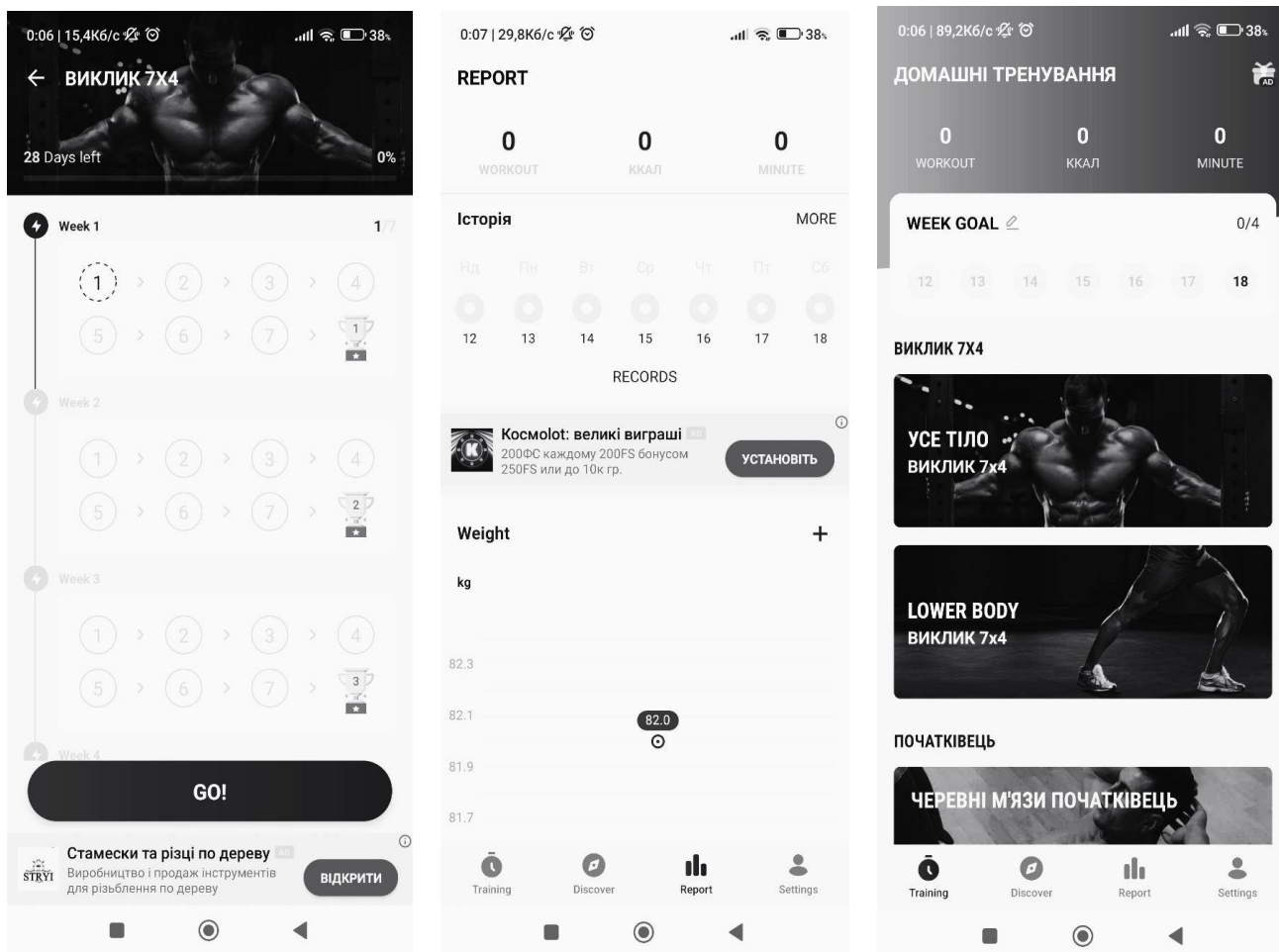


Рисунок 1.1 – Знімки екрану додатку «Домашні тренування»

1.2.2 Додаток Nike Training Club[3]

Додаток Nike Training Club також є одним з найпопулярніших фітнес додатків в усьому світі (рис. 1.2). Його особливістю є велика різноманітність як фітнес тренувань, так і йоги. Також слід виокремити наявність відео інструкції з реальними тренерами до кожної вправи. Серед інших функцій програми слід також згадати налаштування тренувань в залежності від вашої активності протягом дня та зручних для вас вправ та часу доби для їх виконання.

В результаті дослідження роботи програми і особистого досвіду роботи з додатком, були виокремлені переваги і недоліки програми.

Переваги додатку Nike Training Club:

- відео інструкції з реальними тренерами;

- велика різноманітність тренувань та йоги;
- зручна навігація;
- зручне спостереження за власним прогресом;
- наявність преміум версії додатку;
- підлаштування тренувань під власний ритм життя.

Недоліки програми:

- список країн на чий ринок спрямований додаток є малим, як приклад у додатку немає України у списку для вибору країни, та немає української версії продукту і багатьох інших;
- у чотири рази об'ємніший за попередній розглянутий додаток;
- потребує майже передостанніх версій операційних систем для своєї підтримки.

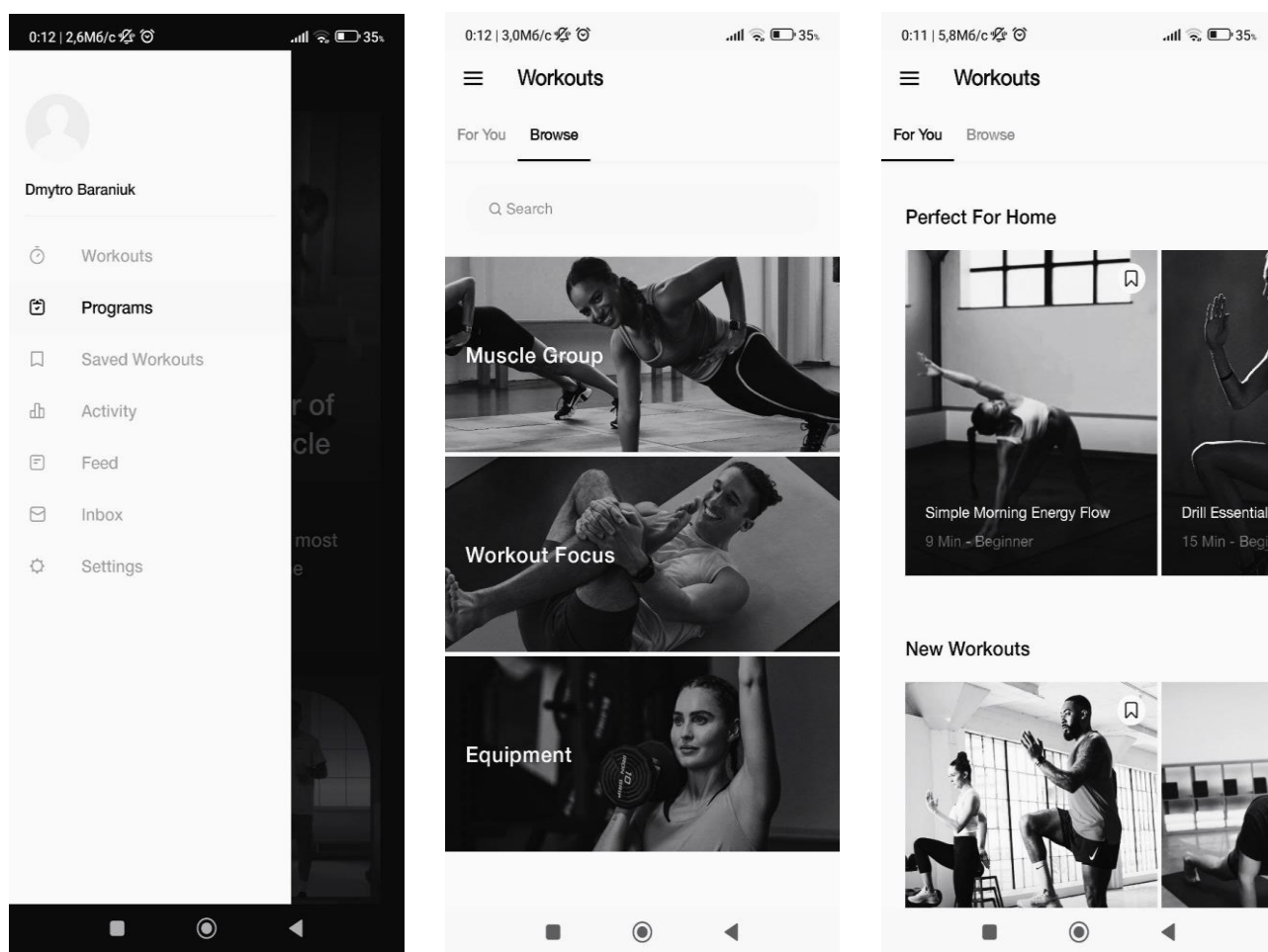


Рисунок 1.2 – Знімки екрану додатку Nike Training Club

1.2.3 Додаток Adidas Training[4]

Додаток Adidas Training також є одним з найпопулярніших фітнес додатків у розвиненому світі (рис. 1.3). Його особливістю є бездоганна навігація, як для такого роду додатку з величезним об'ємом інформації. Також слід виокремити велику різноманітність тренувань як у домашніх умовах, так і на вулиці і в тренажерному залі. Серед інших функцій програми слід також згадати, як і в попередньому додатку, налаштування тренувань в залежності від вашої активності протягом дня та зручних для вас вправ та часу доби для їх виконання.

