Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

РОЗРОБКА СИСТЕМИ УПРАВЛІННЯ ПЕРСОНАЛОМ З ЗАСТОСУВАННЯМ БАЗ ДАНИХ ТА ЗНАНЬ

Текстова частина до курсової роботи

за спеціальністю „Комп’ютерні науки” 122

Керівник курсової роботи

к.т.н., ст. в Ющенко Ю.О.

_____

“____” _____ 2020 р.

Виконала студентка Тютюн

Марина Миколаївна

“____” _____ 2020 р.

Київ 2020

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ

Зав.кафедри інформатики,

проф., д.ф.-м.н.

_____М. М. Глибовець

(підпис)

„____”_____2020 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студенту_____ факультету _____ _____курсу

ТЕМА «РОЗРОБКА СИСТЕМИ УПРАВЛІННЯ ПЕРСОНАЛОМ З ЗАСТОСУВАННЯМ БАЗ ДАНИХ ТА ЗНАНЬ»

Вихідні дані: програмний застосунок для системи керування персоналом.

Зміст ТЧ до курсової роботи:

Індивідуальне завдання

Вступ

1 Огляд інформаційних систем керування персоналом, аналіз існуючих систем.

2 Аналіз задачі та засобів розробки.

3 Розробка продукту.

Висновки

Список літератури

Додатки

Дата видачі „___” _____ 2020 р. Керівник _____

(підпис)

Завдання отримав _____

(підпис)

Календарний план

Тема: «Розробка системи управління персоналом з застосуванням баз даних та знань».

Календарний план виконання роботи:

№ п/п	Назва етапу курсової роботи	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу	10.10.2019	
2.	Огляд технічної та наукової літератури за темою роботи.	29.11.2019	
3.	Виконати аналіз аналогів системи.	06.12.2019	
4.	Розробка алгоритму та структури програми.	27.12.2019	
5.	Розробка програмного продукту.	07.02.2020	
6.	Тестування програмного продукту, заповнення бази тестових даних	06.03.2020	
7.	Виконати перевірку роботи програми на відповідність визначеним вимогам.	20.03.2020	
8.	Написання пояснювальної роботи.	24.04.2020	
9.	Створення презентації для захисту роботи.	28.04.2020	
10.	Аналіз отриманих результатів з керівником.	04.05.2020	
11	Корегування роботи згідно зауважень керівника	07.05.2020	
12.	Остаточне оформлення пояснювальної роботи та презентації.	11.05.2020	
13.	Захист курсової роботи	29.05.2020	

Зміст

Перелік умовних позначень

БД – база даних.

СКБД – система керування базами даних.

БЗ – база знань.

ID (від англ. – Identity Document) – унікальний ідентифікатор об’єкта, в таблицях баз даних виступає як первинний ключ.

NDA (від англ. – Non-disclosure Agreement) – угода про нерозголошення, договір про конфіденційність.

HR (від англ. – Human resources) – людські ресурси.

Проджект менеджер (від англ. – Project manager) – керівник проекту, який узгоджує дії команди та відповідає за досягнення поставленої мети.

ІС – інформаційна система.

ПЗ – програмне забезпечення.

ЗП – заробітна плата.

OSI – абстрактна мережева модель.

HTML (від англ. HyperText Markup Language) – мова розмітки гіпертексту.

PWA (від англ. – Progressive Web App) – прогресивний веб-застосунок, гібрид звичайної веб-сторінки та мобільного додатку.

UML (від англ. – Unified Modeling Language) – уніфікована мова моделювання.

URL (від англ. – Uniform Resource Locator) – єдиний вказівник на ресурс.

SQL (від англ. – Structured Query Language) – декларативна мова структурованих запитів.

CRUD (від англ. – Create/Read/Update/Delete) – набір основних операцій над даними, а саме створення, читання, оновлення та видалення.

ISO (від англ. – International Organization for Standardization) – міжнародна організація, що займається створенням стандартів.

ANSI (від англ. – American National Standards Institute) – об'єднання американських груп для розробки стандартів, член ISO.

ООП – об’єктно-орієнтоване програмування.

ORM (від англ. – Object-Relational Mapping) – об'єктно-реляційна проекція.

Мапінг (від англ. – mapping) – співставлення.

Вступ

Трудові ресурси – найголовніший аспект діяльності будь-якої організації, оскільки від кваліфікації та продуктивності робітників залежить напрям та швидкість розвитку бізнесу. Питання продуманого керування персоналом має бути одним із ключових аспектів стратегії розвитку підприємства. На даний момент немало компаній зберігають дані у паперовому вигляді або в електронних таблицях, тим самим унеможливлюючи швидкий доступ до них та їх аналіз. Окремим аспектом є адаптація нових співробітників, особливо з малим досвідом роботи: зазвичай під час випробувального терміну робітники не приносять компанії великий прибуток, особливо при неправильній організації навчального процесу.
Автоматизована інформаційна система з доступом на основі ролей може забезпечити єдиний та миттєвий доступ до даних для всіх зацікавлених осіб, автоматизувати задачі звітності, документообігу, навчання нових співробітників, понизити навантаження на HR-спеціалістів і таким чином дозволить компанії зменшити витрати та сфокусуватися на розвитку бізнесу.

На даний момент існує немало готових інформаційних управління персоналом, проте вони більшість має обмежений функціонал та потребує настройки під вимоги конкретної організації та немалі витрати на впровадження, подальший супровід та навчання персоналу.

Мета курсової роботи – проектування та розробка універсальної системи з усіма необхідними компонентами, швидким та надійним доступом до даних, простою системою налаштувань та інтуїтивно зрозумілим інтерфейсом. Під час проектування системи буде проведено дослідження та аналіз існуючих рішень, виявлення недоліків, визначення необхідних компонентів системи та розробка рекомендацій для підвищення ефективності. В якості джерел дослідження виступатимуть документації інформаційних систем, наукові статті, присвяченні керуванню персоналу та

впровадженню ІС, технічна література та документація щодо обраних засобів програмування та СКБД.

Робота складається з трьох розділів.

Перший розділ присвячено аналізу предметної області, огляду існуючих інформаційних систем, їх аналізу та порівняння. Визначені основні компоненти подібних інформаційних систем та рекомендації при розробці.

Другий розділ присвячений аналізу засобів розробки та обґрунтування вибору, наведені теоретичні відомості про задачу та підходи до її реалізації.

Третій розділ присвячено проектуванню власного рішення, опису структури та функціональності системи, аналізу використання найоптимальніших прийомів, засобів та бібліотеки для реалізації функціоналу. Виконано аналіз власного рішення та порівняння з існуючими системами.

РОЗДІЛ 1: Огляд структури і функціоналу інформаційних систем керування персоналом, аналіз існуючих систем

1.1 Аналіз предметної області, історія управління людськими ресурсами

Управління людськими ресурсами – стратегічний та чіткий підхід до управління найціннішими активами організації – її співробітниками, які індивідуально та колективно сприяють досягненню її цілей [1]. Простіше кажучи, HR та його підрозділи створені для підвищення ефективності роботи працівників для вигоди їх роботодавця та відповідальні за низку наступних видів діяльності: пошук та прийняття нових працівників на роботу, їх навчання та професійний розвиток, оцінку ефективності та винагородження, улагодження виробничих відносин.

