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

ВИПУСКНА РОБОТА

на тему:

«Інформаційна система обліку користувачів басейну»

Завідувач
випускаючої кафедри **Довбиш А.С.**

Керівник роботи **Олексієнко Г.А.**

Студент групи ІН-63 **Коренев М.І.**

СУМИ 2020

РЕФЕРАТ

Записка: 61 стор., 15 рис., 2 табл., 1 додаток, 13 джерел.

Об’єкт дослідження — Інформаційна система обліку користувачів басейну.

Мета роботи — проектування та розробка інформаційної системи обліку користувачів басейну.

Методи дослідження — системно-інформаційний аналіз, інформаційне моделювання та комп'ютерний експеримент.

Результати — проведений аналіз літератури, розглянуто саме поняття інформаційної системи, класифікації та переваг використання. Проведено аналіз існуючих інформаційних систем та підходів до їх реалізації. Після ознайомлення та аналізу існуючих рішень був розроблений власний алгоритм вирішення поставленої задачі. Серед існуючого різноманіття різних інструментів та підходів обрано та обґрунтовано вибір СУБД, мов програмування, фреймворку та середовища розробки. Спроектований база даних та реалізований додаток у вигляді вебсайту, що відповідає поставленому завданню та дозволяє зручно взаємодіяти користувачам з інформаційною системою обліку користувачів басейну.

ІНФОРМАЦІЙНІ СИСТЕМИ, SPRING FRAMEWORK, JAVA, ORACLE, JAVASCRIPT, THYMELEAF, BOOTSTRAP.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Затверджую_____

Зав. кафедрою Довбиш А.С.

“_____” _____2020 р.

ЗАВДАННЯ

до випускної роботи

Студента четвертого курсу, групи ІН-63 спеціальності “Інформатика” денної форми навчання Коренева Миколи Івановича.

Тема: “**Інформаційна система обліку користувачів басейну**”

Затверджена наказом по СумДУ

№_____ от _____2020 р.

Зміст пояснювальної записки: 1) аналітичний огляд літератури та існуючих рішень; 2) постановка завдання дослідження; 3) огляд технологій та інструментарію для реалізації додатку; 4) проектування інформаційної системи обліку користувачів басейну; 5) розробка додатку; 6) аналіз результатів.

Дата видачі завдання “_____”_____2020 р.

Керівник випускної роботи _____ Олексієнко Г.А.

Завдання прийняв до виконання _____ Коренев М.І.

ЗМІСТ

Вступ

Інформація, її отримання, оброблення та зберігання — невід'ємна частина сьогодення. Правильно організована робота з інформацією - важлива частина функціонування будь-якої організації. Постійне збільшення інформаційного навантаження кидає нові виклики, які потребують сучасних рішень, підвищуються вимоги до швидкості отримання, якості обробки та зберігання інформації. Також з`являються та вдосконалюються вимоги законодавства до організації роботи з інформацією, наприклад, при роботі з персональними даними потрібно дотримуватися вимог передбаченим Законом України «Про захист персональних даних» [1].

Важливо зазначити, більшість операцій з інформацією не може бути ефективно виконана вручну, а потребує сучасних комп'ютерних та автоматизованих рішень. Для забезпечення ефективної роботи з інформацією потрібна відповідна структурована організація зберігання даних.

Сучасний підхід до вирішення питання структурованого зберігання даних напряму пов`язаний з базами даних. Такий підхід сучасний і достатньо використовний в тих сферах, де актуальні проблеми впорядкованого зберігання даних та швидкого доступу до них.

Одним із прикладів організації роботи з інформацією, її отриманням, впорядкуванням, зберіганням та використанням в автоматизованих та комп'ютерних рішеннях, може послужити дана робота.

Необхідно створити просту та ефективну інформаційну систему для басейну, яка зможе забезпечити повний інформаційний супровід при роботі з клієнтами. Забезпечить клієнтів всією необхідною інформацією, а адміністратору надасть інструменти для зручного керуванням цією системою.

1 Аналіз предметної області

1.1 Інформаційні системи

Інформаційна система – система, призначена для пошуку, обробки та зберігання інформації з використанням певних ресурсів, котрі забезпечують функціонування такої системи.

Створення перших інформаційних систем, у сучасному розумінні, почало відбуватися в середині минулого сторіччя. Стрімкий розвиток комп'ютерної техніки безпосередньо впливає і на розвиток інформаційних технологій. Можна відстежити значний успіх у розвитку інформаційних систем, від вирішення задач спрощення процедур математичних розрахунків з використанням електромеханічних машин, в 1950-х роках, до сучасних автоматизованих рішень, навіть без залучення людини.

Тим не менше, за більш ніж пів вікову історію розвитку інформаційних систем, до сьогодні немає чіткого правила класифікації. А. М. БЕРЕЗА у своєму навчальному посібнику “Основи створення інформаційних систем” [2] наводить таку класифікацію інформаційних систем:

“1. За рівнем або сферою діяльності — державні, територіальні (регіональні), галузеві, об'єднань, підприємств або установ, технологічних процесів.”;

“2. За рівнем автоматизації процесів управління — інформаційно-пошукові, інформаційно-довідкові, інформаційно-керівні, системи підтримки прийняття рішень, інтелектуальні ІС.”;

“3. За ступенем централізації обробки інформації — централізовані ІС, децентралізовані ІС, інформаційні системи колективного використання. “;

“4. За ступенем інтеграції функцій — багаторівневі ІС з інтеграцією за рівнями управління (підприємство — об'єднання, об'єднання — галузь і т.ін.), багаторівневі ІС з інтеграцією за рівнями планування і т.ін.”;

“5. За типом ІС розподіляються на фактографічні, документальні ідокументально-фактографічні ІС.”.

Інформаційну систему можна розглянути як сукупність необхідних підсистем для її функціонування. Задача підсистем, показаних на рисунку 1.2, - забезпечення функціонування основної інформаційної системи.

Рисунок 1.1 - Інформаційна система та забезпечуючі її підсистеми

В інформаційній системі можна виділити такі основні процеси:

- отримання інформації з різних джерел
- опрацювання отриманої інформації
- збереження інформації
- вивід інформації для користувача

Рисунок 2.2 - Загальне зображення основних процесів в інформаційній системі [13]

Важливими принципами проектування і побудови ефективної інформаційної системи є принципи інтеграції, системності та комплексності. Відповідно до принципу інтеграції – дані отримані одноразово будуть використані багаторазово, для вирішення більшого числа поставлених задач. Суть принципу системності заключається в використанні даних в різних аспектах та модулях, щоб отримати необхідну інформацію для прийняття рішення, на всіх рівнях ІС. Згідно з принципом комплексності, дані можуть бути оброблені автоматизовано і перетворені відповідно до вимог системи, на всіх рівнях ІС системи.

Використання інформаційної системи може сприяти:
автоматизації рутинних операцій

- забезпеченню достовірності інформації
- заміни паперових носіїв інформації
- покращенню документообігу
- зменшенню часу на пошук необхідної інформації

отриманню більш раціональних варіантів вирішення проблем, за рахунок використання математичних, інтелектуальних та інших методів

- підвищенню якості обслуговування клієнтів
- зменшенню витрат на виробництво товарів і послуг
- меншенню витрат на додаткові послуги

1.2 Акутальність проблеми

За даними Федерації плавання України, наведеними у статті УНІАН: “До 2023 року в Україні спрямують 1 мільярд доларів на будівництво нових водно-спортивних комплексів” [3], в Україні нараховується 534 працюючі басейни та плавальні комплекси (100 – приватної форми власності).

Зокрема Федерація плавання у своєму прогнозі наводить дві тенденції розвитку водноспортивних комплексів. Створення і розвиток водно-спортивних об'єктів у великих та середніх містах (більше ніж 250 тисяч жителів) – інвестиційно приваблива сфера для фітнес-операторів, орієнтованих на побудову бізнес-моделі для отримання максимального прибутку і повернення інвестицій. Водночас, водноспортивні комплекси в середніх та малих містах найчастіше не є інвестиційно привабливими для бізнесу. Економічна модель таких закладів орієнтована для забезпечення окупності їх поточного утримання. Найчастіше такі об'єкти знаходяться в комунальній та державній власності та належать Федерації плавання, закладам освіти та територіальним громадам. Обмежений бюджет таких закладів не може витримати значні фінансові навантаження, а отже доводиться оптимізувати витрати.

Розглядаючи інформаційні системи, як засіб покращення роботи, важливо розуміти загальні тенденції на ринку, а не лише в конкретній сфері.

За даними дослідження проведеного популярним інтернет-журналом у сфері айти та технологій – “ain.ua” [4], серед українських компаній, у 61% з

них Excel – головний інструмент для ведення клієнтської бази, а 3% все ще використовують паперову звітність. Також 86% опитаних компаній не збираються щось змінювати у своїй діяльності в найближчі рік – два.

Аналізуючи дані дослідження проведеного у 2018 році, можна говорити, що загалом серед українських компаній достатньо низький рівень автоматизації. Головними причинами цього наводиться слабке розповсюдження в малому і середньому бізнесі знань про технології автоматизації процесів, небажання “ризикувати” бізнесом і змінювати щось у відлагодженому процесі та не можливість нести додаткові фінансові витрати.

Подібна ситуація спостерігається і в закладах водноспортивного комплексу, особливо в невеликих та середніх містах. Важливо розуміти, що “рішення вчорашнього дня” не завжди будуть працювати в майбутньому, особливо зважаючи на стрімкий розвиток помп’ютерних та автоматизовних систем, а отже їм доведеться щось змінювати у своїй діяльності.

Розглядаючи зокрема проблему ведення клієнтської бази, як частина інформаційної системи, важливо розуміти, що Excel хоча і популярний інструмент, але він не дає багатьох сучасних можливостей, не говорячи уже про паперову звітність. Можна говорити, що такі підходи застарівають і навряд чи зможуть вистояти проти сучасних інформаційних систем у майбутньому. Хоча, напевно однією з неперевершених преваг використання Excel, в порівнянні з різничими інформаційними систмеми, є те що він надає можливість людям, котрі далекі від програмування і математики, робити перерахунок формул “в польоті” при зміні вхідних даних. Але це й несе велику небезпеку. Проблеми безпеки, коректності вводу даних, коректності розрахунків, доступу багатьох користувачів до одного ресурсу та багато іншого повинні змушувати організації шукати більш безпечні та сучасні рішення для задоволення їх потреб.

Розглядаючи підходи і проблеми до ведення клієнтської бази та взагалі організації роботи з даними, можна простежити проблему в не використанні сучасних рішень, а тому далі необхідно розглянути існуючі рішення, врахувати тенденції та особливості наведені в даному пункті, проаналізувати недоліки існуючих рішень та сформувати вимоги до розроблюваної системи.

1.3 Огляд існуючих рішень

Для огляду подібних рішень візьмемо до уваги лише системи заточені на облік користувачів та інформаційний супровід басейну. Відносно CRM систем (системи управління відносинами з клієнтами) доцільно брати лише ті, що орієнтовані на дану тематику, адже загальне призначення CRM – це підвищення рівня продаж, оптимізація маркетингу та аналітики, що не зовсім вкладається в логіку інформаційної системи обліку користувачів басейну та спортивного закладу загалом.