В результаті дослідження роботи програми і особистого досвіду роботи з додатком, були виокремлені переваги і недоліки програми.

Переваги додатку Nike Training Club:

- відео інструкції з реальними тренерами;
- велика різноманітність тренувань та йоги;
- дуже зручна навігація;
- зручне спостереження за власним прогресом;
- наявність преміум версії додатку;
- підлаштування тренувань під власний ритм життя;
- не потребує особливих характеристик від пристрою.

Недоліки програми:

- немає української версії продукту, і багатьох країн також;
- майже у три рази об'ємніший за перший додаток.

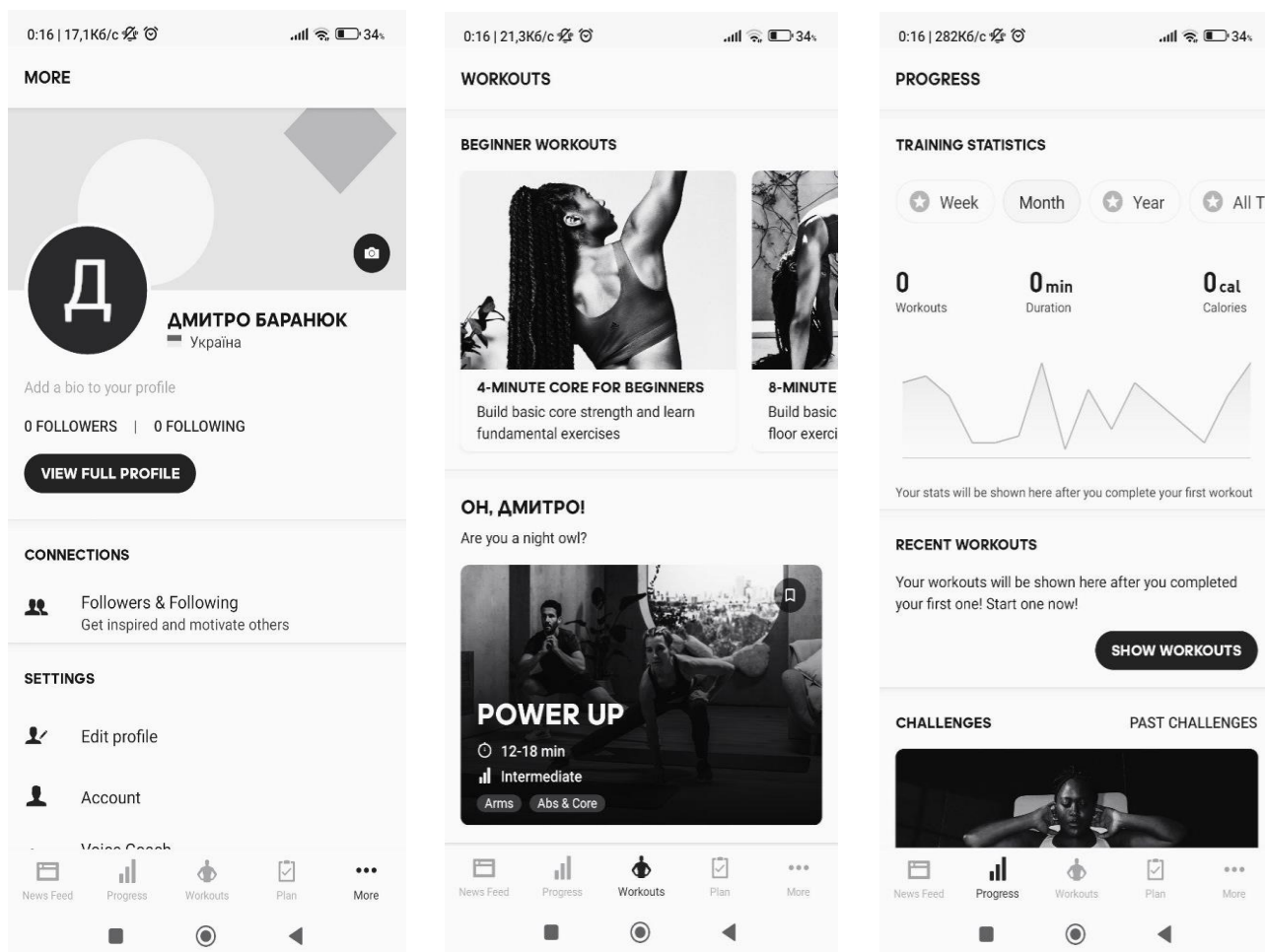


Рисунок 1.3 – Знімки екрану додатку Adidas Training

1.3 Вибір стеку технологій для розробки інформаційної системи

Виходячи зі сформованих базових вимог до функціоналу додатка, можна провести вибір стеку технологій, які дозволять реалізувати інформаційну систему тренувань в домашніх умовах.

1.3.1 Вибір архітектурного шаблону

Одним з можливих архітектурних шаблонів є клієнт-серверна система, яку можна представити триланковою архітектурою (рис. 1.4).

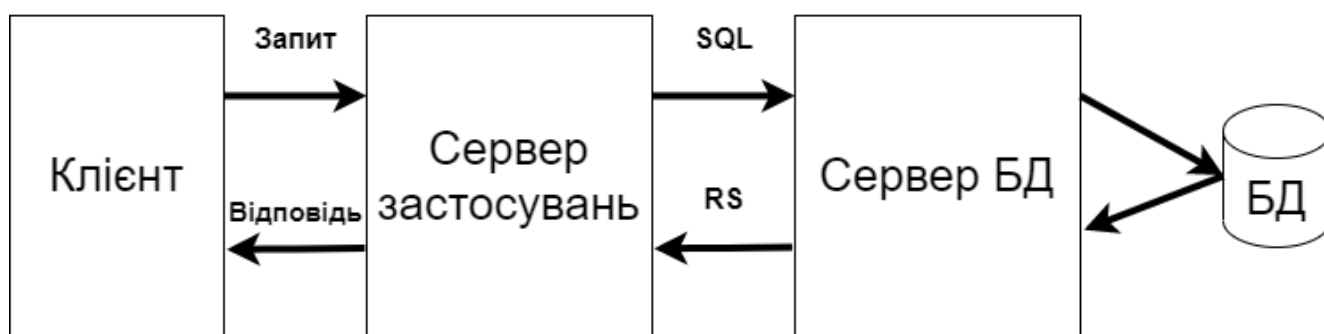


Рисунок 1.3 – Логічна схема класичної триланкової архітектури

Компоненти триланкової архітектури:

- клієнт – цей компонент відповідає за подання даних кінцевому користувачеві;
- виділений сервер додатків – тут міститься бізнес-логіка програми;
- сервер БД – надає запитувані дані.

Трирівнева клієнт-серверна архітектура, яка почала розвиватися з середини 90-х років, передбачає відділення прикладного рівня від управління даними. Відокремлюється окремий програмний рівень, на якому зосереджується прикладна логіка застосунку. Програми проміжного рівня можуть функціювати під управлінням спеціальних серверів застосунків, але запуск таких програм може здійснюватися і під управлінням звичайного вебсервера. Нарешті, управління даними здійснюється сервером даних.

Для роботи з системою користувач використовує стандартне програмне забезпечення — звичайний браузер. Це позбавляє його необхідності завантажувати та інсталиувати спеціальні програми (хоча інколи така необхідність все-таки виникає). Але користувачеві слід надати в розпорядженні інтерфейс, який дозволяв би йому взаємодіяти з системою і формувати запити до неї. Форми, що визначають цей інтерфейс, розміщуються на вебсторінках та завантажуються разом з ними.

Вебглядач формує запит та пересилає його до сервера, який здійснює обробку. При необхідності сервер викликає серверні програмні модулі, які забезпечують обробку запиту і в разі потреби звертаються до сервера даних. Сервер даних здійснює операції з даними, що зберігаються в системі та складають її

інформаційну основу. Зокрема, він може здійснити вибірку з інформаційної бази відповідно до запиту та передати її модулю проміжного рівня для подальшої обробки. Дані, з якими працює сервер даних, найчастіше організовані як реляційна база даних.

Найчастіше вебсервер і серверні модулі проміжного рівня розміщуються на одній машині, хоч і являють собою окремі і логічно незалежні програмні модулі.

1.3.2 Вибір стеку технологій для клієнтської частини

Як клієнт в даній інформаційній системі було вибрано веб-додаток. Цей вибір обумовлений тим, що користувачі у переважній більшості будуть використовувати додаток вдома, зі стабільним інтернет зв'язком.