Система керування персоналом мала місце з давніх часів, проте зазнала значних змін до її теперішнього вигляду [2]. Навіть у часи до нашої ери існували спеціальні робітники, що слідкували за порядком інших, наприклад у Древньому Китаї та Греції [2]. З початком технологічного прогресу до 1940-х років так звані менеджери з персоналу виконували роботу, типову для HR-спеціалістів, причому ефективно, але слабо пов’язану з цілями організації. З 1940-х років по 1960-і HR-спеціаліст стає окремою професією, в управлінні людськими ресурсами є фокус на продуктивність робітників за допомогою мотиваційних технік, виникають описи посад для покращення відбору кандидатів, виникнення стратегій оцінки та винагородження. Наступний етап розвитку HR (1960-і – 1980-і) відзначився впровадженням цифрових технологій для зберігання інформації та звітності, підвищення кваліфікації та розширення можливостей працівників. У 1980-поч. 1990 автоматизація збільшила продуктивність роботи, пройшов перехід від адміністрації працівників до розвитку та залучення працівників. Наступний етап, з 1990-х дотепер, характеризується значним технологічним проривом, що відобразився на системі керування персоналом, а саме поява вдосконалених стратегій для залучення, розвитку та

утримання кандидатів, виникли нові методи оцінювання роботи, електронний відбір, віртуальні команди, покращений нетворкінг тощо.

На даний момент в HR сфері прослідковується тенденція робити акцент на розвиток та постійне навчання персоналу за участі у різноманітних курсах, конференціях, вебінарах, заохочується обмін знаннями, який у підсумку зменшує кількість неоплачуваної клієнтом роботи, практикується самостійне оцінювання та відслідковування робітником свого прогресу.

Одним із найголовніших аспектів системи керування персоналом є надійне зберігання даних та безпомилкова обробка, чого не завжди можна зустріти в реальності – дані про одну особу можуть зберігатися в різних місцях у вигляді електронних таблиць, що унеможливлює швидкий доступ до них та пошук. Іншою проблемою може бути передача знань для нових працівників – при великій кількості охочих можна влаштувати воркшоп чи семінар, але це є затратно та неефективно у випадку найму кількох людей.

Отже, для вирішення типових проблем при працевлаштуванні нових кандидатів та стратегічних цілей бізнесу необхідне впровадження єдиної інформаційної система, в основі якої буде міститися база даних задля збереження даних про співробітників, історію їх роботи на навчання, та база знань для управління і швидким доступом усіх необхідних відомостей про організацію для всіх зацікавлених осіб.

1.2 Структура і функції автоматизованої системи керування персоналом

Автоматизована інформаційна система – організаційно-технічна система, що складається із засобів автоматизації певного виду чи кількох видів діяльності людей та персоналу, що здійснює цю діяльність [3]. Система керування персоналом включає в себе програмне забезпечення та групу користувачів, які нею користуються – HR-спеціалісти, звичайні робітники, можливо, проджект-менеджери та бухгалтери.

Програмне забезпечення, в свою чергу, поділяється на декілька підсистем, кожна з яких виконує задачі певної групи користувачів. Насамперед система має містити в собі всі необхідні дані про співробітників, включаючи копії документів (резюме, договорів про початок співробітництва, паспортні дані, NDA тощо), що значно полегшить роботу HR-відділу та бухгалтерії. Система має містити перелік всіх посад, необхідні компетенції і навички до кожної з них та рівень заробітної плати, що полегшить процеси перегляду винагородження та скорочення штату.

Надзвичайно важливим компонентом системи керування персоналом є модуль навчання. Він дозволить назначати курси як і загальні для ознайомлення з компанії – для пришвидшення процесу адаптації, так і спеціалізовані – для підвищення рівня знань співробітників. Постійна перепідготовка кадрів ефективна не лише для високотехнологічних компаній, а і для торгових, що постійно реалізують нові види товарів, оскільки доскональна обізнаність персоналу з продукцією збільшить рівень та якість продаж.

Ще одним необхідним компонентом є управління відпусками та лікарняними, та щоденне ведення звітності за пророблену роботу, що є корисним для проджект-менеджерів для контролю роботи співробітників та демонстрації клієнту поточного етапу роботи.

Базовий функціонал системи керування персоналом відповідає наступним пунктам:

а) надійне зберігання та можливість оновлення інформації про співробітників, додавання документів, навичок, сертифікатів тощо;

б) управління заробітною платою – відомості про поточний рівень, відомості про заслуги для преміальних виплат;

в) створення та можливість проходження навчальних курсів для підвищення кваліфікації та адаптації нових співробітників;

г) контроль робочого процесу, графіку та відсутності співробітників, починаючи зі створення заявок, закінчуючи їх підтвердження вповноваженими особами.

Проте не всі системи мають обмежуватися цим функціоналом та можуть виконувати наступні вимоги:

а) облік особливих виплат – медичних та пенсійних інвестицій, які додаються до основної заробітної плати;

б) інтеграція з ресурсами пошуку роботи для автоматизації процесу розміщення вакансій, перегляду відгуків та комунікації з кандидатами;

в) план кар’єрного розвитку для огляду продуктивності співробітника, визначення строків та необхідних засобів для підвищення спеціалізації.

Отже, сучасні системи керування персоналом розбиваються над модулі та підсистеми для різних цілей та видів користувачів та можуть виконувати ряд функцій в залежності від потреб клієнтів.

1.3 Аналіз існуючих систем

На даний момент на ринку існує немало програмного забезпечення для керування персоналом, яке відповідає потребам бізнесу різного розміру та напрямку, проте моя задача полягає у створенні системи для організації будь-якого масштабу та спеціалізації. Для реалізації цього необхідно провести аналіз найпопулярніших аналогів, щоб виявити типові проблеми та недоліки та визначити способи їх усунення.

Для виявлення основних тенденцій серед існуючих систем я проведу аналіз трьох систем: Fingercheck, BambooHR та UltiPro. Дані системи призначені для впровадження в організації будь-якого напрямку, проте різних розмірів – для малого, середнього та великого бізнесу відповідно.

Після ознайомлення із документацією вищевказаних систем [4-6] та порівняльного аналізу я виявила схожі риси: кожна з них містить в собі модулі Онбордингу (від англ. – Onboarding – процес ознайомлення із робочими процесами, проектами, командою та корпоративною культурою та адаптації нового співробітника), управління оплатою праці, систему відстеження кандидатами та основні HR функції – зберігання та управління інформації про співробітників, враховуючи заробітну плату та додаткові виплати, в єдиній централізованій базі даних.

Проте проаналізувавши дані ПЗ та кілька аналогів у кожній із категорій (ІС для малого, середнього та великого бізнесу) на рахунок наявності інших підсистем та функціоналу, я виявила ряд відмінностей: низка функцій наявні або відсутні лише у одній із них. В таблиці 1.1 наведено порівняння основних функцій вищевказаних систем керування персоналом.

Таблиця 1.1 – Порівняльний аналіз ІС Fingercheck, BambooHR та UltiPro

Інформаційні системи

Наявність підсистем	Fingercheck	BambooHR	UltiPro
Управління робочою силою	Ні	Так	Ні
Управління продуктивністю	Ні	Так	Так
Відстеження робочого часу	Так	Ні	Так
Відстеження рекрутингу	Ні	Так	Так
Корпоративна система управління навчанням	Ні	Так	Так
Створення розкладу	Ні	Ні	Так
Автоматизований парсинг резюме	Ні	Ні	Так
Створення звітів	Ні	Так	Так

Отже, ряд надзвичайно корисних функцій присутній або у окремих системах, або у системах для організацій певних розмірів, що безсумнівно зменшує можливості розвитку бізнесу, оскільки доводиться вибирати між доступною ціною та багатофункціональністю. Для вирішення даної проблеми необхідно створити систему з усіма необхідними модулями, які при відсутності необхідності можна деактивувати, та з використанням легко масштабованої бази даних, яка витримає навантаження при збільшенні кількості даних в кілька разів.