Після аналізу для порівняння обрано наступні системи:
1) Абонемент – універсальна система для керування об’єктами: фітнес-клуб, салон краси, аквапарк, каток, басейн. Значно спрощує платіжно-пропускну систему. Має інтеграцію з такими системами як Shelter, R-keeper та Store House.
2) LUCKYFIT – проста і надійна система для автоматизації фітнес-клубів, студій, салонів краси, басейнів. Має власний мобільний додаток для клієнта.
3) 1С:Підприємство 8. Фітнес клуб – продукт для автоматизації процесів у фітнес-клубах, басейнах, спортивних комплексах та оздоровних установах.

Для порівняння існуючих рішень будуть використовуватись наступні групові критерії:

- Зручність – можливість працювати з системою з різних пристроїв
- Безпека – можливість розміщувати систему на власному сервері

- Функціональність – можливість системи задовольнити основні потреби в відносинахз клієнтами
- Ліцензія та обмеження – вартість системи для 1-5 користувачів та з кількістю активних клієнтів до 2 000 осіб

Таблиця 1.1

Порівняльна характеристика існуючих систем

	Абонемент	LUCKYFIT	1С:Підприємство 8. Фітнес клуб
Вебсайт	+	+	+
Модулі для наявного вебсайту	-	+	+
Мобільний додаток	-	+	-
Можливість установки системи на власний сервер	+	+	+
Перегляд розкладу занять	+	+	+
Можливість створювати розклад занять	+	+	+
Покупка абонемента через інтернет	+	+	+
Перегляд історії абонементів	+	+	+
Реєстрація відвідувань	+	+	+
Система повідомлень для клієнтів	+	+	+
Можливість додавати та переглядати новини басейну	+	-	-
Можливість залишати відгуки	-	-	-
Кількість клієнтів з дійсним абонементом	-	До 2 000	-

Попри існуючі недоліки, приведені системи дозволяють певною мірою виконувати завдання в рамках Інформаційної системи обліку користувачів басейну. Розглядаючи комплексі рішення потрібно розуміти, що частина наявних функцій програми не зможе повноцінно працювати в конкретній

сфері та частина функціоналу буде “зайвою”, що ускладнює керування такою системою. Також слід звернути увагу на додаткові вимоги при роботі з наведеними системами, адже для повноцінної їх роботи потрібна відповідна матеріально-технічна база (турнікети, електронні картки, електронні замки та інше), що приводить до підвищення разових матеріальних витрат та взагалі відбиватися на ціні впровадження таких систем.

Створюючи власну систему потрібно врахувати всі недоліки та особливості наведені у порівняльній характеристиці існуючих рішень.

2 Вибір методів реалізації

2.1 Визначення характеристик розроблюваної системи

Система повинна бути доступна для клієнта та адміністратора, тому найкращим варіантом такої реалізації буде web-додаток.

Для проектування та реалізації системи необхідно сформувати чіткий мінімальний набір вимог до системи.

Система повинна надавати клієнтові наступні можливості:

- Переглядати розклад занять
- Переглядати інформацію про діяльність басейну
- Зареєструватися в системі
- Переглядати та обирати абонементи
- Переглядати історію візитів
- Переглядати історію абонементів
- Залишати відгук

Система повинна надавати адміністратору наступні можливості:

- Додавати та змінювати публічну інформацію про діяльність басейну
- Створювати та змінювати розклад занять
- Створювати та змінювати абонементи
- Створювати групи та додавати клієнтів до груп
- Назначати та змінювати тренера для групи
- Додавати нових користувачів до системи
- Змінювати інформацію про користувачів
- Підтверджувати візити клієнтів
- Додавати абонементи клієнтам
- Змінювати статус плавальних доріжок
- Переглядати історію змін

- Проводити нотифікацію учасників системи

2.2 Вибір СУБД

Система управління базою даних (СУБД) – це комплекс програм та мовних засобів, призначених розробки, підтримки та виконання пошуку в базі даних.

Основні функції СУБД:

- керування даними на дисках
- керування даними в оперативній памяті
- ведення журналу змін та відновлення стану бази даних після збоїв
- підтримка мов бази даних

DB-Engines Ranking – рейтинг баз даних за їх популярністю, створений і підтримується австрійською компанією “Solid IT”. Рейтинг розраховується комбіновано, враховується частота пошуків в Google Trends, згадки в пошукових системах, кількість вакансій та інші дані [5].

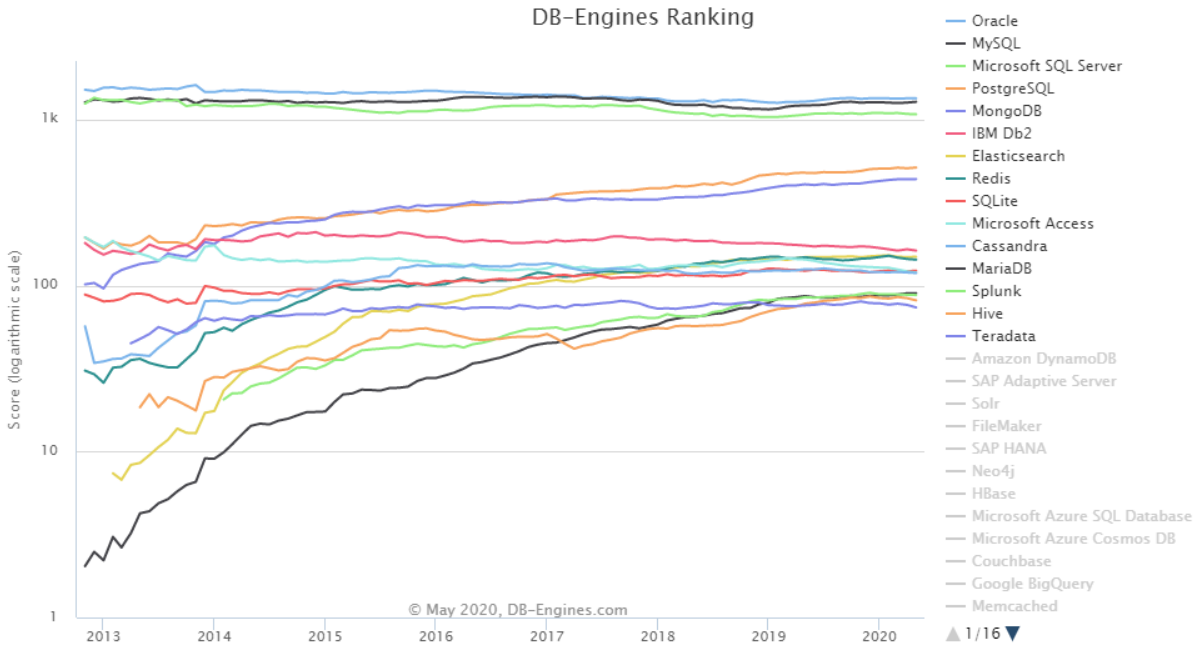

Рисунок 2.1 - DB-Engines Ranking

Oracle – найпопулярніша СУБД з достатньо широким набором різних можливостей, починаючи від підтримки MVCC – багатоверсійність даних для керування паралельними транзакціями й завершуючи можливістю внесення змін в метадані та бізнес-логіку, з використанням PL/SQL, без необхідності зупиняти сервер. Має підтримку великої кількості мов програмування: C, C#, C++, Clojure, Groovy, Haskell, Java, JavaScript, Erlang, Fortran, Scala, Tcl, Visual Basic, Lisp, Ruby, Cobol, Delphi, Objective C, OCaml, Perl, PHP, Python, R, Eiffel,

Oracle – комерційна система, що не заважає їй бути настільки популярною, оскільки має декілька редакцій, для задоволення потреб організацій різного рівня з різними фінансовими можливостями.

Oracle Database XE - безкоштовна версія бази, але з обмеженнями.

Oracle Database 18c Express Edition - одна з останніх безкоштовних версії, чудово підходить для академічного використання, невеликих компаній та стартапів, а також для оцінки потенціалу бази великими компаніями [6].

Характеристики та обмеження Oracle Database 18c Express Edition:

- Максимальний розмір бази — до 12 GB
- До 2 GB оперативної пам’яті
- До 2 потоків CPU
- До 3 підключених баз даних

2.3 Вибір мов програмування

Сервіс інформаційної системи – web-додаток. Головними частинами такого додатку буде backend та frontend. Потрібно обрати мови програмування для кожної із частин.

Обираючи мову програмування бекенда, потрібно звернути увагу на взаємодію обраної мови та базами даних. Список підтримуваних мов, наведено в попередньому розділі, для СУДБ Oracle. Також ця мова повинна бути сучасна, а також добре інтегрована з мовою фронтенда.

TIOBE Programming Community index – індекс популярності мов програмування. Індекс визначається з 2003 року швейцарською компанією “TIOBE” [7].

May 2020	May 2019	Change	Programming Language	Ratings	Change
1	2	^	C	17.07%	+2.82%
2	1	v	Java	16.28%	+0.28%
3	4	^	Python	9.12%	+1.29%
4	3	v	C++	6.13%	-1.97%
5	6	^	C#	4.29%	+0.30%
6	5	v	Visual Basic	4.18%	-1.01%
7	7		JavaScript	2.68%	-0.01%
8	9	^	PHP	2.49%	-0.00%
9	8	v	SQL	2.09%	-0.47%
10	21	^^	R	1.85%	+0.90%

Рисунок 2.2 - TIOBE Programming Community index

Java – об'єктно-орієнтована та строго типізована мова програмування, синтаксично подібна до C і відповідає підходу - “Написати один раз і використовувати скрізь”. Дійсно, програми написані мовою Java не мають прив’язки до певної операційної системи, а можуть бути виконані на різних операційних системах, що не можна сказати, наприклад, про C++. Для запуску програми написаною мовою Java потрібне певне середовище - Java Runtime Environment (JRE) та Java Virtual Machine (JWM). Саме підхід з користанням проміжного коду дозволяє зробити програми кросплатформеними.

Java використовується в веб-додатках, в розробці для Android, в додатках на десктоп та в технологіях обробки великих даних, наприклад Hadoop.

Не дивлячись на свій вік, а Java розпочала свій розвиток більш ніж 20 років тому, в 1995 році, залишається однією з найпопулярніших мов програмування у світі. TIOBE Index для Java показано на рисунку 2.3.

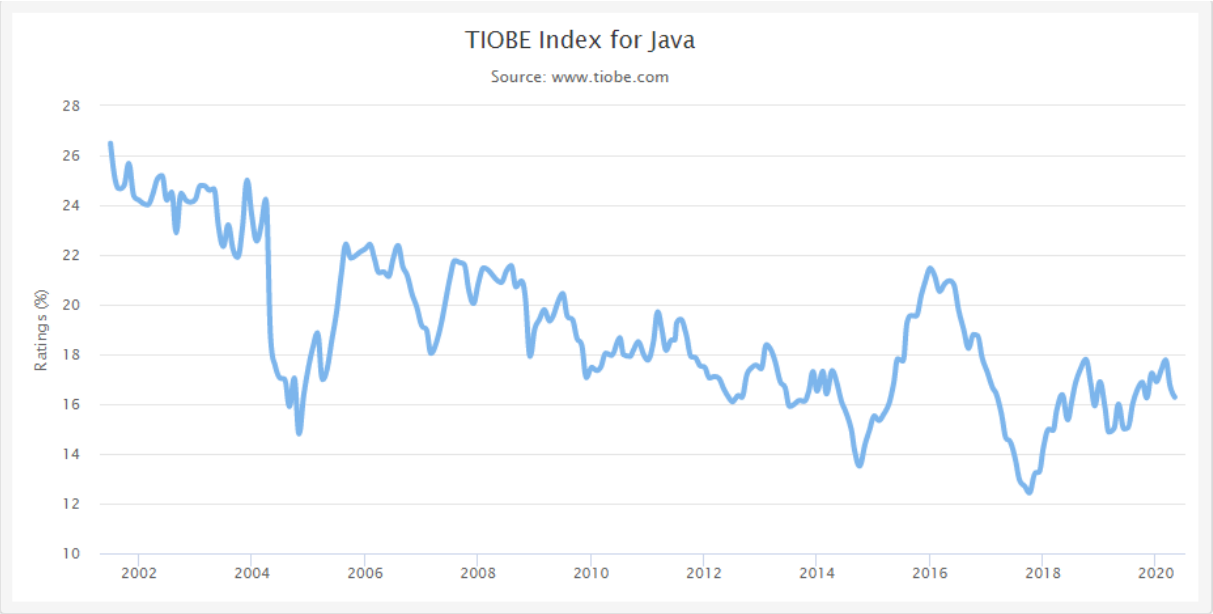

Рисунок 2.3 - TIOBE Index for Java

Обираючи мову програмування, важливо звернути увагу на головні переваги та недоліки.