Використання веб-додатка замість мобільного додатка обумовлено тим, що веб-додаток не використовує вбудовану пам'ять пристрою. Таким чином, клієнт повинен мати тільки вільний доступ до інтернету. Також розробка даного продукту означає, що немає ніякої різниці, яка операційна система використовується пристроєм і не буде потреби в особливих можливостей пристрою, який буде використовуватися.

До того ж веб-додаток має більше можливостей, щодо розміщення реклами, а це прямо сказується на прибуток проєктованого продукту. Дохід від використання реклами у додатках є одним з найбільших. Також, якщо користувачам реклама набридає, є можливість придбати преміум підписку, в якій не буде реклами, що в свою чергу також є одним з основних видів доходів проєктованого продукту.

Також одним з плюсів веб-додатку є менше витрачання часу на розробку, якщо ми будемо порівнювати з часом розробки мобільного додатку. А це є гроші замовника, який платить заробітну платню працівникові. І не можна не згадати також відносну легкість планових модифікацій продукту, які будуть проходити регулярно після його реалізації.

1.3.3 Вибір стеку технологій для серверної частини

У якості фреймворку для реалізації серверної частини був обраний Java Spring [5]. Вибір пав на даний фреймворк впершу чергу через використання мови програмування Java, яку ми вивчали протягом двох семестрів і безпосередньо через ознайомлення з ним у поточному семестрі курсу дисципліни «об'єктно-орієнтоване програмування».

Spring забезпечує вирішення багатьох завдань, з якими стикаються Java-розробники та організації, які хочуть створити інформаційну систему, що базується на платформі Java. Через широку функціональність важко визначити найбільш значущі структурні елементи, у тому числі він складається. Spring не повністю пов'язаний з платформою Java Enterprise, незважаючи на масштабну інтеграцію з нею, що є важливою причиною його популярності.

Spring, ймовірно, найбільш відомий як джерело розширень (features), необхідних для ефективної розробки складних бізнес-додатків поза великоваговими програмними моделями, які історично були домінуючими в промисловості. Ще одна його перевага в тому, що він ввів функціональні можливості, що раніше не використовуються, в сьогоденні панівні методи розробки, навіть поза платформою Java.

Цей фреймворк пропонує послідовну модель і робить її застосовною до більшості типів програм, які вже створені на основі платформи Java. Вважається, що Spring реалізує модель розробки, засновану на найкращих стандартах індустрії, і робить її доступною у багатьох сферах Java.

Модулі Spring, які допоможуть розробити проєктований продукт:

- Spring Core - ядро платформи, надає базові засоби - управління компонентами (бінами, beans), впровадження залежностей, MVC фреймворк, транзакції, базовий доступ до БД;
- Spring MVC - архітектура паттерна Model-View-Controller;
- Spring Data - доступ до даних БД, сховища тощо;
- Spring Security - авторизація та аутентифікація.

Spring Boot[6] дозволяє легко створювати автономні, виробничі програми на основі Spring, які можна «просто запускати». Розробники дотримуються упевненого погляду на платформу Spring та сторонні бібліотеки, щоб користувачі могли розпочати роботу з мінімальною метушнею. Більшість програм Spring Boot потребують мінімальної конфігурації Spring.

Особливості:

- Створюйте окремі програми Spring;
- має у собі Tomcat, Jetty або Undertow безпосередньо (не потрібно розгортати файли WAR);
- надає упевнені залежності «початківця», щоб спростити конфігурацію збірки;
- За можливості автоматично налаштовує бібліотеки Spring та сторонніх розробників;
- надає готові для виробництва функції, такі як показники, перевірки справності та зовнішні конфігурації;
- абсолютно без генерації коду та без вимог до конфігурації XML.

1.4 Висновки першого розділу

В даному розділі була проаналізована предметна область – фітнес тренування у домашніх умовах. Було виявлено, що дана сфера не є новою проте в останні кілька років пережила своє переродження, тому стрімко розвивається і має величезний попит серед людей.

Був проведений аналіз найбільш популярних мобільних додатків у раніше згаданих сфері. У результат порівняння переваг та недоліків цих додатків було виявлено, що їх розробники дуже швидко реагують на потреби ринку і намагаються максимально відповідати йому.

У ході вище зазначених аналізів були обрані технології для реалізації інформаційної системи. Була обрана трирівнева клієнт-серверна архітектура. У якості клієнтської частини був обраний веб-додаток, який завдяки браузеру надасть

доступ до роботи з ним користувачам з різних машин, з різними операційними системами та індивідуальними характеристиками. В результаті аналізу було вибрано технологію Java Spring Boot, за допомогою якої буде розроблено інформаційну систему.

2 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Мета та задачі інформаційної системи

Розроблювана інформаційна система являє собою веб-додаток для фітнес тренувань у домашніх умовах.

Мета інформаційної системи: додаток допомагає користувачам тренуватися у домашніх умовах, слідкувати за своїм фізичним станом, дозволяє слідкувати за фізичним розвитком тіла.

Цільова аудиторія продукту – усі бажаючі займатися спортом будь-якого виду, від початківців без фізичної підготовки – до професійних спортсменів.

Головними функціями програми є:

- вибір тренування на власний смак;
- роз'яснення тренувань;
- слідкування за прогресом та характеристиками власного тіла;
- поради тренерів та користувачів;

2.2 Типи користувачів в системі

У інформаційній системі, що розробляється, існує три типи користувачів:

- адміністратор – управління користувачами та підтримка роботи сервера;
- спортсмен – користувач, який використовує програму для власних цілей, а саме тренується, та слідкує за своїм прогресом;
- тренер – користувач, який створює свої програми тренувань для спортсменів, надає їм поради.

Для більш чіткого і повного розуміння функціоналу користувачів інформаційної системи, була розроблена діаграма прецедентів, в якій відображається основний функціонал розроблюваної інформаційної системи (рис. 2.2).

Розроблена діаграма дозволяє більш детально описати функціональні вимоги, які пред'являються до інформаційної системи, що розробляється.



Рисунок 2.2.1 – Діаграма прецедентів інформаційної системи (частина 1)



Рисунок 2.2.2 – Діаграма прецедентів інформаційної системи (частина 2)

2.3 Визначення функціональних вимог до інформаційної системи

До інформаційної системи пред'являються наступні функціональні вимоги.

FR1 Реєстрація користувача.

FR1.1 Незареєстрований користувач натискає кнопку «Реєстрація», після чого вводить бажаний логін (не менше 8 символів), пароль (не менше 8 символів) і заповнює поле підтвердження паролю, яке повинно співпадати з введеним раніше паролем, після чого натискає кнопку «Зареєструватися»;

FR1.2 У разі успішної реєстрації користувач потрапляє на вітальну сторінку додатка;

FR1.3 Після потрапляння на вітальну сторінку користувач може продовжити використання додатку натиснувши на кнопку «Продовжити»;

FR1.4 У разі невдачі реєстрації (логін зайнятий), додаток просить користувача змінити логін, так як такий логін вже зайнятий. Поля логіна і пароля в це випадку очищаються;

FR2 Авторизація користувача.

FR2.1 Зареєстрований користувач (спортсмен або тренер) вводить логін і пароль в форму введення і натискає кнопку «Увійти»;

FR2.2 Якщо користувач ввів логін і \ або пароль некоректно, система видає помилку і просить ввести логін і пароль повторно. Поля введення логіна і пароля очищаються;

FR2.4 У разі вдалої авторизації, користувач потрапляє на вітальну сторінку додатка;

FR2.4 Після потрапляння на вітальну сторінку користувач може продовжити використання додатку натиснувши на кнопку «Продовжити», після чого потрапляє на головну сторінку додатка;

F3 Заповнення власних параметрів спортсмена.

FR3.1 Користувач може натиснути на кнопку «Статистика» і перейти на цю сторінку;

FR3.2 Користувачу надається форма для заповнення необхідною інформацією, щодо своєї фізичної складової;

FR3.3 Після заповнення форми, користувач натискає кнопку «Додати характеристику», і вона буде відображатися у відділі статистики;

FR3.7 Користувач може придбати преміум підписку на додаток у розділі головної сторінки;

FR4 Вибір тренування користувачем

FR4.1 Користувач може вибрати тренування, по своїм критеріям та можливостям, передивляючи список тренувань на вкладці «Тренування»;

FR4.2 Користувач натискає на кнопку «Детальніше» на тренуванні, яке він вибрав і йому відкривається повний його опис;

FR4.3 Користувач може закріпити вибране тренування на вершині списку, задля подальшого легкого користування;

FR5 Коментування тренувань користувачами

FR5.1 Після пропрацювання тренування користувач може залишити відгук, щодо нього натиснувши на кнопку «Додати коментар», після чого його дається поле для вводу для нього;

FR6 Редагування тренувань

FR6.1 Користувач тренер може перейти у вкладку «Тренування» і вибрати необхідне для редагування тренування;

FR6.2 Тренер натискає на кнопку «Редагувати тренування», і йому видається форма заповнена даними старого тренування;

FR6.3 Тренер редагує форму, і натискає на кнопку «Зберегти» для збереження редагування;

FR7 Адміністрування системи.