РОЗДІЛ 2: Аналіз задачі та засобів розробки

2.1 Аналіз поставленої задачі

При розробці системи необхідно визначити її основні складові, способи їх взаємодії, вимоги до кінцевих користувачів і, виходячи з цього, архітектуру. В залежності від предметної області треба розробити бізнес-логіку програми, яка буде найточніше відповідати дійсності.

Інформаційні системи для управління – системи на основі баз даних, що надають інформацію для персоналу організації [7] і складаються в основному з наступних компонентів:

а) апаратного забезпечення;

б) програмного забезпечення, що в свою чергу ділиться на компоненти, кожен з яких забезпечує:

1) ввід даних;
2) зберігання даних;
3) обробку даних;
4) виведення інформації.

в) власне даних;

г) процедур з використання, експлуатації та обслуговування системи, тобто документації;

д) обслуговуючого персоналу [8].

Під час розробки ІС необхідно забезпечити швидку, надійну та безперебійну взаємодію цих компонентів між собою та роботу кожного з них по окремості, а також треба враховувати можливість великої кількості користувачів з різними технічними

можливостями, тому за архітектурою система має обов’язково бути розподіленою, тобто СКБД та БД буде зберігатися на сервері, а клієнтські застосування – на робочих станціях.

Іншим важливим аспектом при розробці є розбиття логіки програми на слабо пов’язані компоненти, що робить розробку гнучкішою, а тестування – простішим. У випадку даної інформаційної системи, модулі програми відповідають групам користувачів у ній. Кожен користувач має права у рамках своєї ролі і не має доступу до сторонніх даних. Ролі користувачів є наступними:

а) звичайні робітники;
б) HR-менеджери;
в) менеджери з навчання персоналу;
г) адміністратори.

Звичайні робітники можуть переглядати власні дані, змінювати їх частину, звітувати робочий час, проходити навчальні модулі. HR-менеджери можуть вносити дані про нових робітників у систему, змінювати їх, керувати поточними позиціями, ЗП та необхідними для них навичками, а також призначати навчання співробітникам. Менеджери з навчання персоналу відповідальні за створення навчальних курсів та бази знань. Адміністратор має повні права на читання та зміну даних, може створювати користувачів та призначати їм ролі, активувати чи дезактивувати компоненти системи. Також необхідно забезпечити розробку бази знань та зручну навігацію по ній для пошуку необхідної інформації, яка буде доступна для всіх користувачів.

Отже, ключовий момент у задачі створення інформаційної системи керування – правильне розбиття її та логіки програми на компоненти, забезпечення їх коректної взаємодії та визначення обмежень для кінцевих користувачів.

2.2 Аналіз методів та способів реалізації задачі

Архітектура інформаційної системи через велику кількість кінцевих користувачів із машинами різної потужності, як зазначалося вище, має бути розподіленою, а саме клієнт-серверною. Цей підхід забезпечить розділення відповідальності між фронт-ендом (відповідає за рівень представлення в моделі OSI[9]) та бек-ендом (рівнем доступу до даних).

Одним із шаблонів проектування, що забезпечить відповідність даній архітектурі є Модель-Вид-Контролер (далі – MVC, від англ. Model-View-Controller), що забезпечує розбиття програми на три компоненти:

а) модель – є найголовнішою складовою шаблону, відображає структуру програми без прив’язки до реалізації, безпосередньо керує даними і її логікою;

б) вид (представлення) – будь-яке представлення інформації у вигляді таблиць, діаграми, графіків або HTML-сторінок без логіки;

в) контролер – частина програми, що приймає дані та перетворює їх на команди для моделі або контролеру[10].

Взаємодію складових шаблону MVC між собою та користувачем наведено у додатку А. Завдяки розділенню відповідальності між окремими компонентами програми забезпечується надійна обробка та зберігання даних (оскільки валідація та перевірка на відповідність правилам бізнес-логіки виконується на сервері) та зменшується навантаження на клієнта.

Отже, компонентом шаблону, що взаємодіє з даними, є контролер, проте розміщувати в ньому код для прямого зв’язку з БД є недоцільним з точки зору його чистоти, подальшого доповнення та тестування. Щоб інкапсулювати логіку доступу до даних доцільно використовувати шаблон Repository, що дозволяє абстрагуватися від

конкретних підключень до джерел даних, та є проміжною ланкою між класами ORM-фреймворків, що безпосередньо взаємодіють з даними, і рештою програми. Паттерн Repository полягає у створенні окремого компоненту для доступу і обробки даних для кожної із сутностей, що відповідає таблиці БД.

Часто разом із «Репозиторієм» використовується шаблон Service Layer, що інкапсулює бізнес-логіку застосунку, встановлюючи набір доступних операцій та координує відповідь програми на кожну із них. Як і в попередньому патерні, сервіси розбиті на логічні рівні для взаємодії із кожною окремою сутністю. Поєднання цих архітектурних підходів дозволяє легко змінювати бізнес логіку та методи взаємодії із даними та не нагромаджувати код.

Ще одним способом покращення структури програми за допомогою збільшення її модульності та можливості швидкого розширення є використання принципу інверсії контролю – певного набору рекомендацій, що дозволяє проектувати застосунки зі слабим зв’язуванням окремих компонентів. Одним із механізмів реалізації даного принципу є впровадження залежностей, що полягає у наданні зовнішньої залежності програмному компоненту. Простіше кажучи, він надає можливість об’єкту отримувати доступ до певного сервісу, не викликає напряму методи для реалізації його окремого типу, а дозволяє фреймворку робити це автоматично – при створенні об’єкт уже матиме посилання на потрібний йому сервіс.

Отже, важливим етапом при проектуванні системи є розділення її на логічні компоненти, що залежать від ролей та прав доступу користувачів, та рівні, кожен із яких відповідальний тільки за одну задачу, наприклад, абстракцію доступу до даних або визначення правил бізнес-логіки.

РОЗДІЛ 3: Розробка продукту

3.1 Визначення вимог до розробки програмного продукту

Мета даної курсової роботи – розробка програмного забезпечення для управління персоналом із застосуванням баз даних та знань. Для реалізації цієї задачі необхідно визначити структуру бази даних та власне системи.

В якості сховища даних було обрано реляційну СКБД через їх розповсюдженість та легкість використання. Далі, після аналізу предметної області, має бути розроблена схема даних, створені відповідні таблиці, зв’язки між ними, визначені обмеження домену, створені необхідні представлення та процедури.

Через відсутність обмежень на тип ПЗ вибір був зупинений на створенні веб-застосунку – вони платформо незалежні, не займають пам’яті на машині користувача, при перетворені їх на PWA можуть застосуватися на будь-яких пристроях. На даному етапі розробки система призначена для десктопних пристроїв з сучасними веб-браузерами, але має адаптивний дизайн, тобто може відкриватися з мобільних телефонів, планшетів тощо.

Оскільки для даних інформаційних систем найдоречніше використовувати клієнт-серверну архітектуру, використовуючи шаблон MVC, а зменшення залежності між програмними компонентами – шаблони Repository та Service Layer разом з механізмом Dependency Injection (див. розділ 2.2), то в ній мають обов’язково міститися наступні компоненти:

а) моделі даних, що відповідатимуть реальним таблицям;

б) представлення, тобто веб-сторінки, для проведення всіх можливих операцій над даними для всіх користувачів (робітники, HR-менеджери, менеджери з навчання персоналу та адміністратори);

в) контролери для взаємодії між моделями та представленнями;

г) сервіси, що визначатимуть перелік можливих операцій над даними;

д) репозиторії для інкапсуляції доступу до даних;

е) інтерфейси для реалізації впровадження залежностей для сервісів та репозиторіїв.

Отже, маючи конкретні вимоги до реалізації, необхідно розробити алгоритм та структуру програми, провести аналіз та вибір засобів розробки.