До головних переваг використовувати Java можна віднести:

- Об'єктно-орієнтований підхід (ООП) – сучасний підхід до програмування з використанням об'єктів, що дає переваги в повторному використанні коду, допомагає краще організувати структуру програми та спрощує підтримку коду.
- Java – мова високого рівня, з великою кількістю абстракцій, котрі зрозумілі людині. Мова високого рівня набагато легша для сприйняття людиною, що значно спрощую написання кода та його подальший супровід. Відхід від машинних команд забезпечується за допомогою компілятора та інтерпретатора.
- Синтаксична подібність до C++, адже при розробці Java синтаксис C++ був взятий за основу. Порівнюючи синтаксис Java та C++ можна говорити проте, що синтаксично Java навіть легша ніж C++, має

достатньо велику кількість абстракцій та змогла врахувати існуючі недоліки своєї базової мови.

- Безпека та вразливості – дуже важливий фактор, але жодна з існуючих мов не здатна повністю захистити ваш код від різних вразливостей. В порівнянні з C, в Java відсутні показники, що значно підвищують безпеку коду. Наявність Security Manager також позитивно впливає на безпеку, также можливість вказати права доступу і виконувати запуск додатків в “пісочниці”, запобігаючи таким чином вразливостям.
- Автоматичне керування пам'яттю – саме так, не потрібно вручну видаляти об'єкти, робота з якими закінчилась, Java зробить це сама. Програма перевіряє існуючі об’єкти і видаляє ті, що більше не використовуються.
- Багатопотоковість дозволяє максимально ефективно використовувати доступні ресурси для стабільної та швидкої роботи додатків.
- Кросплатформенність Java – невід'ємна перевага над багатьма існуючими мовами.
- Велика кількість бібліотек та фреймворків дає достатньо сильну перевагу мові, а відповідно і існування широкої спільноти розробників. За даними дослідження StackOverflow 2019 [8] більше ніж 40% опитаних використовують Java.

При достатні великій кількості переваг, як і будь-яка мова, Java не позбавлена і недоліків, основні з них перераховані нижче:

- Належність мови певній компанії. Мова Java належить компанії Oracle та є платною для використання в комерційних цілях, відповідно до оновлення ліцензії від 16 квітня 2019 року [9].
- Швидкість роботи та низька продуктивність роботи програм в певних ситуаціях. Використання проміжного коду та JVM окрім очевидних

переваг має в собі й недоліки, зокрема витрати часу на виконання таких програм. Проблема з правильних налаштуваванням кешуванням та очистки памяті також вимагає певних людських ресурсів, що не є перевагою Java.

Враховуючи наведені переваги та недоліки, для реалізації бекенда було обрано Java. Наступним кроком буде вибір мови фронтенда і в якості такої мови було обрано JavaScript. Загальні інформація та основні переваги та недоліки мови наведено нижче.

JavaScript – сучасна мультипарадигмова мова програмування. Найбільшої популярності JavaScript здобула в браузерах, для надання інтерактивності вебсторінкам. Це високорівнева мова програмування з динамічною типізацією. JavaScript займає провідне місце за кількістю репозиторіїв на GitHub [10]. Рейтинг мови за кількістю репозиторіїв представлено на рисунку 2.4.

# Ranking	Programming Language	Percentage (Change)	Trend
1	JavaScript	18.703% (-1.406%)	
2	Python	16.238% (-1.654%)	
3	Java	10.938% (+0.538%)	
4	Go	9.005% (+0.978%)	
5	C++	7.423% (+0.040%)	
6	Ruby	6.812% (+0.342%)	
7	TypeScript	6.769% (+1.522%)	^
8	PHP	5.127% (-0.458%)	v
9	C#	3.835% (+0.141%)	
10	C	3.181% (-0.203%)	

Рисунок 2.4 - Рейтинг мов програмування за кількістю репозиторіїв GitHub

Переваги використання JavaScript:

- Широка сфера застосування, котра не обмежується лише вебсайтами

- Підтримка JavaScript практично всіма відомими браузерами
- Можливість виконувати код лише на стороні клієнта (наприклад валідація даних), що не потребує звернень до сервера і підвищує швидкодію запитів
- Пряме підключення скриптів написаних на JavaScript до HTML коду
- Велика кількість проектів та широка спільнота розробників

До головних недоліків можна віднести:

- Компіляція під час виконання. Компіляція відбувається кожен раз при відкриванні сайту на javascript, що підвищує час на виконання програми.
- Динамічна типізація накладає свій негативний відбиток на пошук помилок та час роботи з такими типами більше в порівнянні зі строгою типізацією.
- Не схожість з C++/C . Відмінність від де-факто стандартних рішень, що потребує часу для вивчення особливостей об'єктної моделі мови.
- Проблема безпеки пов'язана з вільним доступом до коду, що спонукає використовувати додаткові рішення для запобігання різним вразлвостям.

Для частини відомих недоліків давно уже знайдено рішення, наприклад для зменшення об'єму кода використовується алгоритм зжимання (наприклад gzip), для вирішення проблеми з динамічною типізацією існує рішення від google - closure compiler, а для наближення моделі JavaScript до C++/C використовують спеціальні інструменти – Mootools та AWeb library.

2.4 Вибір фрейморку для реалізації

Для реалізації web-додатку потрібно обрати певний веб-фрейморк. Фреймворк – програмне забезпечення створене для полегшення розробки. Найчастіше складається з двох частин: постійна частина (каркас) та змінні

модулі (точки розширення). Рейтинг найпопулярніших веб-фрейворків за версією сайту hotframeworks [11] наведено на рисунку 2.5.

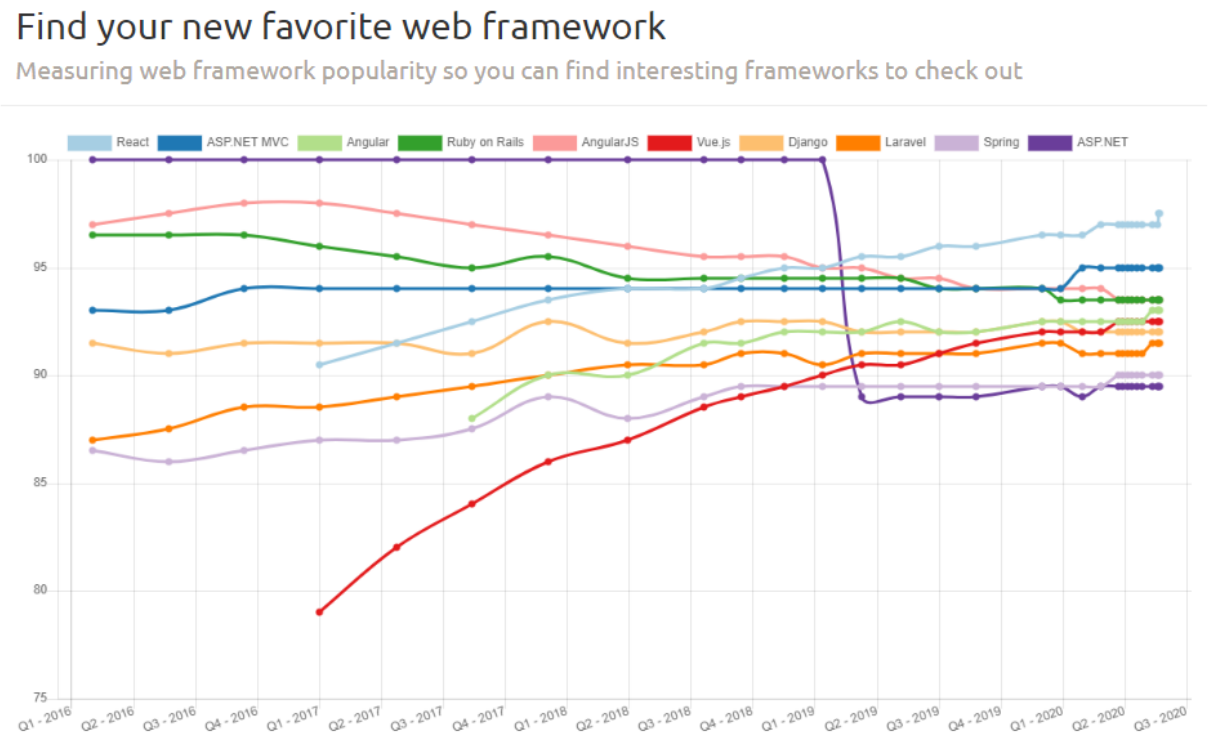

Рисунок 2.5 - Рейтинг веб фреймфорків

В попередньому пункті, мовою бекенда, було обрано Java, що значно звужує кількість доступний веб-фреймворків, тому доцільно розглядати лише Java фреймворки. Рейтинг Java фреймворків за версією сайта hotframeworks [12] наведено на рисунку 2.6.

Java

Framework	Score
Spring	90
JSF	81
Google Web Toolkit	77
JHipster	71
Blade	68
Vert.x	66
Dropwizard	64
Wicket	63
Vaadin	61
Struts	58
Tapestry	53
Restlet	52
Ninja	51
ZK	48
Mojarra	41
Stripes	40

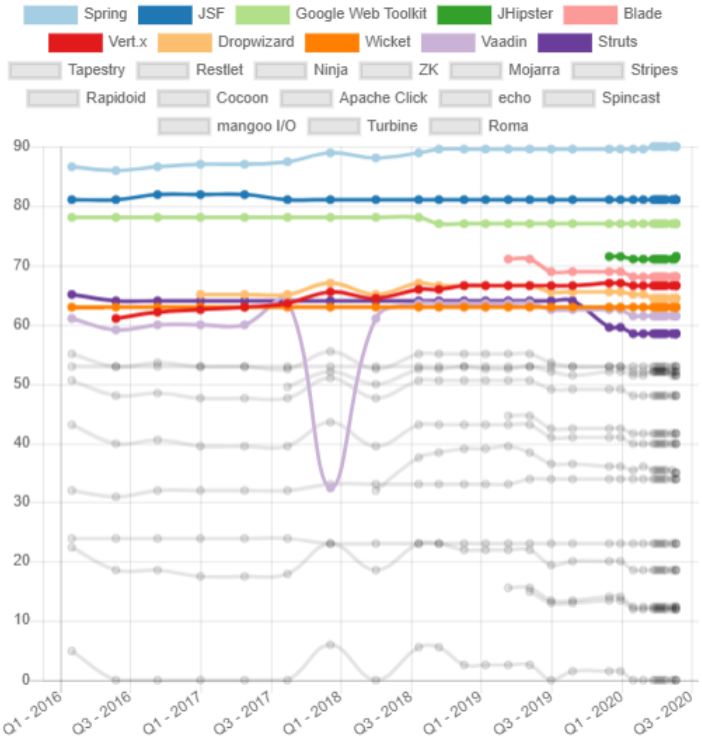

Рисунок 2.6 – Рейтинг Java фреймворків

Spring – один з найпопулярніших веб-фрейморків та найпопулярніший Java фреймворк. Розробники по всьому світу використовують Spring для створення надійних та якісних додатків.