FR7.1 Адміністратору надаються усі повноваження даного додатку задля можливостей наглядання та коригування додатку;

FR8 Управління користувачами.

FR8.1 Адміністратор може створювати нових користувачів;

FR8.2 Адміністратор може створюват нових адміністраторів;

FR8.3 Адміністратор може продивлятися профілі користувачів;

FR8.4 Адміністратор може переглядати коментарі користувачів, залишені під тренуваннями;

FR8.5 Адміністратор може видалити коментар користувача. Для цього він повинен натиснути на кнопку «Видалити» близько коментаря;

FR8.6 Адміністратор може заблокувати користувача. Для цього він повинен натиснути кнопку «Заблокувати» навпроти певного користувача, після чого система попросить його підтвердити виконувану дію.

FR8.7 Заблокований користувач не зможе зайти в додаток під своїм обліковим записом;

2.4 Проєктування структури додатку

Проєктований веб-додаток містить занадто багато класів для того, щоб приводити повну діаграму класів. На рисунку 2.4 показана концептуальна діаграма класів, на якій вказані основні класи додатку, які важливі для розуміння структури додатка в цілому, також в діаграмі зазначені основні операції в класах.

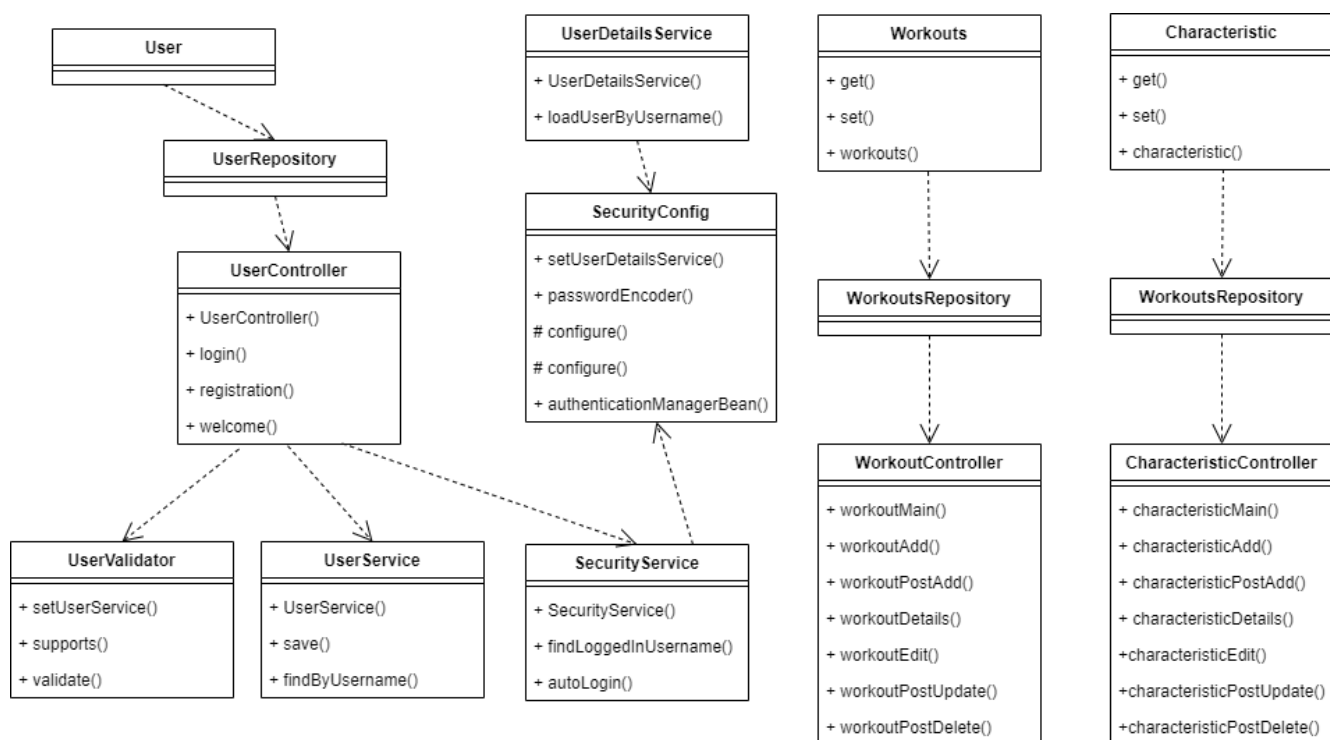


Рисунок 2.4 Діаграма класів

2.5 Проектування сценаріїв

Для розуміння поведінки системи в динаміці необхідно змодельовати процеси, які відбуваються в системі в процесі реалізації сценаріїв роботи програми.

Розглянемо сценарій реєстрації користувача у додатку. В рамках даного сценарію виконуються наступна послідовність дій:

- користувач вводить ім'я та пароль;
- введені дані приймаються і йдуть на перевірку;
- йде перевірка на правильність заповнення полів імені паролю та підтвердження паролю;
- йде перевірка на унікальність користувача у системі;
- обирається роль користувача;
- дозволяється доступ до системи користувачу;
- користувач додається до бази даних та відображається у привітальному повідомленні додатку.

На рисунку 2.5.1 представлена діаграма послідовності, яка дозволить більш детально описати взаємодію між об'єктами інформаційної системи.

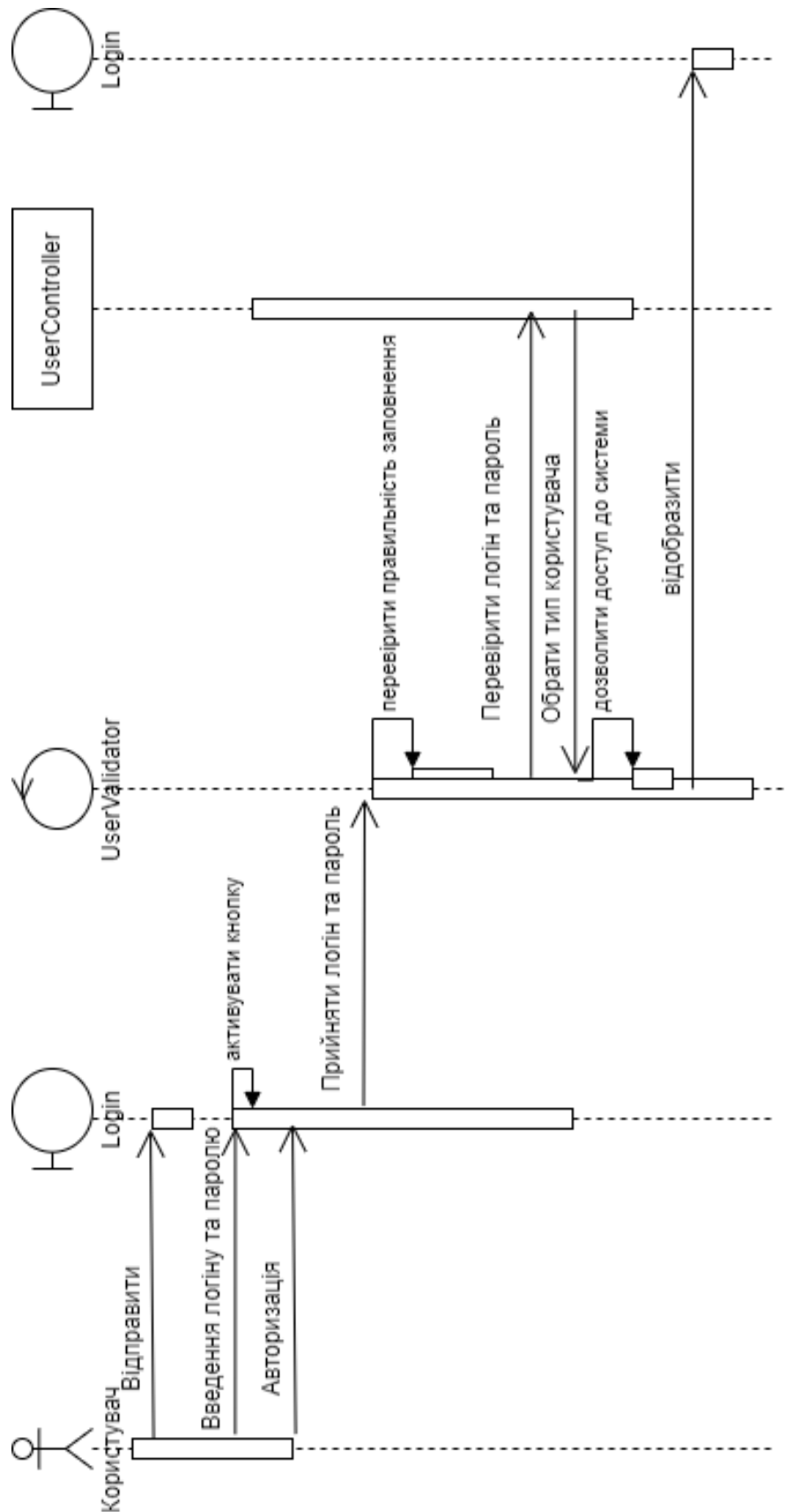


Рисунок 2.5.1 – Діаграма послідовностей сценарію реєстрації користувач

3 РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Структура програмного проєкту

Формування структури програмного проєкту інформаційної системи вироблено згідно з ранішим проєктуванням. Інформаційна система складається з клієнтської і серверної частини. Разом вони являють собою веб-додаток, який базується на фреймворку Java Spring.