3.2 Обґрунтування алгоритму та структури програми

Після уточнення вимог до реалізації необхідно розробити структуру та алгоритм програми, які будуть найточніше їм відповідати та забезпечувати її надійну роботу.

Після детального аналізу предметної області та існуючих аналогів були розроблені функціональні вимоги, які має задовольняти система, які наведена на рисунку 3.1 у формі UML-діаграми.

Рисунок 3.1 – Функціональні вимоги системи керування персоналом

В даній діаграмі фігури людей позначають користувачів системи, овали – функціонал, лінії – зв'язок між користувачами та вимогами. Кожен користувач, окрім адміністратора, має обмежений перелік дозволених дій, що дозволяє розбити цілу систему на незалежні підсистеми для роботи кожного з них. Це полегшує процес розробки програми та подальшого її тестування.

Для обмеження прав доступу користувачів та забезпечення зв'язку між підсистемами необхідно розробити список ролей, визначити їх обмеження та пов’язати їх із зареєстрованими користувачами. Наступним кроком є розробка механізму авторизації та аутентифікації на основі ролей, що забезпечуватиме доступ лише до певної підсистеми (рисунок 3.2).

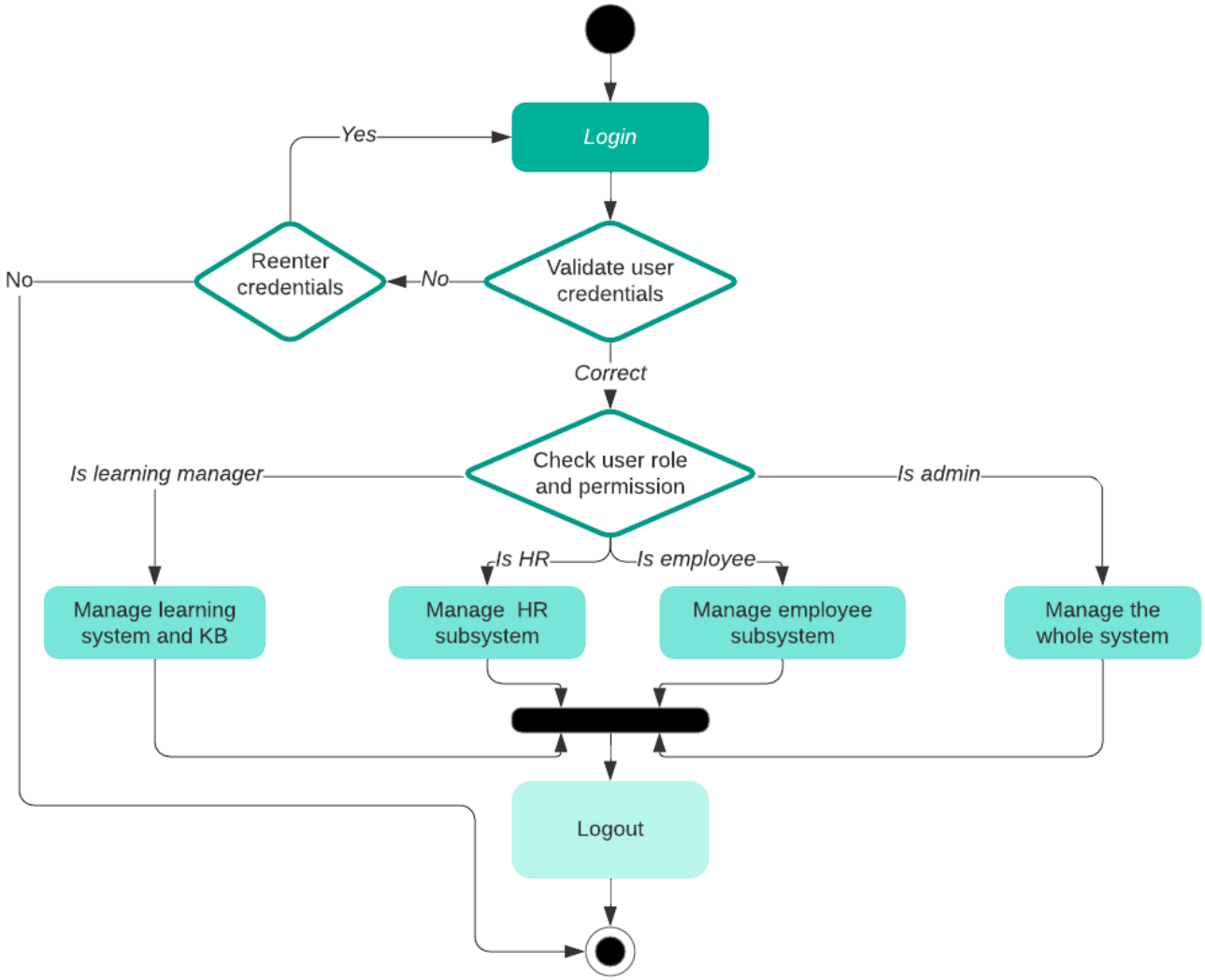


Рисунок 3.2 – Схема аутентифікації користувачів у системі

При успішному вході у систему здійснюється перевірка ролі користувача і відбувається перехід на стартову сторінку підсистеми. Іншим важливим аспектом є обмеженням доступу до складових підсистем не лише на етапі аутентифікації, а і при загальному використанні. Це дозволить уникнути несанкціонованого доступу до

заборонених сторінок за допомогою вводу URL – в такому випадку користувача переадресує на сторінку входу.

На стартовій сторінці доцільно розмістити меню з усіма доступними для підсистеми діями (наприклад, для HR – це перегляд та редагування вакансій, списку всіх користувачів, підтвердження відпусток тощо). Користувач може виконати будь-яку з них при натисканні на пункт меню або повернутися до стартової сторінки.

Отже, алгоритм роботи програми має бути наступним: інформаційна система має розбиватися на незалежні компоненти; для взаємозв'язку між ними використовується керування доступом на основі ролей: кожному користувачу призначається роль, в залежності від неї накладаються обмеження на ряд дозволених дій (функціональних вимог), користувач може перейти до кожної з них за допомогою навігаційного меню.

3.3 Обґрунтування вибору засобів розробки

Мовою програмування для розробки інформаційної системи було обрано C# через доволі глибокі знання та наявність в ній бібліотек, що допомагають виконати поставлені вимоги.

Для створення веб-сайту ІС було обрано фреймворк ASP.NET MVC – технологію створення веб-сервісів та веб-застосунків від компанії Microsoft, що реалізує шаблон проектування MVC. Засоби середовища розробки Microsoft Visual Studio для роботи із даним фреймворком дозволяють автоматично створювати представлення, що відповідають конкретній дії у базі даних (перегляд даних про модель, сторінки додавання та редагування тощо) та дозволяють легко пов'язати їх з моделлю за допомогою контролера – достатньо створювати класи з однаковим ім'ям.

Даний фреймворк містить багато бібліотек для надійного зберігання та обробки даних. Так, простір імен System.Security.Cryptography має криптографічні служби, включаючи безпечне кодування та декодування пакети, та ряд інших криптографічних функцій [11], що забезпечують надійне зберігання конфіденційних даних, наприклад, паролів. Іншим прикладом є класи з простору імен System.ComponentModel.DataAnnotations, які дозволяють проводити валідацію на стороні серверу, що допоможе додатково забезпечити вимоги обмеження домена.

Для реалізації шаблонів Repository та Service Layer створюються відповідні інтерфейси, що описують функціонал сервісів та репозиторіїв, та класи з конкретною реалізацією. Цього дозволяє досягнути об’єктно-орієнтована парадигма мови C#.