Spring фреймворк надає розробникові багато можливостей, лише деякі наведено для прикладу нижче:

- Можливість створювати Spring MVC web-додатки та RESTful web-сервіси
- Впровадження залежності
  - JDBC шар позбавляє розробника від необхідності писати надлишковий код для роботи за базою даних
  - AOP, Aspects, Instrumentation, Messaging та інші модулі

2.5 Вибір середовища розробки

IntelliJ IDEA – інтегроване середовище розробки, створене компанією JetBrains. Підтримує програмування на мові Java, JavaScript, Python та на інших мовах. Надає розробникові достатньо велику кількість інструментів для рефакторинга і дебагінга кода, можливість використовувати сторонні плагіни, працювати з різними системами контролю версіями та багато іншого.

Починаючи с 9 версії має дві редакції: Community Edition і Ultimate Edition. Ultimate Edition – доступна під комерційною ліцензією або з використанням студентської підписки. В Ultimate Edition реалізована підтримка Java EE, UML діаграм та різних фреймворків і мов. Також в Ultimate Edition надано інструменти для тестування з підтримкою JUnit, TestNG, Spock, ScalaTest та spec2.

Підтримка Spring MVC, велика кількість підказок для Spring Boot, підтримка Thymeleaf та багато інших можливостей допомагають IntelliJ IDEA залишатись одним з найпопулярніших середовищ розробки.

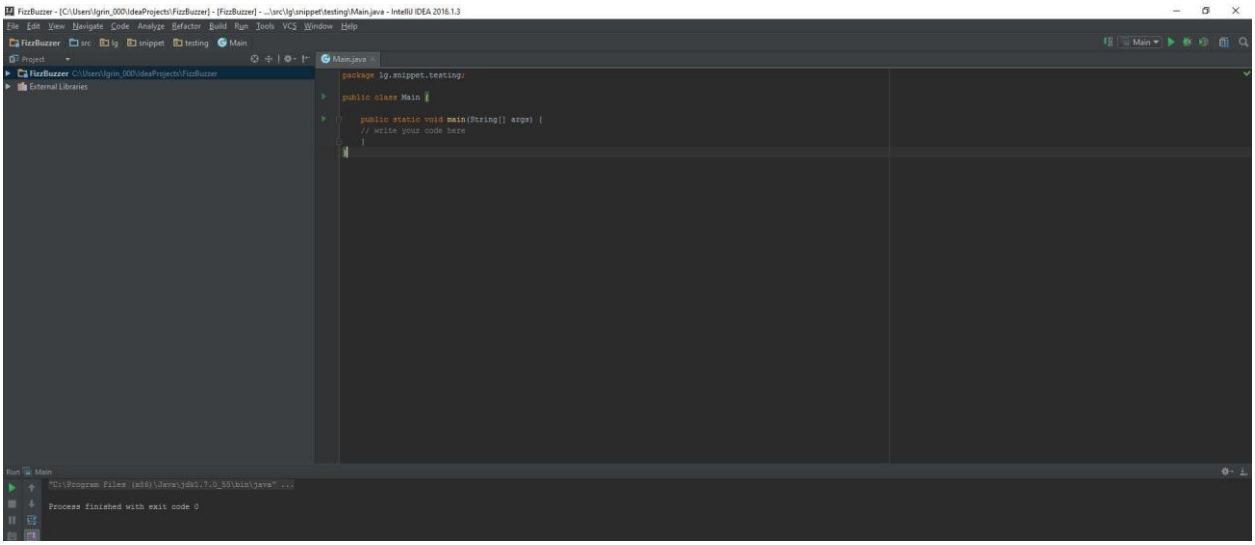

Рисунок 2.7 – Скріншот головного вікна середовища розробки Intellij idea

3 Практична реалізація

3.1 Проектування бази даних

Зобразимо отримані відносини і їх зв'язки на ER-діаграмі (див. рис. 3.1).

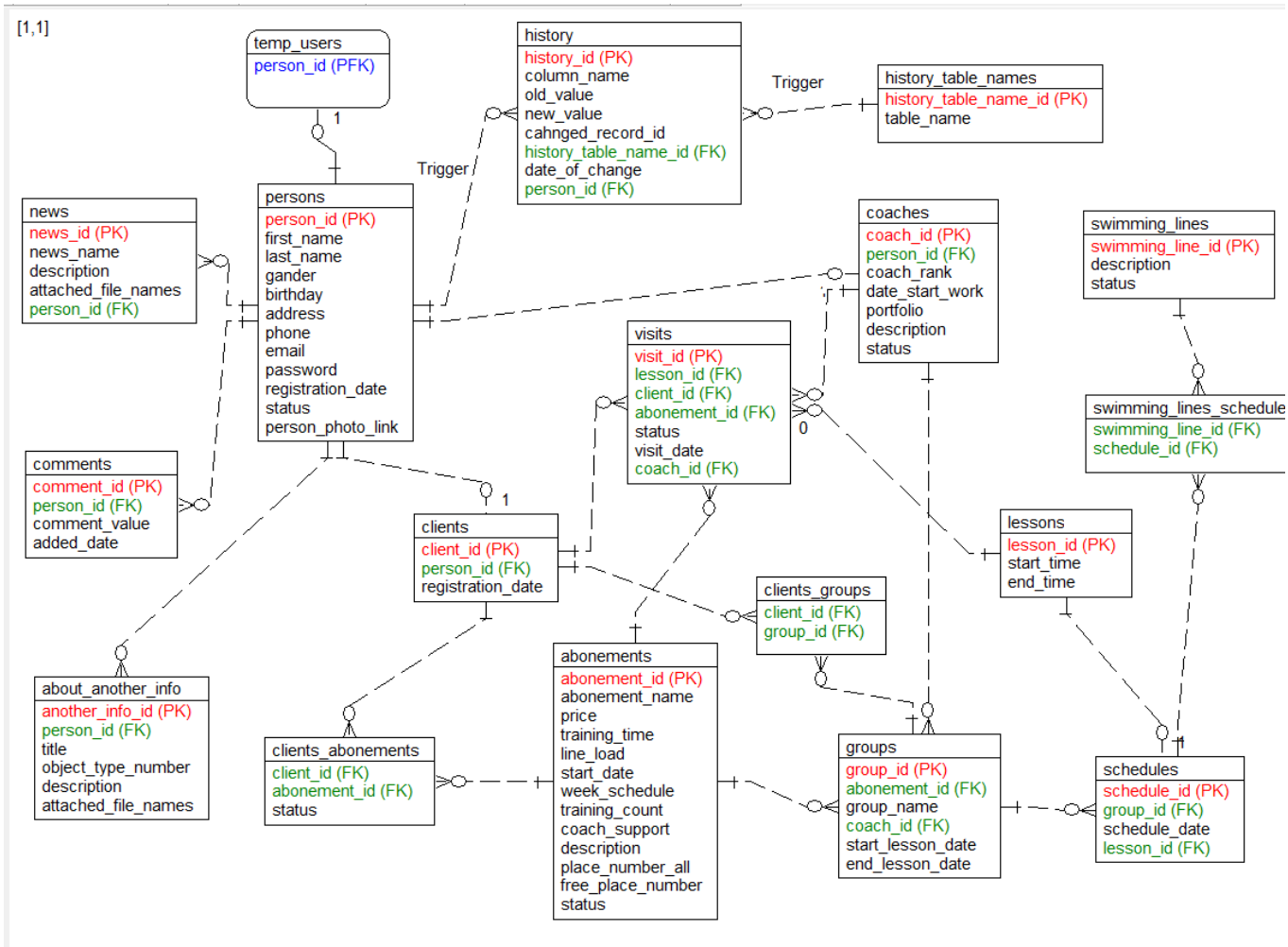

Рисунок 3.1 - ERD Інформаційної системи обліку користувачів басейну

3.2 Логічна реалізація бази даних

Проаналізувавши сутності, використані в моделі ІС, перейдемо до реалізації структури БД. Для цього представимо імена необхідних таблиць, атрибутів, типів, їх призначення та обмеження (див. табл. 3.1).

Таблиця 3.1

Структура бази даних

Таблиця	Поле	Зміст	Тип	Ключі	Обмеження
persons	person_id	Ідентифікатор особи	INTEGER	PK	Не пустий
	first_name	Ім’я особи	VARCHAR2(50)		Не пустий
	last_name	Прізвище особи	VARCHAR2(50)		Не пустий
	gander	Стать особа	VARCHAR2(6)		Не пустий
	birthday	Дата народження	DATE		Не пустий
	address	Адреса	VARCHAR2 (100)		
	phone	Телефон	NUMBER(12)		Не пустий 12 цифр
	email	Електронна адреса	VARCHAR2(50)		Унікальний
	password	Пароль	VARCHAR2(50)		
	registration_date	Дата реєстрації	Date		Не пустий
	status	Статус особи	VARCHAR2(20)		Не пустий
	person_photo_link	Шлях до картинки профілю	VARCHAR2(50)		
coaches	coach_id	Ідентифікатор тренера	INTEGER	PK	Не пустий

	person_id	Ідентифікатор особи	INTEGER	FK	Не пустий
	coach_rank	Ранг тренера	VARCHAR2(50)		
	date_start_work	Дата початку роботи тренера	DATE		Не пустий
	portfolio	Портфоліо тренера	VARCHAR2 (500)		Не пустий
	description	Додаткова інформація про тренера	VARCHAR2 (500)		
	status	Статус тренера	VARCHAR2(20)		Не пустий
clientses	client_id	Ідентифікатор клієнта	INTEGER	PK	Не пустий
	person_id	Ідентифікатор особи	INTEGER	FK	Не пустий
	registration_date	Дата реєстрації клієнта	DATE		Не пустий
groups	group_id	Номер групи	INTEGER	PK	Не пустий
	abonement_id	Номер абонемента	INTEGER	FK	Не пустий
	group_name	Назва групи	VARCHAR2(20)		Не пустий Унікальний
	coach_id	Ідентифікатор тренера	INTEGER	FK	Не пустий
	start_lesson_date	Дата початку занять	DATE		

	end_lesson_ date	Дата закінчення занять	DATE		
abonements	abonement_id	Номер абонемента	INTEGER	PK	Не пустий
	abonement_name	Назва абонемента	VARCHAR2(50)		Не пустий Унікальний
	price	Ціна	NUMBER(6)		Не пустий Не менше 0
	training_time	Час для тренування	VARCHAR2(50)		Не пустий
	line_load	Навантаження доріжку	VARCHAR2(50)		Не пустий
	start_date	Дата початку занять	DATE		Не пустий
	week_schedule	Попередній розклад занять	VARCHAR2(50)		Не пустий
	training_count	Кількість занять	NUMBER(3)		Не пустий Не менше 0
	coach_support	Підтрика тренера	VARCHAR2 (100)		Не пустий
	description	Опис	VARCHAR2 (2000)		
	place_number_ all	Всього місць	NUMBER(6)		Не пустий Не менше 0