Структура проєкту зображена на рисунку 3.1.

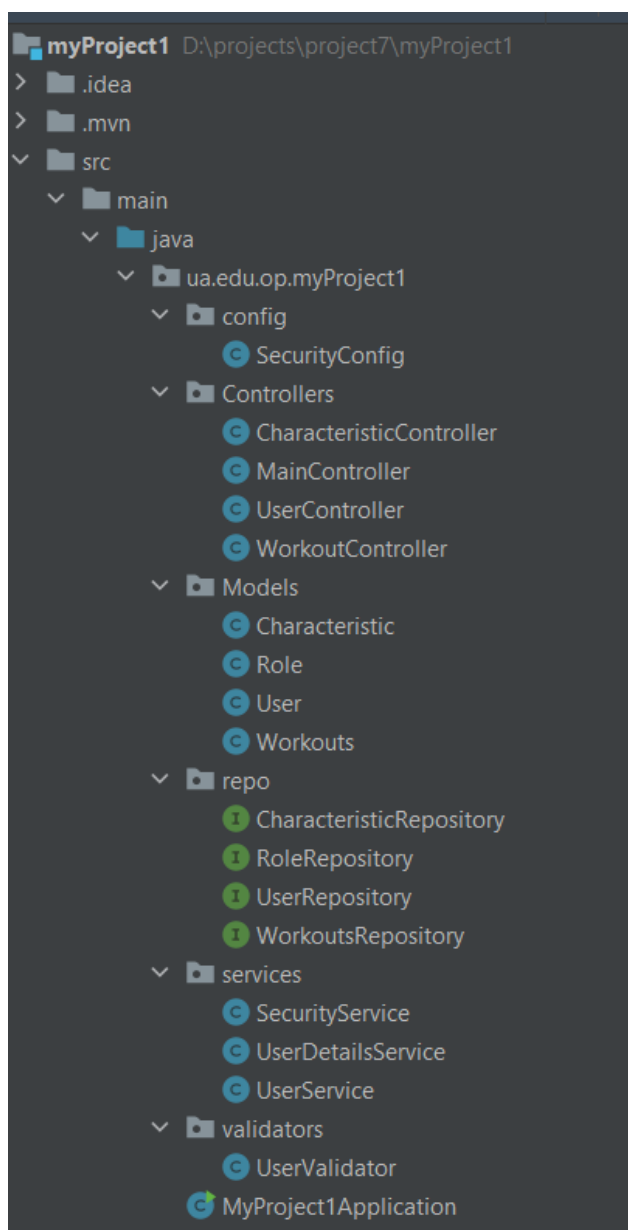


Рисунок 3.1 – Структура проєкту додатка

Особливістю побудови веб-додатка за допомогою Java Spring є наявність схожих класів, які мають схожі обов'язки, проте кожен клас несе відповідальність за свої сутності. Завдяки цьому проєкт раціонально ділити на пакети класів аніж на окремі класи.

Для веб-додатку було спроектовано наступні пакети:

- пакет `config` – даному пакет відповідає за налагодження безпеки, авторизацію та аутентифікацію;
- пакет `controller` – в даному пакеті знаходяться класи REST-контролерів, які обробляють вхідні запити і формують вміст HTTP-відповідей;
- пакет `model` – пакет в якому знаходяться класи моделей (сутностей) предметної області;
- пакет `repository` – в даному пакеті знаходяться інтерфейси-репозиторії, виходячи з яких Spring Data JPA генерує класи для взаємодії з базою даних;
- пакет `services` – пакети сервісів які, відповідають за створення користувачів та їх повноваження;
- пакет `validators` – відповідає за перевірку користувачів при реєстрації.

Також у проєкті задіяні 13 HTML-шаблонів різного призначення для успішної реалізації веб-складової. Також спроектований пакет ще для двох шаблонів, які містять в собі верхню та нижню частину усіх шаблонів відповідно. Вони потрібні для легшої роботи та полегшення роботи з майбутніми правками. Без шаблонів уявити даного роду додаток вкрай важко.

Структуру шаблонів показано на рисунку 3.2.

У проєкті також є файл `application.properties` в якому містяться налаштування підключення до вибраної СУБД `mysql`.

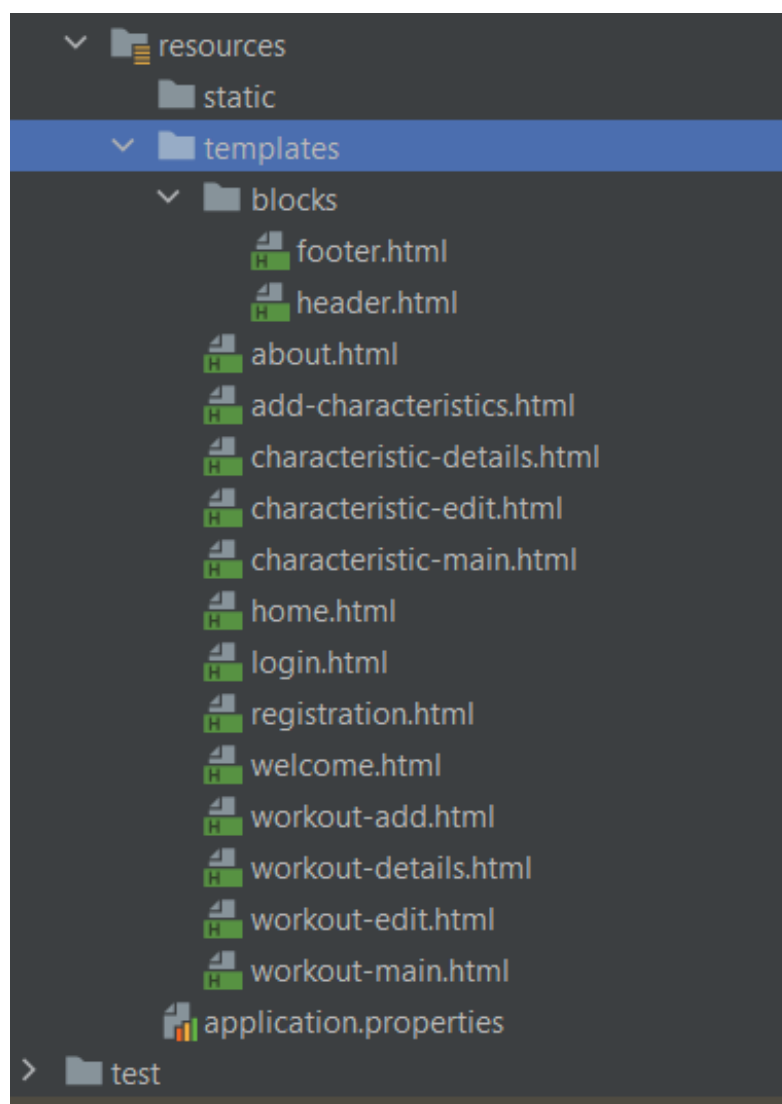


Рисунок 3.2 Шаблони задіяні у програмному проєкті

3.2 Реалізація основних алгоритмів

Виберемо декілька реалізацій алгоритмів для прикладу.

Додавання тренування. В результаті заповнення тренером форми додавання тренування буде натиснута кнопка для відправлення даних серверу. Після чого він його прийме і відправить дані в базу даних додатку.

Код прикладу:

```
@PostMapping("/workout/add")
```

```
public String workoutPostAdd(@RequestParam String name, @RequestParam
String type, @RequestParam String level,
```

```

        @RequestParam String description, @RequestParam String
trainer, Model model){
    Workouts workouts = new Workouts(name, type, level, description, trainer);
    workoutsRepository.save(workouts);
    return "redirect:/workout";
}

```

Тепер подивимося зворотній процес. Користувач спортсмен коли вибере собі необхідне йому тренування натисне кнопку «Детальніше», у цьому разі сервер вже запросить дані у бази даних, а вона в свою чергу надасть їх серверу, який і відобразить їх для користувача.