Задля забезпечення принципу інверсії управління та впровадження залежностей використовується бібліотека SimpleInjector, що має просту реалізацію з ретельно підібраним і повним набором функцій. Конфігурація міститься у коді, в окремому файлі, де визначені усі залежності, і виконується на початку роботи програми. Це дозволяє користувачу визначити конкретну реалізацію кожного із сервісів та використовувати в контролерах інтерфейси без прив’язки до імплементації.

Для зберігання та обробки даних була обрана реляційна СКБД SQL Server – продукт корпорації Microsoft, що використовує для запитів мову Transact-SQL, яка є реалізацією стандарту ANSI / ISO щодо мови SQL із певними розширеннями. Дана система підходить як і для невеликих баз даних, так і для масштабних баз даних підприємств, що буде найоптимальнішим рішенням при створенні ІС для будь-якого типу підприємств. SQL Server можна використовувати, встановивши на локальній машині або створивши сервер у хмарній платформі Microsoft Azure, яка забезпечує найкраще співвідношення між ціною та якістю серед конкурентів [12].

Для доступу до даних у програмному коді використовується ORM-фреймворк Entity Framework, що містить засоби для зв’язування БД з концепцією об'єктно-

орієнтованих мов програмування. Він забезпечує роботу з даними в термінах класів, а не реляцій, і навпаки, перетворює дані класів у дані, які можна зберігати в СКБД; також спрощуються CRUD-операції.

3.4 Опис розробки програми

C# – універсальна, мультипарадигмальна мова програмування, що включає в себе в тому числі і об'єктно-орієнтовану, засоби якої використовуються при розробці системи. Вся програма представляє собою множину об'єктів, які взаємодіють між собою, тому розробка класів є ключовим моментом для досягнення поставленої мети.

Інформаційна система розробляється із використанням шаблонів MVC, Service Layer та Repository, тому основні класи програми, відповідно, моделі даних, контролери, сервіси та репозиторії.

Найголовнішим компонентом системи є модель, яка відображає структуру даних застосунку і є незалежною від реалізації інтерфейсу. У даному застосунку в основному клас моделі визначає однойменну реляцію у базі даних, його поля – стовпці таблиці. В якості прикладу нижче наведена структура класу CourseCompletion, яка є ідентичною таблиці БД, що показує прогрес проходження користувачем системи навчального курсу.

```
public class CourseCompletion
    {
        public int id { get; set; }
        public int EmpId { get; set; }
        public int CourseId { get; set; }
        public int PercentCompleted { get; set; }
    }
```

Якщо поля класу ініціалізуються через складові інтерфейсу і потребують додаткової валідації (наприклад, на сторінці входу у систему або створення нового користувача), є

необхідним застосуванням спеціальних атрибутів, що забезпечать обмеження домену на стороні сервера та визначать, як поля відображатимуться на веб-сторінці. Класи простору імен System.ComponentModel.DataAnnotations містять атрибути для таких цілей. Нижче наведені деякі з них:

а) DisplayAttribute – клас, який задає назву, опис, порядок, підказку та інші атрибути відображення елементу;

б) RequiredAttribute – клас, який показує, що дане поле є обов'язковим для вводу, інакше виникає попереджувальне повідомлення;

в) DataTypeAttribute – клас, який надає засоби форматування та перевірки для різних типів даних полів вводу, наприклад, електронної пошти, паролю, валюти, номеру кредитної карти;

г) StringLengthAttribute – клас, що дозволяє задати мінімальну і максимальну довжину поля вводу.

Нижче наведений клас Employee, що відповідає за об'єкт співробітника та має кілька із вищезазначених атрибутів задля коректного заповнення бази даних.

```
public class Employee
    {
        public int id { get; set; }
        [Required]
        [Display(Name = "Last name")]
        public string LastName { get; set; }
        [Required]
        [Display(Name = "First name")]
        public string FirstName { get; set; }
        [Required]
        [Display(Name = "Patronymic")]
        public string Patronymic { get; set; }
        [Required]
        [DataType(DataType.PhoneNumber)]
        [Display(Name = "Telephone number")]
        public string TelNum { get; set; }
        [Required]
        [DataType(DataType.DateTime)]
        [Display(Name = "Birthday")]
        public DateTime Birthday { get; set; }
        [Required]
        [Display(Name = "Hiring date")]
        public DateTime HiringDate { get; set; }
        [Required]
        [Display(Name = "Houses per week")]
```

```
        public short HousesPerWeek { get; set; }
        public int PositionId { get; set; }
    }
```

Іншими класами-моделями, які не слугують представленнями реляцій бази даних, але грають важливу роль у процесі роботи програми, є наступні:

а) класи, що відповідають за обробку даних із представлень. Відрізняються від попередніх часом створення: спочатку створюється представлення, а потім клас для мапінгу, таблиці ж створюються на основі класи-моделей;

б) класи ViewModel для авторизації, які містять такі поля, як «Запам’ятати мене», «Підтвердити пароль» тощо, які не потребують збереження в базі. При успішній валідації їх дані перетворюються у дані програмного користувача.

Наступним важливим класом є контролер, який є сполучною ланкою між даними та їх представленням і відповідає за обробку. Фреймворк ASP.NET містить абстрактний клас Controller, що знаходиться у збірці System.Web.Mvc, який має бути базовим для всіх контролерів. Нижче наведено приклад контролеруEmployee , що відповідає за обробки запитів у підсистемі для звичайного співробітника.

```
[Authorize]
public class EmployeeController : Controller
   {
   // Your code goes here
   }
```

Даний клас містить ряд корисних полів та методів, наприклад, Request, що надає всю інформацію про поточний запит – URL, фізичний шлях, параметри, імя хосту, назву HTTP-методу; RouteData, що містить дані маршруту для поточного запиту, та інші. Контролери в ASP.NET MVC містять в собі спеціальні методи – Action, що відповідальні за виконання HTTP-запиту та генерування відповіді на нього. Усі публічні методи MVC-контролера – це методи Action; для створення допоміжного методу, який не буде обробляти запит, необхідно застосувати атрибут «NonAction». Методи контролеру можуть містити параметри, наприклад ID сутності, та призначені для обробки GET та

POST HTTP-запитів (позначаються атрибутами HttpGet та HttpPost). Всі методи за замовчуваннями є GET. Зазвичай GET-методи виконують запит для отримання даних зі спеціального ресурсу, а POST – для надання даних, що мають оброблятися, певному ресурсу, наприклад серверу.

Нижче міститься приклад GET-методу, що шукає співробітника за його ID та повертає веб-сторінку для зміни персональних даних.

```
public ActionResult EditPersonalProfile(int id)
        {
            var model = _employeeService.GetEmployee(id);
            return View(model);
        }
```

При зміні інформації та натисканні на кнопку підтвердження викликається POST-метод, який надсилає дані у базу та перенаправляє на сторінку персонального акаунту. При виникненні помилки виконується її перехоплення та обробка і користувач має ввести дані знову.

```
        [HttpPost]
        public ActionResult EditPersonalProfile(Employee model)
        {
            try
            {
                _employeeService.Update(model);
                return RedirectToAction("PersonalProfile");

            }
            catch (Exception e)
            {
                Debug.WriteLine(e);
                return RedirectToAction("EditPersonalProfile");
            }
         }
```

Для реалізації шаблонів Repository та Service Layer необхідно розробити класи з реалізацією методів, що відповідають функціональним вимогам. Для забезпечення механізму впровадження залежностей мають бути створені інтерфейси, які будуть реалізувати вищевказані класи. У даній роботі використовується механізм впровадження через конструктор – клас, якому необхідна залежність, містить відкритий

конструктор, який приймає інтерфейс сервісу (репозиторію) в якості параметру. Одним із приватних полів даного класу є зміна інтерфейсного типу, яка ініціалізується в конструкторі. Засоби бібліотеки Simple Injector дозволяють користувачу зробити співставлення інтерфейсу із потрібною реалізацією.

```
public class ServiceResolver
    {
        private static Container _container;

        public static Container Configure(Container container, Lifestyle lifestyle)
        {
            container.Register<IArchiveRepository, ArchiveRepository>(lifeStyle);
            //other repositories
            container.Register<IArchiveService, ArchiveService>(lifeStyle);
            //other services
            _container = container;
            return container;
        }
    }
```

Інтерфейс сервісу для обробки даних, пов’яза аступним чином, інші – створюються за аналогією.

```
public interface IProjectService
    {
        Project Get(int id);
        IEnumerable<Project> GetAllProjects();
        void CreateProject(Project project);
        void UpdateProject(Project project);
        void DeleteProject(int id);
    }
```

Реалізацією методів сервісів є виклик методів із відповідних репозиторіїв.

```
public Project Get(int id)
    {
        return _projectRepository.Get(id);
    }
```

Сервіси, що впроваджуються в контролер, далі використовуються в Action методі для обробки даних.