	free_place_ number	Вільних місць	NUMBER(6)		Не пустий Не менше 0
	status	Статус	VARCHAR2(20)		Не пустий
clients_ abonements	client_id	Ідентифікатор клієнта	INTEGER	FK	Не пустий
	abonement_id	Номер абонемента	INTEGER	FK	Не пустий
	status	Статус	VARCHAR2(20)		Не пустий
visits	visit_id	Номер візиту	INTEGER	PK	Не пустий
	lesson_id	Номер пари	INTEGER	FK	Не пустий
	client_id	Ідентифікатор клієнта	INTEGER	FK	Не пустий
	abonement_id	Номер абонемента	INTEGER	FK	Не пустий
	status	Статус	VARCHAR2(20)		Не пустий
	visit_date	Дата візиту	DATE		Не пустий
	coach_id	Ідентифікатор тренера	INTEGER	FK	Не пустий
clients_ groups	client_id	Ідентифікатор клієнта	INTEGER	FK	Не пустий
	group_id	Номер групи	INTEGER	FK	Не пустий
swimming_ liness	swimming_line _ id	Номер доріжки	NUMBER(3)	PK	Не пустий Унікальний Не менше 0
	description	Додаткова інформація	VARCHAR2 (500)		
	status	Статус	VARCHAR2(20)		Не пустий

swimming_ lines_ schedule	swimming_line _ id	Номер доріжки	INTEGER	FK	Не пустий
	schedule_id	Номер заняття	INTEGER	FK	Не пустий
schedules	schedule_id	Номер заняття	INTEGER	PK	Не пустий
	group_id	Номер групи	INTEGER	FK	Не пустий
	schedule_date	Дата заняття	DATE		Не пустий
	lesson_id	Номер пари	INTEGER	FK	Не пустий
lessons	lesson_id	Номер заняття	NUMBER(3)	PK	Не пустий Не менше 0
	start_time	Час початку	VARCHAR2(5)		Не пустий
	end_time	Час закінчення	VARCHAR2(5)		Не пустий
history	history_id	Номер запису	INTEGER	PK	Не пустий
	column_name	Назва колонки	VARCHAR2 (50)		Не пустий
	old_value	Старе значення	VARCHAR2 (4000)		
	new_value	Нове значення	VARCHAR2 (4000)		
	cahnged_recod _id	Номер змінюваного Запису	INTEGER		Не пустий
	history_table _name_id	Номер змінюваної таблички	INTEGER	FK	Не пустий
	date_of_ change	Дата зміни	DATE		Не пустий
	person_id	Ідентифікатор	INTEGER	FK	Не пустий

		автора змін			
history_ table_names	history_table_ name_id	Номер таблички	NUMBER(2)	PK	Не пустий Не менше 0
	table_name	Назва таблички	VARCHAR2(40)		Не пустий Унікальний
news	news_id	Номер новини	INTEGER	PK	Не пустий
	news_name	Назва новини	VARCHAR2(50)		Не пустий Унікальний
	description	Текст новини	VARCHAR2 (4000)		Не пустий
	attached_file_ names	Посилання на файли	VARCHAR2 (200)		
	person_id	Ідентифікатор автора	INTEGER	FK	Не пустий
comments	comment_id	Номер коментаря	INTEGER	PK	Не пустий
	person_id	Ідентифікатор автора	INTEGER	FK	Не пустий
	comment_valu e	Текст коментаря	VARCHAR2 (500)		Не пустий
	added_date	Дата додавання	DATE		Не пустий
about_ another_ info	another_info_ id	Ідентифікатор запису	INTEGER	PK	Не пустий
	person_id	Ідентифікатор автора	INTEGER	FK	Не пустий
	title	Заголовок запису	VARCHAR2(20)		Не пустий

	object-type_number	Тип об’єкту	Number		Не пустий
	description	Текст запису	VARCHAR2 (2000)		Не пустий
	attached_file_names	Посилання на файли	VARCHAR2 (200)		
temp_users	person_id	Ідентифікатор Поточного користувача	INTEGER	PK	Не пустий

3.3 Розробка додатку

Під час практичної реалізації додатку можна виділити декілька основних частин:

- Розробка бази даних
- Розробка API
- Розробка інтерфейсу
- Поєднання back-end та front-end

Буде доцільно показати невеликий приклад, починаючи з запиту в базу даних та завершуючи виводом інформації в інтерфейс користувача.
Скрипт створення бази даних наведено в додатку 1, а тому не має потреби повторюватися.

Для прикладу розглянуто процес отримання доступних абонементів. Запит відбувається в базу відбувається наступним чином (див. рис.3.2).

```
@Override
public List<Abonement> getAbonements(AbonementStatus abonementStatus) {
    String query = "select abonement_id, abonement_name, price, training_time,line_load,start_date, " +
            "week_schedule,training_count,coach_support,description,place_number_all,free_place_number, status " +
            "from poolapp.abonements " +
            "where status = ?";
    return jdbcTemplate.query(query, new Object[]{abonementStatus.toString()}, new AbonementMapper());
}
```

Рисунок 3.2 – Скріншот запиту для отримання абонементів з бази даних

Для поєднання frontend та backend використовується шаблонізатор. Thymeleaf – стандартний підхід при роботі з Spring. Для цього потрібно використати інтерфейс Model в Sprning (див рисунок 3.3).

```
@GetMapping("/")
public String mainAppPage(Model model) {
    logger.info("mainAppPage start");
...
    model.addAttribute( s: "abonementsList", getAbonements(AbonementStatus.AVAILABLE));
    return "index";
}
```

Рисунок 3.3 – Скріншот передачі реультатів запиту з бази даних на frontend

Поєднання даних переданих за допомогою Thymeleaf з HTML показано на рисунку 3.4.

```
<div class="col-md-3 animate-box" th:each="abonement : ${abonementsList}">
    <div class="price-box">
        <h2 class="pricing-plan"th:inline="text">[[${abonement.abonementName}]]</h2>
        <div class="price"><sup class="currency">₴</sup><span th:text="${abonement.price}"/></div>
        <br>
        <h4>/кількість занять:<span th:text="${abonement.trainingCount}"/></h4>
        <ul class="classes">
            <li><span th:text="${abonement.trainingTime}"/></li>
            <li class="color"><span th:text="${abonement.coachSupport}"/></li>
            <li><span th:text="${abonement.lineLoad}"/></li>
            <li class="color"><span th:text="${abonement.description}"/></li>
            <li>Дата початку <br> <span th:text="${abonement.startDate}"/></li>
            <li class="color"><span th:text="${abonement.weekSchedule}"/></li>
            <li>Всього місць - <span th:text="${abonement.placeNumberAll}"/></li>
            <li class="color">Доступно місць - <span th:text="${abonement.freePlaceNumber}"/></li>
        </ul>
        <a id="abonementId" th:href="${abonement.abonementId}" href="#"
           class="btn btn-select-plan btn-sm">Обрати абонемент</a>
    </div>
</div>
```

Рисунок 3.4 – Скріншот поєднання Thymeleaf та HTML

Для користувача відображення буде відбуватись наступних чином:
потрібно перейти на головну сторінку сайту та опуститись до розділу абоменементів. Приклад почтакового відображення головної сторінки показано на рисуну 3.5, а відображення абонементів на рисунку 3.6.

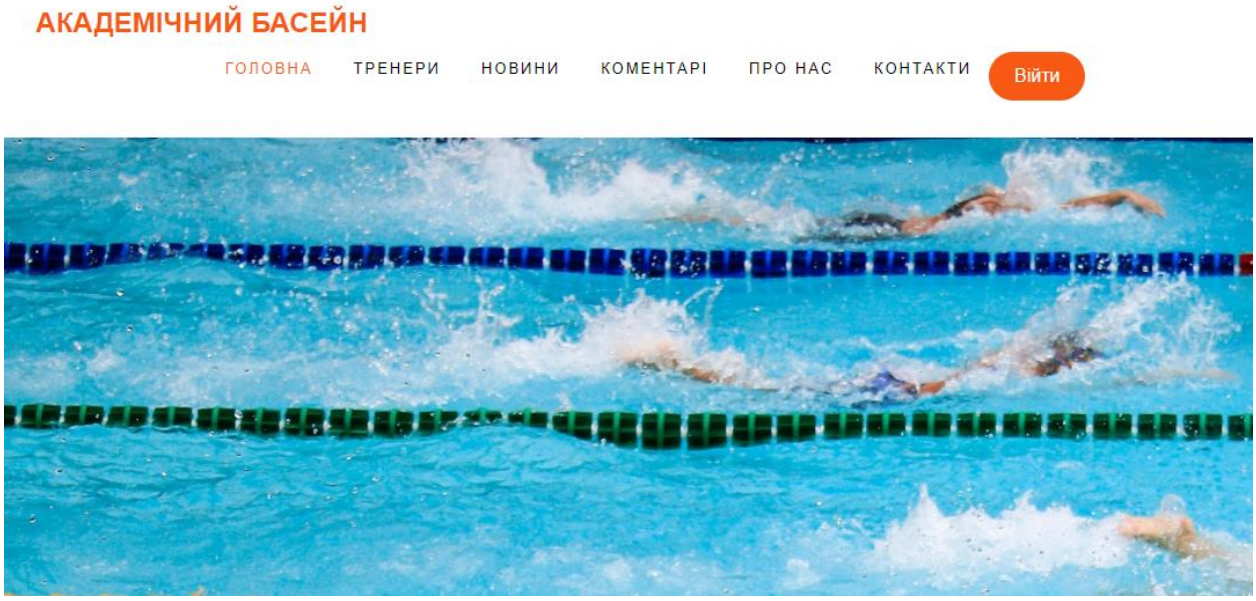

Рисунок 3.5 – Скріншот початкової сторінки

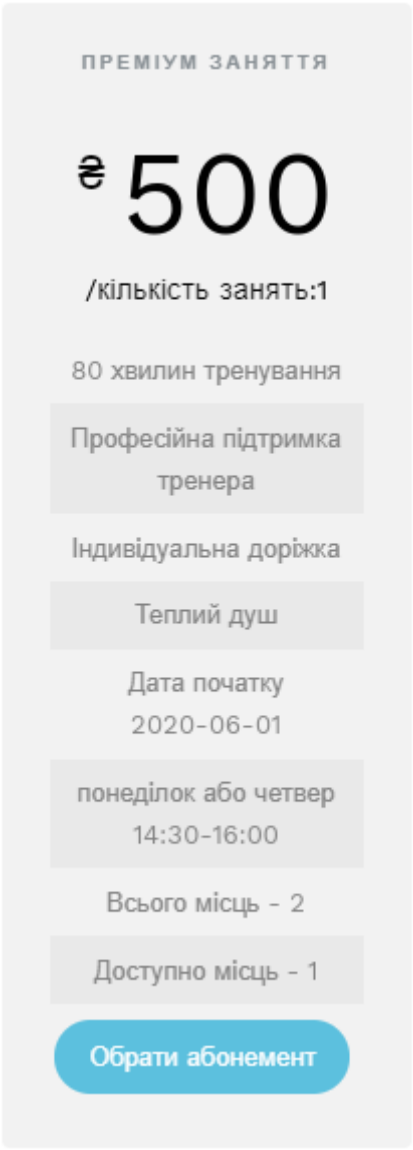

Рисунок 3.6 – Скріншот блока доступних абонементів

Висновки

В процесі виконання даної роботи було розглянуто та проаналізовано теоретичний матеріал на тему інформаційних систем. Розглянуто багато рішень, котрі частково реалізують функціонал відповідно до вимог даної роботи, але на ряду з цим мають і значні недоліки. Отримано корисний досвід до підходів реалізації різних функцій, котрий був застосований у власній системі.