Код прикладу:

```

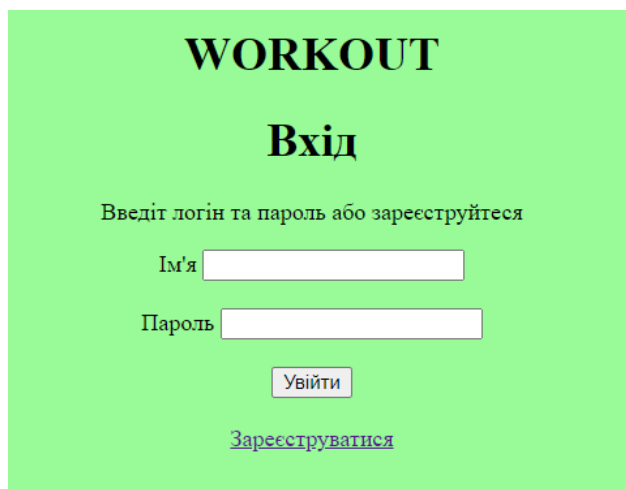
@GetMapping("/workout/{id}")
public String workoutDetails(@PathVariable(value = "id") long id, Model
model){
    if(!workoutsRepository.existsById(id)){
        return "redirect:/workout";
    }
    Optional<Workouts> workout = workoutsRepository.findById(id);
    ArrayList<Workouts> result = new ArrayList<>();
    workout.ifPresent(result::add);
    model.addAttribute("workout",result);
    return "workout-details";
}

```

З даних прикладів можна чітко побачити роботу трирівневої архітектури інформаційної системи, яку було обрано на початку роботи.

3.3 Розгортання додатку та інструкція користувача

Вхід у додаток (рис. 3.3.1) здійснюється з введення ім'я та пароля користувача. Якщо користувач ще не зареєстрований, потрібно натиснути кнопку «Зареєструватися» і ввести своє ім'я та пароль (рис. 3.3.2). Після чого натиснути кнопку «Створити акаунт». У результаті вдалої реєстрації програма привітається з користувачем. Натиснувши на посилання «продовжити» користувач заходить в додаток.



WORKOUT

Вхід

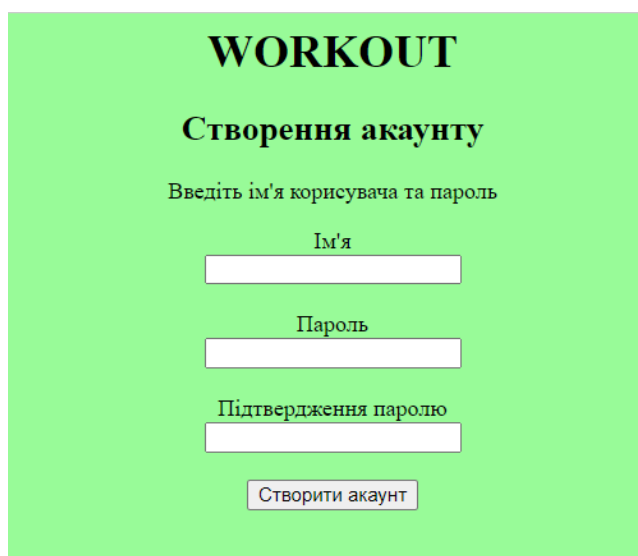
Введіть логін та пароль або зареєструйтеся

Ім'я

Пароль

[Зареєструватися](#)

Рисунок 3.3.1 Вхід у додаток



WORKOUT

Створення акаунту

Введіть ім'я користувача та пароль

Ім'я

Пароль

Підтвердження паролю

Рисунок 3.3.2 Реєстрація у додатку

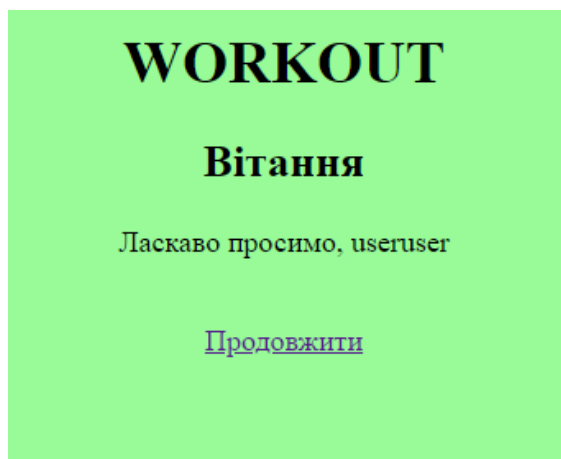
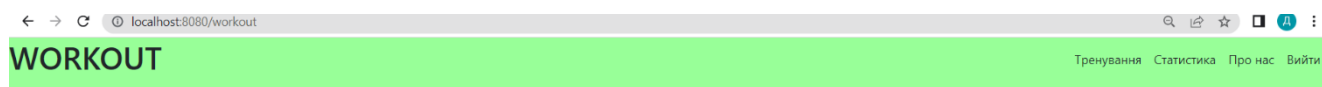


Рисунок 3.3.3 Вітання додатку

Після вдалого входу у додаток відкривається головна сторінка з тренуваннями. Для спортсменів видно лише список тренувань, які можна продивлятися натиснувши на кнопку «детальніше». Для тренерів ще доступна кнопка «+ додати тренування», натиснувши на яку відкриється форма для додавання тренування. Після заповнення форми потрібно натиснути кнопку «Додати тренування», після чого нове тренування буде успішно додане. Також тренерам ще доступна кнопка «Редагувати», після натискання якої виведеться та сама форма що і в додаванні, проте заповнена старим тренуванням для його редагування.



Список тренувань

+ Додати тренування

[Back to top](#)

Рисунок 3.3.4 Головна сторінка «Тренування»

WORKOUT Тренування Статистика Про нас Вийти

Додавання тренування

Введіть назву

Введіть тип

Введіть рівень складності

Опис

Тренер

Додати тренування

[Back to top](#)

Рисунок 3.3.5 Додавання тренування

Для усіх користувачів доступна кнопка статистика, яка знаходиться у шапці додатку. Після її натискання користувачу буде надана його статистика тренування, особисті характеристики. Для додавання особистих даних потрібно натиснути на кнопку «+ Додати свої характеристики». Відкриється форма для додавання, після чого потрібно ввести дані і натиснути на кнопку «Завершити».

WORKOUT Тренування Статистика Про нас Вийти

Статистика

+ Додати свої характеристики

[Back to top](#)

Рисунок 3.3.6 сторінка Статистика

WORKOUT

Тренування Статистика Про нас Вийти

Характеристика

Введіть стат

Введіть вік

Введіть зріст

Додаткові характеристики

Завершити

[Back to top](#)

Рисунок 3.3.6 Додавання індивідуальної характеристики користувача

У шапці додатку існує кнопка «Про нас» де розміщується можлива потрібна інформація для користувачів щодо додатку. Наприклад є можливість придбати преміум підписку додатку, яка надасть вільне коистування додатком без реклами.

Також у шапці додатку є кнопка «Вийти», після наптискання якої користувач виходить із системи та переходить на вкладку «Вхід», де буде повідомлення про успішний вихід з системи.

Майже на кожній сторінці є посилання «Back to top», яка дозволяє після довгого скролення до низу повертатися на верх сторінки.

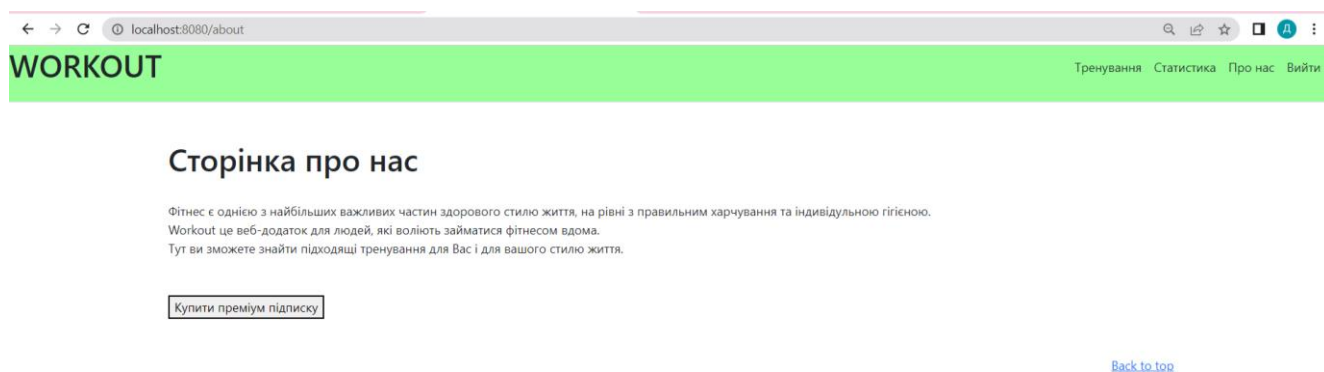
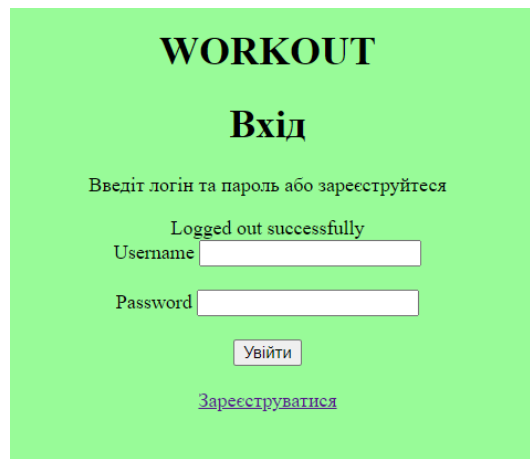


Рисунок 3.3.7 Сторінка «Про нас»



WORKOUT

Вхід

Введіть логін та пароль або зареєструйтеся

Logged out successfully

Username

Password

[Зареєструватися](#)

Рисунок 3.3.8 Вихід з системи

ВИСНОВКИ

У даній курсовій роботі була розроблена інформаційна система для фітнес тренувань, головною перевагою якої є можливість правильно тренуватися у домашніх умовах і безкоштовно отримувати інформацію про фітнес та своє тіло.