них з проектами, виглядає н

```
public class HrController : Controller
    {
        private readonly IProjectService _projectService;
```

```
        public HrController(IProjectService projectService)
        {
            _projectService = projectService;
         }
    }
```

Аналогічно створюються та впроваджуються у сервіси інтерфейси для репозиторіїв, проте їх реалізація в даному випадку відрізняється. Репозиторії наслідують від абстрактного класу Repository, що містить методи для обробки запитів та процедур та контекст бази даних – клас, який використовує заданий користувачем рядок підключення до конкретного серверу та набір типізованих колекцій для кожної сутності. Для обробки даних з БД є два підходи:

а) маніпулювання колекцією при виконанні простих запитів за допомогою методів розширення. Нижче наведені методи для доступу даних про робітника за його ID та для створення нового;

```
public class EmployeeRepository : Repository, IEmployeeRepository
    {
        public Employee GetEmployee(int id)
        {
            return Context.Employees.Find(id);
        }
        public void Create(Employee employee)
        {
            Context.Employees.Add(employee);
            SaveChanges();
        }
    }
```

б) при необхідності виконанні складного запиту (з групуванням, з’єднанням таблиць) краще створити окрему процедуру і забезпечити її виконання через виклик спеціального методу. Нижче наведений виклик процедури, яка повертає кількість годин, відзвітовану користувачем за заданий проміжок часу.

```
public double TimeSpentByUser(int empId, DateTime? from, DateTime? to)
   {
            return Query<double>("exec TimeSpentByUser @p0, @p1, @p2",
             empId, from, to).FirstOrDefault();
    }
```

Іншими важливими класами у програмі є BundleConfig, FilterConfig, RouteConfig, які відповідають за конфігурацію бандлів, фільтрів та маршрутів застосунку. Вони викликаються у методі Configure класу Startup при запуску роботи програми.

Іншими файлами застосунку є представлення – це файли з розширенням .cshtml, які представляють собою код користувацького інтерфейсу з використанням в основному мови розмітки HTML. Також вони містять елементи мови C#, початок яких виділяється символом «@» і які потрібні для зв'язку представлення з моделлю. При компіляції представлення спочатку перетворюється на клас, похідний від System.Web.Mvc.WebViewPage<dynamic>, який потім перетворюється у HTML-сторінку. Також у програмі містяться css та javascript файли задля додання стилю та динамічності до веб-сторінок.

Отже, ключовим моментом при розробці ООП застосунку – моделювання класів, що відповідають правилам бізнес-логіки, та взаємозв'язків між ними.

3.5 Створення об'єктів і розробка головної програми

При розробці об'єктно-орієнтованої системи важливим аспектом є не лише моделювання класів, а і правильне створення їх об'єктів, що не супроводжується генеруванням виключень та значними витратами з боку пам'яті. Після створення об'єкта має бути його виконана валідація, у випадку успіху в базу заноситься новий рядок з даними екземпляру та зберігається для подальшого користування.

Найголовнішими об'єктами інформаційної системи є користувачі, і для забезпечення надійності та безпеки застосунку має бути їх коректне генерування та надання вповноважень.

При введенні системи у експлуатацію в ній уже міститься користувач з роллю адміністратора та визначеним паролем, який можливо потім змінити. Він має виключне

право реєструвати будь-яких користувачів (наведено у додатку Б), змінювати їх належність до ролей (наведено у додатку В), редагувати, видаляти та змінювати пароль.

Для авторизації та аутентифікації використовується бібліотека AspNet.Identity.EntityFramework. В даному застосунку використовується cookies-аутентифікації, завдяки якій дані про поточну сесію зберігаються у браузері, тому при випадковому збої мережі або закритті браузера інформація про останній вхід не втрачається.

Для управління користувачами бібліотека використовує спеціальні класи, а для збереження даних автоматично створює таблиці AspNetRoles, AspNetUserClaims, AspNetUserLogins, AspNetUserRoles та AspNetUsers. Ключовим об’єктом є IdentityUser, який є мапінгом для сутності AspNetUsers, що вже містить ряд важливих полів, наприклад, хеш паролю, номер телефону, електрону пошту, але він може бути зв'язаний із іншою реалізацією класу користувача, виступаючи як базовий.

```
[DisplayColumn("EmployeeId")]
public class Employee : IdentityUser
   {

      //Custom properties...

   }
```

При реєстрації викликається метод CreateAsync класу UserManager, який містить в собі методи API та дозволяє обробляти всю інформацію, пов’язану із користувачами.

Стартовою сторінкою програми є сторінка входу. При вводі даних виконується POST-запит, який перевіряє, чи існує даний користувач у базі даних, і у випадку успіху перевіряє належність до певної ролі також методами класу UserManager. В залежності від ролі відбувається переадресація на стартову сторінку підсистеми для даної групи користувачів. Якщо користувач не має ролі, він перенаправляється знову на сторінку входу.

```
var user = await _userManager.FindAsync(model.UserName, model.Password);
if (user != null)
       {

              await SignInAsync(user, model.RememberMe);
              var roles = _userManager.GetRoles(user.Id);
              var enumerable = roles as string[] ?? roles.ToArray();
              if (enumerable.Contains(Enum.GetName(typeof(Roles), Roles.Admin)))
                return RedirectToAction("Index");
              if (enumerable.Contains(Enum.GetName(typeof(Roles), Roles.HrManager)))
                return RedirectToAction("EmployeesList", "Hr");
              if (enumerable.Contains(Enum.GetName(typeof(Roles), Roles.Employee)))
                return RedirectToAction("PersonalProfile", "Employee");
              if (enumerable.Contains(Enum.GetName(typeof(Roles), Roles.LearningManager)))
                return RedirectToAction("CoursesList", "LearningManager");
              return RedirectToLocal(returnUrl);

       }
```

Користувачі системи при роботі з нею неодноразово створюють інші об’єкти, а потім взаємодіють із ними, переглядаючи, редагуючи або видаляючи. Наприклад, звичайний співробітник, окрім можливості зміни персональних даних, може виконувати ще ряд дій, таких, як звітність за робочий час, проходження навчальних курсів, створення запитів на відпустку та інші.