У результаті аналізу був розроблений власний алгоритм створення Інформаційної системи обліку користувачів басейну. Головна задача такої системи - забезпечити повний інформаційний супровід при роботі з клієнтами, забезпечити клієнтів всією необхідною інформацією, а адміністратору надати інструменти для зручного керуванням цією системою.

Краще рішення для такої системи – web-додаток. Back end реалізовано за допомогою Spring Framework на мові Java, а Front end за допогою JavaScript. Клієнтською базою обрано Oracle СУБД, при цьому було обґрунтовано вибір всіх мов, фреймворку та середовища розробки, використаних в даній роботі.

Отриманий додаток зручний і інтуїтивно зрозумілий, без наявності надлишкових функцій. При цьому зручно використовувати такий додаток як за допомогою телефона, так і з використанням комп’ютера.

Список літератури

1. Про захист персональних даних : закон України від 1 червня 2010 р. № 2297-VI // Відомості Верховної Ради України (ВВР), 2010, № 34, ст. 481.
2. Береза А.М. Основи створення інформаційних систем: Навч. посібник. 2 видання, перероблене і доповнене – К.: КНЕУ, 2001. – 1 с. ISBN 966-574-170-5.
3. До 2023 року в Україні спрямують 1 мільярд доларів на будівництво нових водно-спортивних комплексів бумаге [Електронний ресурс] – Режим доступу: https://www.unian.ua/sport/othersports/2225229-do-2023-roku-v-ukrajini-spryamuyut-1-milyard-dolariv-na-budivnitstvo-novih-vodno-sportivnih-kompleksiv.html
4. Исследование: больше половины украинских компаний все еще ведут учет клиентов в Excel, 3% — на бумаге [Електронний ресурс] – Режим доступу: https://ain.ua/2018/04/03/issledovanie-po-crm/
5. DB-Engines Ranking of Relational DBMS [Електронний ресурс] – Режим доступу: https://db-engines.com/en/ranking/relational+dbms
6. Free Oracle Database for Everyone [Електронний ресурс] – Режим доступу: https://www.oracle.com/database/technologies/appdev/xe.html
7. TIOBE Index for May 2020 [Електронний ресурс] – Режим доступу: https://www.tiobe.com/tiobe-index/
8. Developer Survey Results 2019 [Електронний ресурс] – Режим доступу: https://insights.stackoverflow.com/survey/2019
9. Oracle Java SE Licensing FAQ [Електронний ресурс] – Режим доступу: https://www.oracle.com/technetwork/java/javase/overview/oracle-jdk-faqs.html
10. GitHut 2.0 A SMALL PLACE TO DISCOVER LANGUAGES IN GITHUB [Електронний ресурс] – Режим доступу: https://madnight.github.io/githut/#/pull_requests/2020/1

11. Find your new favorite web framework [Електронний ресурс] – Режим доступу: https://hotframeworks.com/
12. Java web frameworks [Електронний ресурс] – Режим доступу: https://hotframeworks.com/languages/java
13. Інформаційні системи, їх види; апаратне та програмне забезпечення інформаційної системи. [Електронний ресурс] – Режим доступу: http://www.kievoit.ippo.kubg.edu.ua/kievoit/2013/95/95.html