Таким чином, мета, поставлена перед даної кваліфікаційної роботою досягнута в повному обсязі.

Для досягнення мети були вирішені наступні задачі. У першому розділі було проаналізована предметна область тренувань у домашніх та вуличних умовах. Був проведений аналіз популярних мобільних додатків у сфері фітнесу. На підставі аналізу предметної області та аналізу додатків-аналогів були сформовані вимоги до основних функцій розроблюваної інформаційної системи. Також був проведений аналіз та були обрані технології для реалізації розроблюваної інформаційної системи.

У другому розділі було здійснено проектування вище згааної інформаційної системи. Були сформовані мета, завдання системи, цільова аудиторія. Були спроектовані групи користувачів, для яких була сформована діаграма прецедентів та отримані вимоги до системи. Були спроектовані макети інтерфейсу, було розроблено діаграму логічного представлення системи, а також діаграму розгортання. Були сформовані діаграми шарів системи.

У третьому розділі була реалізована раніше спроектована інформаційна система. Була наведена структура проєктів, а також розроблені концептуальні діаграми класів клієнтської і серверної частини. Були наведені описи ключових класів і ключових пакетів додатку. Було проведено аналіз розробленої інформаційної системи. Як документації була розроблена інструкція користувача.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Sport.ua [Електронний ресурс] – Режим доступу: <https://sport.ua/uk/zdorovyj-obraz-zhizni/500402-training-and-exercises-at-home>
2. Домашні тренування [Мобільний додаток]
3. Nike Training Club [Мобільний додаток]
4. Adidas Training [Мобільний додаток]
5. Spring Home [Електронний ресурс] – Режим доступу: <https://spring.io/>
6. Spring Boot [Електронний ресурс] – Режим доступу: <https://spring.io/projects/spring-boot>
7. Вікіпедія [Електронний ресурс] – Режим доступу: <https://uk.wikipedia.org/>
8. Java [Електронний ресурс] – Режим доступу: <https://www.java.com/ru/>

ДОДАТОК А ПРОГРАМНИЙ КОД

```

@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    private UserDetailsService userDetailsService;

    @Autowired
    public void setUserDetailsService(UserDetailsService userDetailsService) {
        this.userDetailsService = userDetailsService;
    }

    @Bean
    public PasswordEncoder passwordEncoder() {
        return NoOpPasswordEncoder.getInstance();
    }

    //Аутентификация
    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception {
        auth.userDetailsService(userDetailsService).passwordEncoder(passwordEncoder());
    }

    //Авторизация
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests()
            .antMatchers("/registration", "/login").permitAll()
            .antMatchers("/", "/welcome").hasRole("USER")
            .antMatchers("/workout", "/workout/", "/workout/add", "/workout/edit").hasRole("USER")
            .antMatchers("/characteristic", "/characteristic/", "/characteristic/add", "/characteristic/edit").hasRole("USER")
            .and()
            .formLogin()
            .loginPage("/login")
            .usernameParameter("username")
            .passwordParameter("password")
            .defaultSuccessUrl("/welcome", true)
            .failureUrl("/login?error")
            .and()
            .logout()
            .logoutUrl("/login?logout")
            .invalidateHttpSession(true)
            .deleteCookies("JSESSIONID");
    }
}

```

```

@Bean
@Override
public AuthenticationManager authenticationManagerBean() throws Exception{
    return super.authenticationManagerBean();
}

}

@Controller
public class CharacteristicController {

    @Autowired
    private CharacteristicRepository characteristicRepository;

    @GetMapping("/characteristic")
    public String characteristicMain(Model model){
        Iterable<Characteristic> characteristics = characteristicRepository.findAll();
        model.addAttribute("characteristic", characteristics);
        return "characteristic-main";
    }

    @GetMapping("/characteristic/add")
    public String characteristicAdd(Model model){
        return "add-characteristics";
    }

    @PostMapping("/characteristic/add")
    public String characteristicPostAdd(@RequestParam String gender, @RequestParam String age, @RequestParam String
growth,
                                     @RequestParam String weight, @RequestParam String additional, Model model){
        Characteristic characteristic = new Characteristic(gender, age, growth, weight, additional);
        characteristicRepository.save(characteristic);
        return "redirect:/characteristic";
    }

    @GetMapping("/characteristic/{id}")
    public String characteristicDetails(@PathVariable(value = "id") long id, Model model){
        if(!characteristicRepository.existsById(id)){
            return "redirect:/characteristic";
        }
        Optional<Characteristic> characteristic = characteristicRepository.findById(id);
        ArrayList<Characteristic> result = new ArrayList<>();

```

```

        characteristic.ifPresent(result::add);
        model.addAttribute("workout",result);
        return "characteristic-details";
    }

    @GetMapping("/characteristic/{id}/edit")
    public String characteristicEdit(@PathVariable(value = "id") long id, Model model){
        if(!characteristicRepository.existsById(id)){
            return "redirect:/characteristic";
        }
        Optional<Characteristic> characteristic = characteristicRepository.findById(id);
        ArrayList<Characteristic> result = new ArrayList<>();
        characteristic.ifPresent(result::add);
        model.addAttribute("characteristic",result);
        return "characteristic-edit";
    }

    @PostMapping("/characteristic/{id}/edit")
    public String characteristicPostUpdate(@PathVariable(value = "id") long id, @RequestParam String gender,
        @RequestParam String age,
        @RequestParam String growth, @RequestParam String weight, @RequestParam String additional,
        Model model){
        Characteristic characteristic = characteristicRepository.findById(id).orElseThrow();
        characteristic.setGender(gender);
        characteristic.setAge(age);
        characteristic.setGrowth(growth);
        characteristic.setWeight(weight);
        characteristic.setAdditional(additional);
        return "redirect:/characteristic";
    }

    @PostMapping("/characteristic/{id}/remove")
    public String characteristicPostDelete(@PathVariable(value = "id") long id, Model model){
        Characteristic characteristic = characteristicRepository.findById(id).orElseThrow();
        characteristicRepository.delete(characteristic);
        return "redirect:/characteristic";
    }
}

@Controller
public class MainController {

```

```

@GetMapping("/")
public String home(Model model) {
    model.addAttribute("title", "Главная страница");
    return "home";
}

@GetMapping("/about")
public String about(Model model) {
    model.addAttribute("title", "Сторінка про нас");
    return "about";
}
}

@Controller
public class UserController {
    private final UserService userService;
    private final UserValidator userValidator;
    private final SecurityService securityService;

    @RequestMapping(value = "/login", method = RequestMethod.GET)
    public String Login(Model model, String error, String logout){

        if(error != null && !error.isEmpty()){
            model.addAttribute("error", "username or password is incorrect");
        }
        if(logout != null){
            model.addAttribute("message", "Logged out successfully");
        }

        return "login";
    }

    @Autowired
    public UserController(UserService userService, UserValidator userValidator, SecurityService securityService) {
        this.userService = userService;
        this.userValidator = userValidator;
        this.securityService = securityService;
    }

    // Передача браузеру страницы с формой

```

```

@RequestMapping(value = {"/registration"}, method = RequestMethod.GET)
public String registration(Model model) {
    model.addAttribute("userForm", new User());

    return "registration";
}

// Обработка данных формы
@RequestMapping(value = "/registration", method = RequestMethod.POST)
public String registration(@ModelAttribute("userForm") User user, BindingResult result, Model model) {

    // Валидация с помощью валидатора
    userValidator.validate(user, result);

    // Если есть ошибки - показ формы с сообщениями об ошибках
    if (result.hasErrors()) {
        return "registration";
    }

    // Сохранение пользователя в базе
    userService.save(user);

    //аутентификация
    securityService.autoLogin(user.getUsername(), user.getConfirmPassword());

    // Перенаправление на приветственную страницу
    return "redirect:/welcome";
}

@RequestMapping(value = {"/", "/welcome"}, method = RequestMethod.GET)
public String welcome(Model model){
    Authentication authentication = SecurityContextHolder.getContext().getAuthentication();
    String name = authentication.getName();

    model.addAttribute("username", name);
    return "welcome";
}

@Controller
public class WorkoutController {