Для переходу до трекеру часу необхідно натиснути на відповідний пункт меню, при генеруванні представлення для якого створюється об’єкт TimeTracker, що містить в якості зовнішнього ключа значення ID робітника.

```
var model = new TimeTracker { EmployeeId = id };
ViewBag.Projects = new SelectList(_projectService.GetAllProjects(), "Id", "Name");
return View(model);
```

Щоб відзвітувати час, треба вибрати назву проекту із випадаючого списку, вказати дату і кількість годин та натиснути на кнопку «Track», що викликає запит на створення об’єкту та занесення у базу.

```
public void TrackTime(TimeTracker timeTracker)
        {

            if (timeTracker == null) return;
            ExecuteNonQuery("insert into TimeTrackers (EmployeeId, ProjectId, [Date], TotalHours) "
+ $"values ({timeTracker.EmployeeId}, {timeTracker.ProjectId},
('{timeTracker.Date:yyyy-MM-dd HH:mm:ss.fff}'), {timeTracker.TotalHours})");
            SaveChanges();

        }
```

При створенні користувача автоматично викликається SQL-процедура, що зберігає дані про кількість днів відпусток та лікарняних для співробітників у таблиці RemainingDaysOff. І при створенні нового запиту для оплачуваного вихідного користувач має за аналогією перейти до відповідного пункту меню, при генеруванні якого створюється екземпляр класу DaysOff із зовнішнім ключем ID робітника. Користувач вводить тип вихідного, дату початку та кінця та кількість днів (необхідно у випадку, якщо проміжок відпустки припадає на вихідні дні). Перед занесенням даних об’єкта DaysOff у таблицю здійснюється перевірка на коректність дат, перевищення кількості дозволених днів, а при успішному виконанні операції оновлюються також дані реляції RemainingDaysOff.

```
public void UseDayOff(DaysOff model)

        {
            if (model == null) return;
            if ((model.EndDate - model.StartDate).Days < model.AmountOfDays) return;
            if (model.EndDate < model.StartDate) return;
            var vacay = RemainingDaysOff(model.EmployeeId);
            switch (model.Discriminator)
            {
                case "sick leave" when vacay.SickLeaveDays < model.AmountOfDays:
                    return;
                case "sick leave":
                    vacay.SickLeaveDays -= model.AmountOfDays;
                    ExecuteNonQuery("insert into DaysOffs
([EmployeeId],[Discriminator],[StartDate],[EndDate],[AmountOfDays],[Approved]) " +
                                    $"values({model.EmployeeId}, '{model.Discriminator}',
('{model.StartDate:yyyy-MM-dd HH:mm:ss.fff}'),  ('{model.EndDate:yyyy-MM-dd HH:mm:ss.fff}'),
{model.AmountOfDays}, {(model.Approved ? 1 : 0)})");
                    SaveChanges();
                    Context.Entry(vacay).State = EntityState.Modified;
                    SaveChanges();
                    break;
                case "vacation" when vacay.VacationDays < model.AmountOfDays:
                    return;
                case "vacation":
                    vacay.VacationDays -= model.AmountOfDays;
                    ExecuteNonQuery("insert into DaysOffs
([EmployeeId],[Discriminator],[StartDate],[EndDate],[AmountOfDays],[Approved]) " +
                                    $"values({model.EmployeeId}, '{model.Discriminator}',
('{model.StartDate:yyyy-MM-dd HH:mm:ss.fff}'),  ('{model.EndDate:yyyy-MM-dd HH:mm:ss.fff}'),
{model.AmountOfDays}, {(model.Approved ? 1 : 0)})");
                    SaveChanges();
                    Context.Entry(vacay).State = EntityState.Modified;
                    SaveChanges();
                    break;
              }

            }
```

Інші користувачі системи аналогічно під час усього процесу її користування створюють об’єкти, що зберігаються у базі, далі отримують їх та взаємодіють.

Отже, ключові аспекти в ООП застосунку – моделювання об’єктів, подальше правильне створення та взаємодія один з одним, що забезпечує безперебійну роботу системи та її відповідність вимогам предметної області.

3.6 Опис файлів даних та інтерфейсу програми

При створенні інформаційної системи необхідно розробити зручний та інтуїтивно зрозумілий користувацький інтерфейс, створити модель даних, яка найбільш точно відповідає бізнес-об’єктам, їх взаємовідношенням, та вибрати найбільш підходяще сховище даних. Також необхідно забезпечити зберігання файлів, які програма використовує як джерело вхідних даних чи кінцевих результатів.

В якості системи керування базами даних було обрано SQL Server, що має можливості для зберігання даних наступних видів: цілі числа, числа з плаваючою точкою, дата та час, символьні рядки, символьні рядки Unicode, двійкові дані, просторові дані, XML та інші, що може задовольнити майже будь-які користувацькі потреби.

При аналізі предметної області була розроблена модель, що складається з наступних реляцій (додаток Г):

а) AspNetUsers – найголовніша реляція, що відповідає за зберігання даних програмних користувачів. Складається з наступних полів:
   1) Id – PK, nvarchar(128) , NOT NULL;
   2) UserId – AK, int, identity, NOT NULL;
   3) UserName – nvarchar(256), NOT NULL;
   4) LastName – nvarchar(MAX), NOT NULL;

5) FirstName – nvarchar(MAX), NOT NULL;
6) Patronymic – nvarchar(MAX), NOT NULL;
7) Email – nvarchar(256), NULL;
8) Birthday – datetime, NOT NULL;
9) HiringDate – datetime, NOT NULL;
10) HousesPerWeek – smallint, NOT NULL;
11) PositionId – FK, int, NOT NULL;
12) EmailConfirmed – bit, NOT NULL;
13) PasswordHash – nvarchar(MAX), NULL;
14) SecurityStamp – nvarchar(MAX), NULL;
15) PhoneNumber – nvarchar(MAX), NULL;
16) PhoneNumberConfirmed – bit, NOT NULL;
17) TwoFactorEnabled – bit, NOT NULL;
18) LockoutEndDateUtc – datetime, NULL;
19) LockoutEnabled – bit, NOT NULL;
20) AccessFailedCount – int, NOT NULL;

б) AspNetRoles – таблиця, що відповідає за збереження ролей користувачів. Складається з наступних полів:
1) Id – nvarchar(128), NOT NULL;
2) Name – nvarchar(256), NOT NULL;

в) AspNetUserRoles – таблиця, що пов’язує ролі із користувачами. Складається з наступних полів:
1) UserId – PPK, FK, nvarchar(128), NOT NULL;
2) RoleId – PPK, FK, nvarchar(128), NOT NULL;

г) Archives – таблиця для зберігання файлів користувачів або навчальних курсів. Складається з наступних полів:
1) Id – PK, int, identity, NOT NULL;
2) Data – varbinary(MAX), NOT NULL;

3) OwnerId – FK, int, NULL;
4) Title – nvarchar(MAX), NOT NULL;

д) OnboardingCourses – містить інформацію про навчальний курс. Складається з наступних полів:
1) Id – PK, int, identity, NOT NULL;
2) Name – nvarchar(MAX), NOT NULL;
3) Description – nvarchar(MAX), NOT NULL;

е) CourseModules – містить інформацію про частини (модулі) навчального курсу. Складається з наступних полів:
1) Id – PK, int, identity, NOT NULL;
2) OnboardingCourseId – FK, int, NOT NULL;
3) Name – nvarchar(MAX), NOT NULL;
4) Description – nvarchar(MAX), NOT NULL;

ж) CourseCompletions – повязує користувача та його прогрес проходження курсу. Складається з наступних полів:
1) Id – PK, int, identity, NOT NULL;
2) EmployeeId – FK, int, NOT NULL;
3) OnboardingCourseId – FK, int, NOT NULL;
4) PercentCompleted – int, NOT NULL;

з) DaysOffs – слугує для обліку вихідних днів співробітників. Складається з наступних полів:
1) Id – PK, int, identity, NOT NULL;
2) EmployeeId – FK, int, NOT NULL;
3) Discriminator – nvarchar(MAX), NOT NULL;
4) StartDate – datetime, NOT NULL;
5) EndDate – datetime, NOT NULL;
6) AmountOfDays – smallint, NOT NULL;
7) Approved – bit, NOT NULL;