Додаток 1. Реалізація БД

```
-- create users and schemas
alter session set "_ORACLE_SCRIPT"=true;
CREATE USER poolapp IDENTIFIED BY password DEFAULT TABLESPACE users;
GRANT create session TO poolapp;
GRANT UNLIMITED TABLESPACE TO poolapp;
commit;

CREATE USER poolappconnect IDENTIFIED BY password;
GRANT create session TO poolappconnect;
commit;
-- create tables
Create table poolapp.persons (
	person_id Integer GENERATED BY DEFAULT ON NULL AS IDENTITY (START WITH 1 INCREMENT BY 1),
	first_name Varchar2 (50) NOT NULL ,
	last_name Varchar2 (50) NOT NULL ,
	gander Varchar2 (6) NOT NULL ,
	birthday Date NOT NULL,
	address Varchar2 (100),
	phone Number(12) Check (phone between 111111111111 and 999999999999 ) UNIQUE NOT NULL,
	email Varchar2 (50) UNIQUE,
	password Varchar2 (20),
	registration_date Date NOT NULL,
	status Varchar2 (20) NOT NULL ,
	person_photo_link Varchar2(50),
	UNIQUE(first_name, last_name, birthday),
primary key (person_id)
)
```

```
/
Create table poolapp.coaches (
        coach_id Integer GENERATED BY DEFAULT ON NULL AS IDENTITY (START WITH 1
INCREMENT BY 1),
        person_id Integer NOT NULL ,
        coach_rank Varchar2 (50),
        date_start_work Date NOT NULL ,
        portfolio Varchar2 (500) NOT NULL ,
        description Varchar2 (500),
        status Varchar2 (20) NOT NULL ,
primary key (coach_id)
)
/
Create table poolapp.clients (
        client_id Integer GENERATED BY DEFAULT ON NULL AS IDENTITY (START WITH 1
INCREMENT BY 1),
        person_id Integer NOT NULL ,
        registration_date Date NOT NULL,
primary key (client_id)
)
/
Create table poolapp.groups (
        group_id Integer GENERATED BY DEFAULT ON NULL AS IDENTITY (START WITH 1
INCREMENT BY 1),
        abonement_id Integer NOT NULL ,
        group_name Varchar2 (20) UNIQUE NOT NULL ,
        coach_id Integer NOT NULL ,
        start_lesson_date Date,
        end_lesson_date Date,
primary key (group_id)
)
```

```
/
Create table poolapp.abonements (
        abonement_id Integer GENERATED BY DEFAULT ON NULL AS IDENTITY (START WITH 1
INCREMENT BY 1),
        abonement_name Varchar2 (50) UNIQUE NOT NULL,
        price Number(6) Default 0 check(price >= 0) NOT NULL ,
        training_time Varchar2 (50) NOT NULL,
        line_load Varchar2 (50) NOT NULL,
        start_date Date NOT NULL,
        week_schedule Varchar2 (500) NOT NULL,
        training_count Number(3) Default 0 check(training_count >= 0) NOT NULL,
        coach_support Varchar2 (100) NOT NULL,
        description Varchar2 (2000),
        place_number_all Number(6) Default 0 check(place_number_all >= 0) NOT NULL,
        free_place_number Number(6) Default 0 check(free_place_number >= 0) NOT NULL,
        status Varchar2 (20) NOT NULL ,
primary key (abonement_id)
)
/
Create table poolapp.clients_abonements (
        client_id Integer NOT NULL ,
        abonement_id Integer NOT NULL,
        status Varchar2 (20) NOT NULL
)
/
Create table poolapp.visits (
        visit_id Integer GENERATED BY DEFAULT ON NULL AS IDENTITY (START WITH 1
INCREMENT BY 1),
        lesson_id Integer NOT NULL ,
        client_id Integer NOT NULL ,
        abonement_id Integer NOT NULL ,
```

```
	status Varchar2 (20) NOT NULL ,
	visit_date Date NOT NULL ,
	coach_id Integer NOT NULL ,
primary key (visit_id)
)
/
Create table poolapp.clients_groups (
	client_id Integer NOT NULL ,
	group_id Integer NOT NULL
)
/
Create table poolapp.swimming_lines (
	swimming_line_id Number(3) Default 0 check(swimming_line_id >= 0) NOT NULL,
	description Varchar2 (500),
	status Varchar2 (20) NOT NULL ,
primary key (swimming_line_id)
)
/
Create table poolapp.swimming_lines_schedules (
	swimming_line_id Integer NOT NULL ,
	schedule_id Integer NOT NULL
)
/
Create table poolapp.schedules (
	schedule_id Integer GENERATED BY DEFAULT ON NULL AS IDENTITY (START WITH 1
INCREMENT BY 1),
	group_id Integer NOT NULL ,
	schedule_date Date NOT NULL ,
	lesson_id Integer NOT NULL ,
primary key (schedule_id)
)
```

```
/
Create table poolapp.lessons (
	lesson_id Number(3) Default 0 check(lesson_id >= 0) NOT NULL,
	start_time Varchar2 (5) NOT NULL,
	end_time Varchar2 (5) NOT NULL,
primary key (lesson_id)
)
/
Create table poolapp.history (
	history_id Integer GENERATED BY DEFAULT ON NULL AS IDENTITY (START WITH 1
INCREMENT BY 1),
	column_name Varchar2 (50) NOT NULL ,
	old_value Varchar2 (4000),
	new_value Varchar2 (4000),
	cahnged_record_id Integer NOT NULL,
	history_table_name_id Integer NOT NULL ,
	date_of_change Date NOT NULL,
	person_id Integer NOT NULL ,
primary key (history_id)
)
/
Create table poolapp.history_table_names (
	history_table_name_id Number(2) Default 0 check(history_table_name_id >= 0) NOT
NULL,
	table_name Varchar2 (40) UNIQUE NOT NULL ,
primary key (history_table_name_id)
)
/
Create table poolapp.news (
	news_id Integer GENERATED BY DEFAULT ON NULL AS IDENTITY (START WITH 1
INCREMENT BY 1),
```

```
        news_name Varchar2 (50) UNIQUE NOT NULL ,
        description Varchar2 (4000) NOT NULL ,
        attached_file_names Varchar2 (200),
        person_id Integer NOT NULL ,
primary key (news_id)
)
/
Create table poolapp.comments (
        comment_id Integer GENERATED BY DEFAULT ON NULL AS IDENTITY (START WITH 1
INCREMENT BY 1),
        person_id Integer NOT NULL ,
        comment_value Varchar2 (500) NOT NULL ,
        added_date Date NOT NULL ,
primary key (comment_id)
)
/
Create table poolapp.about_another_info (
        another_info_id Integer GENERATED BY DEFAULT ON NULL AS IDENTITY (START WITH 1
INCREMENT BY 1),
        person_id Integer NOT NULL ,
        title Varchar2 (20) UNIQUE NOT NULL,
        object_type_number Number NOT NULL,
        description Varchar2 (2000) NOT NULL,
        attached_file_names Varchar2 (200),
primary key (another_info_id)
)
/
Create table poolapp.temp_users(
        person_id Integer NOT NULL,
primary key (person_id)
)
```

```
/
-- create foreign keys
Alter table poolapp.coaches add  foreign key (person_id) references poolapp.persons
(person_id)
/
Alter table poolapp.clients add  foreign key (person_id) references poolapp.persons (person_id)
/
Alter table poolapp.history add  foreign key (person_id) references poolapp.persons
(person_id)
/
Alter table poolapp.news add  foreign key (person_id) references poolapp.persons (person_id)
/
Alter table poolapp.comments add  foreign key (person_id) references poolapp.persons
(person_id)
/
Alter table poolapp.about_another_info add  foreign key (person_id) references
poolapp.persons (person_id)
/
Alter table poolapp.groups add  foreign key (coach_id) references poolapp.coaches (coach_id)
/
Alter table poolapp.visits add  foreign key (coach_id) references poolapp.coaches (coach_id)
/
Alter table poolapp.clients_abonements add  foreign key (client_id) references poolapp.clients
(client_id)
/
Alter table poolapp.clients_groups add  foreign key (client_id) references poolapp.clients
(client_id)
/
Alter table poolapp.visits add  foreign key (client_id) references poolapp.clients (client_id)
/
Alter table poolapp.clients_groups add  foreign key (group_id) references poolapp.groups
```

```
(group_id)
/
Alter table poolapp.schedules add  foreign key (group_id) references poolapp.groups
(group_id)
/
Alter table poolapp.clients_abonements add  foreign key (abonement_id) references
poolapp.abonements (abonement_id)
/
Alter table poolapp.groups add  foreign key (abonement_id) references poolapp.abonements
(abonement_id)
/
Alter table poolapp.visits add  foreign key (abonement_id) references poolapp.abonements
(abonement_id)
/
Alter table poolapp.swimming_lines_schedules add  foreign key (swimming_line_id) references
poolapp.swimming_lines (swimming_line_id)
/
Alter table poolapp.swimming_lines_schedules add  foreign key (schedule_id) references
poolapp.schedules (schedule_id)
/
Alter table poolapp.schedules add  foreign key (lesson_id) references poolapp.lessons
(lesson_id)
/
Alter table poolapp.visits add  foreign key (lesson_id) references poolapp.lessons (lesson_id)
/
Alter table poolapp.history add  foreign key (history_table_name_id) references
poolapp.history_table_names (history_table_name_id)
/
Alter table poolapp.temp_users add  foreign key (person_id) references poolapp.persons
```

```
(person_id)
/
commit;
```

```
-- grants for connect
GRANT SELECT ON poolapp.about_another_info TO poolappconnect;
GRANT SELECT ON poolapp.comments TO poolappconnect;
GRANT SELECT ON poolapp.news TO poolappconnect;
GRANT SELECT ON poolapp.history_table_name TO poolappconnect;
GRANT SELECT ON poolapp.history TO poolappconnect;
GRANT SELECT ON poolapp.lessons TO poolappconnect;
GRANT SELECT ON poolapp.schedules TO poolappconnect;
GRANT SELECT ON poolapp.swimming_lines_schedules TO poolappconnect;
GRANT SELECT ON poolapp.swimming_lines TO poolappconnect;
GRANT SELECT ON poolapp.clients_groups TO poolappconnect;
GRANT SELECT ON poolapp.visits TO poolappconnect;
GRANT SELECT ON poolapp.clients_abonements TO poolappconnect;
GRANT SELECT ON poolapp.abonements TO poolappconnect;
GRANT SELECT ON poolapp.groups TO poolappconnect;
GRANT SELECT ON poolapp.clients TO poolappconnect;
GRANT SELECT ON poolapp.coaches TO poolappconnect;
GRANT SELECT ON poolapp.persons TO poolappconnect;
GRANT SELECT ON poolapp.temp_users TO poolappconnect;

GRANT INSERT ON poolapp.about_another_info TO poolappconnect;
GRANT INSERT ON poolapp.comments TO poolappconnect;
GRANT INSERT ON poolapp.news TO poolappconnect;
GRANT INSERT ON poolapp.lessons TO poolappconnect;
GRANT INSERT ON poolapp.schedules TO poolappconnect;
GRANT INSERT ON poolapp.swimming_lines_schedules TO poolappconnect;
GRANT INSERT ON poolapp.swimming_lines TO poolappconnect;
GRANT INSERT ON poolapp.clients_groups TO poolappconnect;
GRANT INSERT ON poolapp.visits TO poolappconnect;
GRANT INSERT ON poolapp.clients_abonements TO poolappconnect;
GRANT INSERT ON poolapp.abonements TO poolappconnect;
```

```
GRANT INSERT ON poolapp.groups TO poolappconnect;
GRANT INSERT ON poolapp.clients TO poolappconnect;
GRANT INSERT ON poolapp.coaches TO poolappconnect;
GRANT INSERT ON poolapp.persons TO poolappconnect;
GRANT INSERT ON poolapp.temp_users TO poolappconnect;


GRANT UPDATE ON poolapp.about_another_info TO poolappconnect;
GRANT UPDATE ON poolapp.comments TO poolappconnect;
GRANT UPDATE ON poolapp.news TO poolappconnect;
GRANT UPDATE ON poolapp.lessons TO poolappconnect;
GRANT UPDATE ON poolapp.schedules TO poolappconnect;
GRANT UPDATE ON poolapp.swimming_lines_schedules TO poolappconnect;
GRANT UPDATE ON poolapp.swimming_lines TO poolappconnect;
GRANT UPDATE ON poolapp.clients_groups TO poolappconnect;
GRANT UPDATE ON poolapp.visits TO poolappconnect;
GRANT UPDATE ON poolapp.clients_abonements TO poolappconnect;
GRANT UPDATE ON poolapp.abonements TO poolappconnect;
GRANT UPDATE ON poolapp.groups TO poolappconnect;
GRANT UPDATE ON poolapp.clients TO poolappconnect;
GRANT UPDATE ON poolapp.coaches TO poolappconnect;
GRANT UPDATE ON poolapp.persons TO poolappconnect;
GRANT UPDATE ON poolapp.temp_users TO poolappconnect;

GRANT DELETE ON poolapp.comments TO poolappconnect;
GRANT DELETE ON poolapp.temp_users TO poolappconnect;
commit;
```

```
-- init script
insert into poolapp.history_table_names values(1,'poolapp.persons');
insert into poolapp.history_table_names values(2,'poolapp.news');
insert into poolapp.history_table_names values(3,'poolapp.comments');
insert into poolapp.history_table_names values(4,'poolapp.about_another_info');
insert into poolapp.history_table_names values(5,'poolapp.clients');
insert into poolapp.history_table_names values(6,'poolapp.clients_abonements');
insert into poolapp.history_table_names values(7,'poolapp.abonement');
insert into poolapp.history_table_names values(8,'poolapp.visits');
insert into poolapp.history_table_names values(9,'poolapp.coaches');
insert into poolapp.history_table_names values(10,'poolapp.clients_groups');
insert into poolapp.history_table_names values(11,'poolapp.groups');
insert into poolapp.history_table_names values(12,'poolapp.lessons');
insert into poolapp.history_table_names values(13,'poolapp.swimming_lines');
insert into poolapp.history_table_names values(14,'poolapp.swimming_lines_schedules');
insert into poolapp.history_table_names values(15,'poolapp.schedules');
commit;
/
--create or replace procedures
CREATE OR REPLACE PROCEDURE poolapp.add_history_record (v_column_name in
varchar2,v_old_value in varchar2,
v_new_value in varchar2, v_changed_record_id in integer, v_history_name_id in integer)as
begin
INSERT INTO poolapp.history select
null,v_column_name,v_old_value,v_new_value,v_changed_record_id,
                        v_history_name_id,sysdate,person_id
                        from poolapp.temp_users;
end;
/
CREATE OR REPLACE PROCEDURE poolapp.add_history_record_diffirence_value
```

```
(v_column_name in varchar2,v_old_value in varchar2,
v_new_value in varchar2, v_changed_record_id in integer, v_history_name_id in integer)as
begin
INSERT INTO poolapp.history select
null,v_column_name,v_old_value,v_new_value,v_changed_record_id,
                        v_history_name_id,sysdate,person_id
                        from poolapp.temp_users
                        WHERE NVL(v_new_value,'0') != NVL(v_old_value,'0');
end;
/
commit;
/
-- create or replace triggers
CREATE OR REPLACE TRIGGER poolapp.history_trig_persons
AFTER INSERT or UPDATE ON poolapp.persons
FOR EACH ROW
BEGIN
        IF INSERTING THEN