```

```

@Autowired
private WorkoutsRepository workoutsRepository;

@GetMapping("/workout")
public String workoutMain(Model model){
    Iterable<Workouts> workouts = workoutsRepository.findAll();
    model.addAttribute("workouts",workouts);
    return "workout-main";
}

@GetMapping("/workout/add")
public String workoutAdd(Model model){
    return "workout-add";
}

@PostMapping("/workout/add")
public String workoutPostAdd(@RequestParam String name, @RequestParam String type, @RequestParam String level,
                             @RequestParam String description, @RequestParam String trainer, Model model){
    Workouts workouts = new Workouts(name, type, level, description, trainer);
    workoutsRepository.save(workouts);
    return "redirect:/workout";
}

@GetMapping("/workout/{id}")
public String workoutDetails(@PathVariable(value = "id") long id, Model model){
    if(!workoutsRepository.existsById(id)){
        return "redirect:/workout";
    }
    Optional<Workouts> workout = workoutsRepository.findById(id);
    ArrayList<Workouts> result = new ArrayList<>();
    workout.ifPresent(result::add);
    model.addAttribute("workout",result);
    return "workout-details";
}

@GetMapping("/workout/{id}/edit")
public String workoutEdit(@PathVariable(value = "id") long id, Model model){
    if(!workoutsRepository.existsById(id)){
        return "redirect:/workout";
    }
    Optional<Workouts> workout = workoutsRepository.findById(id);
    ArrayList<Workouts> result = new ArrayList<>();

```



```

        workout.ifPresent(result::add);
        model.addAttribute("workout",result);
        return "workout-edit";
    }

```

```

@PostMapping("/workout/{id}/edit")
public String workoutPostUpdate(@PathVariable(value = "id") long id, @RequestParam String name, @RequestParam
String type,
                                @RequestParam String level, @RequestParam String description, @RequestParam String trainer,
                                Model model){
    Workouts workouts = workoutsRepository.findById(id).orElseThrow();
    workouts.setName(name);
    workouts.setType(type);
    workouts.setLevel(level);
    workouts.setDescription(description);
    workouts.setTrainer(trainer);
    workoutsRepository.save(workouts);
    return "redirect:/workout";
}

```

```

@PostMapping("/workout/{id}/remove")
public String workoutPostDelete(@PathVariable(value = "id") long id, Model model){
    Workouts workouts = workoutsRepository.findById(id).orElseThrow();
    workoutsRepository.delete(workouts);
    return "redirect:/workout";
}
}

```

@Entity

@Data

```
public class Characteristic {
```

```
    @Id
```

```
    @GeneratedValue(strategy = GenerationType.AUTO)
```

```
    private Long id;
```

```
    private String gender, age, growth, weight, additional;
```

```
    public Characteristic(){ }
```

```
    public Characteristic(String gender, String age, String growth, String weight, String additional) {
        this.gender = gender;
    }

```

```
this.age = age;
this.growth = growth;
this.weight = weight;
this.additional = additional;
}

public Long getId() {
    return id;
}

public void setId(Long id) {
    this.id = id;
}

public String getGender() {
    return gender;
}

public void setGender(String gender) {
    this.gender = gender;
}

public String getAge() {
    return age;
}

public void setAge(String age) {
    this.age = age;
}

public String getGrowth() {
    return growth;
}

public void setGrowth(String growth) {
    this.growth = growth;
}

public String getWeight(){return weight;}

public void setWeight(String weight){this.weight = weight;}
```

```

public String getAdditional() {
    return additional;
}

public void setAdditional(String additional) {
    this.additional = additional;
}
}

@Entity
@Data
public class Role {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    @Column(name = "name")
    private String name;

    @ManyToMany(mappedBy = "roles")
    @JsonIgnore
    private Set<User> users;
}

@Entity
@Table(name = "users")
@Data
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    @Column(name = "username")
    private String username;

    @Column(name = "password")
    private String password;

    @Transient
    private String confirmPassword;
}

```

```

@ManyToMany
@JoinTable(
    name = "user_roles",
    joinColumns = @JoinColumn(name= "user_id"),
    inverseJoinColumns = @JoinColumn(name = "role_id")

)

private Set<Role> roles;
}

@Entity
public class Workouts {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    private String name, type, level, trainer;

    private String description;

    private int views;

    public Long getId(){
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getName(){
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getType(){
        return type;
    }

```

```
}

public void setType(String type) {
    this.type = type;
}

public String getLevel(){
    return level;
}

public void setLevel(String level) {
    this.level = level;
}

public String getDescription(){
    return description;
}

public void setDescription(String description) {
    this.description = description;
}

public String getTrainer(){
    return trainer;
}

public void setTrainer(String trainer) {
    this.trainer = trainer;
}

public int getViews() {
    return views;
}

public void setViews(int views) {
    this.views = views;
}

public Workouts() {
}

public Workouts(String name, String type, String level, String description, String trainer){
```

```

        this.name = name;
        this.type = type;
        this.level = level;
        this.description = description;
        this.trainer = trainer;
    }

}

public interface CharacteristicRepository extends CrudRepository<Characteristic, Long> {
}

public interface RoleRepository extends JpaRepository<Role, Long> {
}

public interface UserRepository extends JpaRepository<User, Long> {
    User findByUsername(String username);
}

public interface WorkoutsRepository extends CrudRepository<Workouts, Long> {
}

@Service
public class SecurityService {

    private UserDetailsService userDetailsService;
    private AuthenticationManager manager;

    @Autowired
    public SecurityService(UserDetailsService userDetailsService, AuthenticationManager manager) {
        this.userDetailsService = userDetailsService;
        this.manager = manager;
    }

    public String findLoggedInUsername(){
        Object details = SecurityContextHolder.getContext().getAuthentication().getDetails();
        if(details instanceof UserDetails){
            UserDetails ud = (UserDetails) details;
            return ud.getUsername();
        }
        return null;
    }
}

```

```

public void autoLogin(String username, String password){
    // Мы получим объект стандартный User из UserDetailsService
    UserDetails userDetails = userDetailsService.loadUserByUsername(username);

    // Сгенерируем специальный токен аутентификации
    UsernamePasswordAuthenticationToken token =
        new UsernamePasswordAuthenticationToken(userDetails, password, userDetails.getAuthorities());

    // Обратимся к менеджеру аутентификации и аутентифицируем токена
    manager.authenticate(token);

    // Аутентифицируем пользователя с помощью токена
    if(token.isAuthenticated()){
        SecurityContextHolder.getContext().setAuthentication(token);
    }
}

@Service
public class UserDetailsService implements org.springframework.security.core.userdetails.UserDetailsService {

    private UserRepository userRepository;

    @Autowired
    public UserDetailsService(UserRepository userRepository){
        this.userRepository = userRepository;
    }

    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException{

        User user = userRepository.findByUsername(username);

        //Granted Authorities

        Set<GrantedAuthority> authorities = new HashSet<>();

        for(Role role : user.getRoles()){
            authorities.add(new SimpleGrantedAuthority(role.getName()));
        }
    }
}

```

```

        return new org.springframework.security.core.userdetails.User(user.getUsername(), user.getPassword(), authorities);
    }
}

@Service
public class UserService {

    private UserRepository userRepository;
    private PasswordEncoder passwordEncoder;
    private RoleRepository roleRepository;

    @Autowired
    public UserService(UserRepository userRepository, PasswordEncoder passwordEncoder, RoleRepository roleRepository)
    {
        this.userRepository = userRepository;
        this.passwordEncoder = passwordEncoder;
        this.roleRepository = roleRepository;
    }

    public void save(User user){
        user.setPassword(passwordEncoder.encode(user.getPassword()));

        Set<Role> roles = new HashSet<>();
        roles.add(roleRepository.findById(1L).get());
        user.setRoles(roles);

        userRepository.save(user);
    }

    public User findByUsername(String username){
        return userRepository.findByUsername(username);
    }
}

@Service
public class UserValidator implements Validator {

    private UserService userService;

    @Autowired
    public void setUserService(UserService userService) {

```



```

        this.userService = userService;
    }

    @Override
    public boolean supports(Class<?> clazz) {
        return User.class.equals(clazz);
    }

    @Override
    public void validate(Object target, Errors errors) {
        User user = (User) target;

        // Поле username не должно быть пустым
        ValidationUtils.rejectIfEmptyOrWhitespace(errors, "username", "", "This field is required.");
        // Поле username должно быть длиной от 8 до 32 символов
        if (user.getUsername().length() < 8 || user.getUsername().length() > 32) {
            errors.rejectValue("username", "", "Username must be between 8 and 32 characters");
        }
        // Поле username должно быть уникальным в системе
        if (userService.findByUsername(user.getUsername()) != null) {
            errors.rejectValue("username", "", "Username is already exists.");
        }

        // Поле password не должно быть пустым
        ValidationUtils.rejectIfEmptyOrWhitespace(errors, "password", "", "This field is required.");
        // Поле password должно быть длиной от 8 до 32 символов
        if (user.getPassword().length() < 8 || user.getPassword().length() > 32) {
            errors.rejectValue("password", "", "password must be between 8 and 32 characters");
        }
        // Поле password должно совпадать с полем confirmPassword
        if (!user.getConfirmPassword().equals(user.getPassword())) {
            errors.rejectValue("password", "", "Passwords don't match!");
        }
    }
}

```