        poolapp.add_history_record('first_name',null,:NEW.first_name,:NEW.person_id,1);
        poolapp.add_history_record('last_name',null,:NEW.last_name,:NEW.person_id,1);
        poolapp.add_history_record('gander',null,:NEW.gander,:NEW.person_id,1);
        poolapp.add_history_record('birthday',null,:NEW.birthday,:NEW.person_id,1);
        poolapp.add_history_record('address',null,:NEW.address,:NEW.person_id,1);
        poolapp.add_history_record('phone',null,:NEW.phone,:NEW.person_id,1);
        poolapp.add_history_record('email',null,:NEW.email,:NEW.person_id,1);
        poolapp.add_history_record('registration_date',null,:NEW.registration_date,:NEW.pers
on_id,1);
        poolapp.add_history_record('status',null,:NEW.status,:NEW.person_id,1);
        poolapp.add_history_record('person_photo_link',null,:NEW.person_photo_link,:NEW.p
```

```
erson_id,1);
        ELSIF UPDATING THEN
        add_history_record_diffirence_value('first_name',:OLD.first_name,:NEW.first_name,:NE
W.person_id,1);
        add_history_record_diffirence_value('last_name',:OLD.last_name,:NEW.last_name,:NE
W.person_id,1);
        add_history_record_diffirence_value('gander',:OLD.gander,:NEW.gander,:NEW.person_i
d,1);
        add_history_record_diffirence_value('birthday',:OLD.birthday,:NEW.birthday,:NEW.pers
on_id,1);
        add_history_record_diffirence_value('last_name',:OLD.last_name,:NEW.last_name,:NE
W.person_id,1);
        add_history_record_diffirence_value('address',:OLD.address,:NEW.address,:NEW.perso
n_id,1);
        add_history_record_diffirence_value('phone',:OLD.phone,:NEW.phone,:NEW.person_id
,1);
        add_history_record_diffirence_value('email',:OLD.email,:NEW.email,:NEW.person_id,1);
        add_history_record_diffirence_value('registration_date',:OLD.registration_date,:NEW.r
egistration_date,
                        :NEW.person_id,1);
        add_history_record_diffirence_value('status',:OLD.status,:NEW.status,:NEW.person_id,
1);
        add_history_record_diffirence_value('person_photo_link',:OLD.person_photo_link,:NE
W.person_photo_link,
                :NEW.person_id,1);
END IF;
END;
/
```

```
CREATE OR REPLACE TRIGGER poolapp.history_trig_news
AFTER INSERT or UPDATE ON poolapp.news
FOR EACH ROW
BEGIN
        IF INSERTING THEN
        poolapp.add_history_record('news_name',null,:NEW.news_name,:NEW.news_id,2);
        poolapp.add_history_record('description',null,:NEW.description,:NEW.news_id,2);
        poolapp.add_history_record('attached_file_names',null,:NEW.attached_file_names,:NE
W.news_id,2);
        ELSIF UPDATING THEN
        poolapp.add_history_record_diffirence_value('news_name',:OLD.news_name,:NEW.ne
ws_name,:NEW.news_id,2);
        poolapp.add_history_record_diffirence_value('description',:OLD.description,:NEW.descr
iption,:NEW.news_id,2);
        poolapp.add_history_record_diffirence_value('attached_file_names',:OLD.attached_file
_names,
                        :NEW.attached_file_names,:NEW.news_id,2);
END IF;
END;
/
CREATE OR REPLACE TRIGGER poolapp.history_trig_comments
AFTER INSERT or DELETE ON poolapp.comments
FOR EACH ROW
BEGIN
        IF INSERTING THEN
        poolapp.add_history_record('comment_value',null,:NEW.comment_value,:NEW.comme
nt_id,3);
   ELSIF INSERTING THEN
        poolapp.add_history_record_diffirence_value('comment_value',:OLD.comment_value,:
```

```
NEW.comment_value,:NEW.comment_id,2);
        ELSIF DELETING THEN
        poolapp.add_history_record('comment_value',:OLD.comment_value,null,:OLD.commen
t_id,3);
END IF;
END;
/
CREATE OR REPLACE TRIGGER poolapp.history_trig_about_another_info
AFTER INSERT or UPDATE ON poolapp.about_another_info
FOR EACH ROW
BEGIN
        IF INSERTING THEN
        poolapp.add_history_record('title',null,:NEW.title,:NEW.another_info_id,4);
        poolapp.add_history_record('object_type_number',null,:NEW.object_type_number,:NE
W.another_info_id,4);
        poolapp.add_history_record('description',null,:NEW.description,:NEW.another_info_id,
4);
        poolapp.add_history_record('attached_file_names',null,:NEW.attached_file_names,:NE
W.another_info_id,4);
        ELSIF UPDATING THEN
        poolapp.add_history_record_diffirence_value('title',:OLD.title,:NEW.title,:NEW.another
_info_id,4);
        poolapp.add_history_record_diffirence_value('object_type_number',:OLD.object_type_
number,:NEW.object_type_number,
                        :NEW.another_info_id,4);
        poolapp.add_history_record_diffirence_value('description',:OLD.description,:NEW.descr
iption,
                        :NEW.another_info_id,4);
        poolapp.add_history_record_diffirence_value('attached_file_names',:OLD.attached_file
```

```
_names,
                        :NEW.attached_file_names,:NEW.another_info_id,4);
END IF;
END;
/
CREATE OR REPLACE TRIGGER poolapp.history_clients_abonements
AFTER INSERT or UPDATE ON poolapp.clients_abonements
FOR EACH ROW
BEGIN
        IF INSERTING THEN
        poolapp.add_history_record('abonement_id',null,:NEW.abonement_id,:NEW.client_id,6
);
        poolapp.add_history_record('status',null,:NEW.status,:NEW.client_id,6);
        ELSIF UPDATING THEN
        poolapp.add_history_record_diffirence_value('abonement_id',:OLD.abonement_id,
                        :NEW.abonement_id,:NEW.client_id,6);
        poolapp.add_history_record_diffirence_value('status',:OLD.status,
                        :NEW.status,:NEW.client_id,6);
END IF;
END;
/
CREATE OR REPLACE TRIGGER poolapp.history_abonements
AFTER INSERT or UPDATE ON poolapp.abonements
FOR EACH ROW
BEGIN
        IF INSERTING THEN
        poolapp.add_history_record('abonement_name',null,:NEW.abonement_name,:NEW.ab
onement_id,7);
        poolapp.add_history_record('price',null,:NEW.price,:NEW.abonement_id,7);
        poolapp.add_history_record('training_time',null,:NEW.training_time,:NEW.abonement_
```

```
id,7);
        poolapp.add_history_record('line_load',null,:NEW.line_load,:NEW.abonement_id,7);
        poolapp.add_history_record('start_date',null,:NEW.start_date,:NEW.abonement_id,7);

        poolapp.add_history_record('week_schedule',null,:NEW.week_schedule,:NEW.abonem
ent_id,7);
        poolapp.add_history_record('training_count',null,:NEW.training_count,:NEW.aboneme
nt_id,7);
        poolapp.add_history_record('coach_support',null,:NEW.coach_support,:NEW.aboneme
nt_id,7);
        poolapp.add_history_record('description',null,:NEW.description,:NEW.abonement_id,7)
;
        poolapp.add_history_record('place_number_all',null,:NEW.place_number_all,:NEW.abo
nement_id,7);
        poolapp.add_history_record('free_place_number',null,:NEW.free_place_number,:NEW.
abonement_id,7);
        poolapp.add_history_record('status',null,:NEW.status,:NEW.abonement_id,7);
        ELSIF UPDATING THEN
        poolapp.add_history_record_diffirence_value('abonement_name',:OL
.abonement_name,:NEW.abonement_name,
                        :NEW.abonement_id,7);
        poolapp.add_history_record_diffirence_value('price',:OLD.price,:NEW.price,:NEW.abon
ement_id,7);
        poolapp.add_history_record_diffirence_value('training_time',:OLD.training_time,:NEW.
training_time,
                :NEW.abonement_id,7);
        poolapp.add_history_record_diffirence_value('line_load',:OLD.line_load,:NEW.line_load
,:NEW.abonement_id,7);
        poolapp.add_history_record_diffirence_value('start_date',:OLD.start_date,:NEW.start_
```

```
date,
                    :NEW.abonement_id,7);
        poolapp.add_history_record_diffirence_value('week_schedule',:OLD.week_schedule,:N
EW.week_schedule,
                    :NEW.abonement_id,7);
        poolapp.add_history_record_diffirence_value('training_count',:OLD.training_count,
                    :NEW.training_count,:NEW.abonement_id,7);
        poolapp.add_history_record_diffirence_value('coach_support',:OLD.coach_support,
                    :NEW.coach_support,:NEW.abonement_id,7);
        poolapp.add_history_record_diffirence_value('description',:OLD.description,:NEW.descr
iption,
                    :NEW.abonement_id,7);
        poolapp.add_history_record_diffirence_value('place_number_all',:OLD.place_number_
all,
                    :NEW.place_number_all,:NEW.abonement_id,7);
        poolapp.add_history_record_diffirence_value('free_place_number',:OLD.free_place_nu
mber,
                    :NEW.free_place_number,:NEW.abonement_id,7);
        poolapp.add_history_record_diffirence_value('status',:OLD.status,:NEW.status,:NEW.ab
onement_id,7);
END IF;
END;
/
CREATE OR REPLACE TRIGGER poolapp.history_coaches
AFTER INSERT or UPDATE ON poolapp.coaches
FOR EACH ROW
BEGIN
        IF INSERTING THEN
        poolapp.add_history_record('coach_rank',null,:NEW.coach_rank,:NEW.coach_id,9);

        poolapp.add_history_record('date_start_work',null,:NEW.date_start_work,:NEW.coach
```

```
_id,9);
        poolapp.add_history_record('portfolio',null,:NEW.portfolio,:NEW.coach_id,9);
        poolapp.add_history_record('description',null,:NEW.description,:NEW.coach_id,9);

        poolapp.add_history_record('status',null,:NEW.status,:NEW.coach_id,9);

        ELSIF UPDATING THEN
        poolapp.add_history_record_diffirence_value('coach_rank',:OLD.coach_rank,:NEW.coac
h_rank,
                        :NEW.coach_id,9);
        poolapp.add_history_record_diffirence_value('date_start_work',:OLD.date_start_work,:
NEW.date_start_work,
                        :NEW.coach_id,9);
        poolapp.add_history_record_diffirence_value('portfolio',:OLD.portfolio,:NEW.portfolio,
                        :NEW.coach_id,9);
        poolapp.add_history_record_diffirence_value('description',:OLD.description,:NEW.descr
iption,
                        :NEW.coach_id,9);
        poolapp.add_history_record_diffirence_value('status',:OLD.status,:NEW.status,
                        :NEW.coach_id,9);
END IF;
END;
/
CREATE OR REPLACE TRIGGER poolapp.history_groups
AFTER INSERT or UPDATE ON poolapp.groups
FOR EACH ROW
BEGIN
        IF INSERTING THEN
        poolapp.add_history_record('group_name',null,:NEW.group_name,:NEW.group_id,11);
```

```
        poolapp.add_history_record('coach_id',null,:NEW.coach_id,:NEW.group_id,11);
        poolapp.add_history_record('start_lesson_date',null,:NEW.start_lesson_date,:NEW.gro
up_id,11);
        poolapp.add_history_record('end_lesson_date',null,:NEW.end_lesson_date,:NEW.group
_id,11);
        ELSIF UPDATING THEN
        poolapp.add_history_record_diffirence_value('group_name',:OLD.group_name,:NEW.gr
oup_name,
                        :NEW.group_id,11);
        poolapp.add_history_record_diffirence_value('coach_id',:OLD.coach_id,:NEW.coach_id,
                        :NEW.group_id,11);
        poolapp.add_history_record_diffirence_value('start_lesson_date',:OLD.start_lesson_dat
e,
                        :NEW.start_lesson_date,:NEW.group_id,11);
        poolapp.add_history_record_diffirence_value('end_lesson_date',:OLD.end_lesson_date
,
                        :NEW.end_lesson_date,:NEW.group_id,11);
END IF;
END;
/
CREATE OR REPLACE TRIGGER poolapp.history_lessons
AFTER INSERT or UPDATE ON poolapp.lessons
FOR EACH ROW
BEGIN
        IF INSERTING THEN
        poolapp.add_history_record('start_time',null,:NEW.start_time,:NEW.lesson_id,12);

        poolapp.add_history_record('end_time',null,:NEW.end_time,:NEW.lesson_id,12);
        ELSIF UPDATING THEN
        poolapp.add_history_record_diffirence_value('start_time',:OLD.start_time,:NEW.start_t
```

```
ime,
                    :NEW.lesson_id,12);
        poolapp.add_history_record_diffirence_value('end_time',:OLD.end_time,:NEW.end_tim
e,
                    :NEW.lesson_id,12);
END IF;
END;
/
CREATE OR REPLACE TRIGGER poolapp.history_swimming_lines
AFTER INSERT or UPDATE ON poolapp.swimming_lines
FOR EACH ROW
BEGIN
        IF INSERTING THEN
        poolapp.add_history_record('description',null,:NEW.description,:NEW.swimming_line_i
d,13);
        poolapp.add_history_record('status',null,:NEW.status,:NEW.swimming_line_id,13);
        ELSIF UPDATING THEN
        poolapp.add_history_record_diffirence_value('description',:OLD.description,:NEW.descr
iption,
                    :NEW.swimming_line_id,13);
        poolapp.add_history_record_diffirence_value('status',:OLD.status,:NEW.status,
                    :NEW.swimming_line_id,13);
END IF;
END;
/
CREATE OR REPLACE TRIGGER poolapp.history_swimming_lines_schedules
AFTER INSERT or UPDATE ON poolapp.swimming_lines_schedules
FOR EACH ROW
BEGIN
        IF INSERTING THEN
        poolapp.add_history_record('swimming_line_id',null,:NEW.swimming_line_id,:NEW.sch
```

```
edule_id,14);
        ELSIF UPDATING THEN
        poolapp.add_history_record_diffirence_value('swimming_line_id',:OLD.swimming_line_
id,:NEW.swimming_line_id,
                        :NEW.schedule_id,14);
END IF;
END;
/
CREATE OR REPLACE TRIGGER poolapp.history_schedules
AFTER INSERT or UPDATE ON poolapp.schedules
FOR EACH ROW
BEGIN
        IF INSERTING THEN
        poolapp.add_history_record('schedule_date',null,:NEW.schedule_date,:NEW.schedule_i
d,15);
        poolapp.add_history_record('lesson_id',null,:NEW.lesson_id,:NEW.schedule_id,15);

        ELSIF UPDATING THEN
        poolapp.add_history_record_diffirence_value('schedule_date',:OLD.schedule_date,:NE
W.schedule_date,
                        :NEW.schedule_id,15);
        poolapp.add_history_record_diffirence_value('lesson_id',:OLD.lesson_id,:NEW.lesson_i
d,
                        :NEW.schedule_id,15);
END IF;
END;
/
commit;
```