и) RemainingDaysOffs – таблиця, що показує кількість вихідних днів, які залишились у користувача. Складається з наступних полів:
   1) Id – PK, int, identity, NOT NULL;
   2) EmployeeId – FK, int, NOT NULL;
   3) VacationDays – smallint, default value – 20, NOT NULL;
   4) SickLeaveDays – smallint, default value – 10, NOT NULL;

к) Positions – містить в собі перелік існуючих вакансій компанії. Складається з наступних полів:
   1) Id – PK, int, identity, NOT NULL;
   2) Name – nvarchar(MAX), NOT NULL;
   3) Salary – decimal(18, 2), NOT NULL;

л) Projects – таблиця, що містить інформацію про поточні проекти. Складається з наступних полів:
   1) Id – PK, int, identity, NOT NULL;
   2) Name – nvarchar(MAX), NOT NULL;
   3) Description – nvarchar(MAX), NOT NULL;
   4) ClientName – nvarchar(MAX), NOT NULL;

м) RequiredSkills – таблиця, що містить перелік необхідних навичок для всіх вакансій. Складається з наступних полів:
   1) Id – PK, int, identity, NOT NULL;
   2) Name – nvarchar(MAX), NOT NULL;
   3) Description – nvarchar(MAX), NOT NULL;
   4) MinReqYears – float, NOT NULL;
   5) MaxReqYears – float, NOT NULL;
   6) PositionId – FK, int, NOT NULL;

н) Skills – таблиця, що містить перелік навичок співробітників. Складається з наступних полів:
   1) Id – PK, int, identity, NOT NULL;

2) Name – nvarchar(MAX), NOT NULL;
3) Description – nvarchar(MAX), NOT NULL;
4) Years – float, NOT NULL;
5) EmployeeId – FK, int, NOT NULL;

o) TimeTrackers – таблиця, що містить записи про відзвітований час кожного користувача. Складається з наступних полів:

1) Id – PK, int, identity, NOT NULL;
2) TotalHours – float, NOT NULL;
3) Date – datetime, NOT NULL;
4) PositionId – FK, int, NOT NULL;
5) EmployeeId – FK, int, NOT NULL.

Обмеження на типи атрибутів, допускання NULL значень або значень за замовчуванням та визначення взаємозв'язків між таблицями дає можливість забезпечити вимоги бізнес логіки та цілісність даних.

При авторизації створюються об'єкти користувачів та заносяться у таблицю БД, частина даних якої має наступний вигляд.

LastName	FirstName	Patronymic	Email	Birthday	Hiring...	Houses...	P...	Id	UserId	EmailConf...
Admin	Global	Admin	globaladmin@domai...	2020-05-0...	2020-...	40	1	01	2	0
Main	Learning	Manager	lms@domain.com	1990-10-0...	2020-...	40	3	0bca68ae-350f-49e4-90...	4	0
Ivanov	Ivan	Ivanovych	i.ivanov@domain.com	1999-11-1...	2020-...	40	4	12fbbc33-73d3-433b-aa...	5	0
Main	HR	manager	hr@domain.com	1990-10-0...	2020-...	40	2	e276b5ff-5527-4724-8b...	3	0

Співставлення даних користувача із роллю відбувається наступним чином.

UserId	RoleId
01	28f009e3-6db4-442c-bc16-68f48f97a75b
0bca68ae-350f-49e4-9027-e01bb7352697	2caa0ea2-94dd-4153-86ec-1cf91ed2abf1
12fbbc33-73d3-433b-aad5-aaf49f96980b	b6489fe2-26b7-449e-bb5f-462719c84731
e276b5ff-5527-4724-8bee-8f827ec52660	b6eedf49-5431-41d8-b4b8-e540906d9db3

Оскільки дана СКБД дозволяє зберігати дані не лише простих типів, у базі даних можна заносити файли різних форматів – у бінарному вигляді. Реляція Archives створена

для зберігання файлів та містить інформацію про них та самий набір байтів. Наприклад, при створенні/редагуванні профілю користувача часто є необхідним додати фотографію користувача (додаток Д). Для цього необхідно вибрати необхідний файл та вивантажити на сервер. Це відбувається завдяки виклику методу ReadBytes класу BinaryReader, що зчитує байти з потоку об'єкту класу HttpPostedFileBase.

Отримані дані будуть виглядати наступним чином.

Id	Data	OwnerId	Title
8	0x89504E470D0A1A0A00...	5	Image photo

При оновленні сторінки виконається запит на отримання даних про фотографію профілю даного користувача і файл відобразиться після виклику Action методу.

```
public ActionResult Display(int id)
   {
            var cover = _archiveService.GetArchive(id)?.Data;
            return cover != null ? File(cover, "image/jpg") : null;

   }
```

У роботі системи часто виникає необхідність об'єднувати та фільтрувати таблиці для виведення вичерпних і зрозумілих для звичайного користувача даних. Щоб зменшити довжину запиту, можна створювати спеціальні віртуальні таблиці – представлення, що динамічно оновлюються при кожному виконанні. Прикладом такого представлення може слугувати EmpDayOffs, яке показує ПІБ співробітника та його запит на оплачуваний вихідний. Результат відображається у підсистемі для HR-менеджерів, які матимуть змогу схвалити заявку або відхилити. Представлення формується наступним чином.

```
CREATE VIEW EmpDayOffs AS
SELECT dbo.DaysOffs.Id, dbo.DaysOffs.EmployeeId, dbo.DaysOffs.Discriminator, dbo.DaysOffs.StartDate,
dbo.DaysOffs.EndDate, dbo.DaysOffs.AmountOfDays, dbo.DaysOffs.Approved,
dbo.AspNetUsers.LastName, dbo.AspNetUsers.FirstName, dbo.AspNetUsers.Patronymic
FROM dbo.DaysOffs INNER JOIN
                    dbo.AspNetUsers ON dbo.DaysOffs.EmployeeId = dbo.AspNetUsers.UserId
```

Також у системі міститься представлення CourseAssignments, яке повертає ID співробітника та інформацію про призначений йому курс.

Іншим засобом баз даних, який покращує роботу системи, є процедури – об'єкти, що складаються з SQL-інструкцій та зберігаються на сервері. Мають вхідні та вихідні дані, локальні змінні і мають велику схожість на звичайні процедури мов програмування. Вони дозволяють збільшити продуктивність роботи програми, зберігаючи в собі код для часто повторюваних запитів, забезпечують безпеку даних. В даній системі міститься процедура GetAssignedCourses для формування списку призначених навчальних курсів для користувача, процедура RemainingDays, що викликається при створенні нового користувача та заносить дані про кількість відпускних днів у таблицю RemainingDaysOffs. Для обчислення робочого часу на кожен проект всіма користувачами та конкретним за певний проміжок часу виконуються процедури TimeSpentForProject та TimeSpentByUser. Вони схожі між собою та мають наступну структуру.

```
CREATE PROCEDURE [dbo].[TimeSpentByUser]
	@empId int,
	@from datetime2(7) = null,
	@to datetime2(7) = null
AS
BEGIN
	SET NOCOUNT ON;
	SELECT SUM(TotalHours)
	FROM TimeTrackers
	WHERE EmployeeId = @empId and (@from is null or Date >= @from) and
	 (@to is null or Date <= @to)
	GROUP BY EmployeeId
END
```

База даних створюється на основі існуючих класів і будь-яка зміна в їх структурі передбачає зміну моделі даних. Перетворення даних таблиць в екземпляри класів виконуються за допомогою засобів фреймворку EntityFramework (див. 3.3).

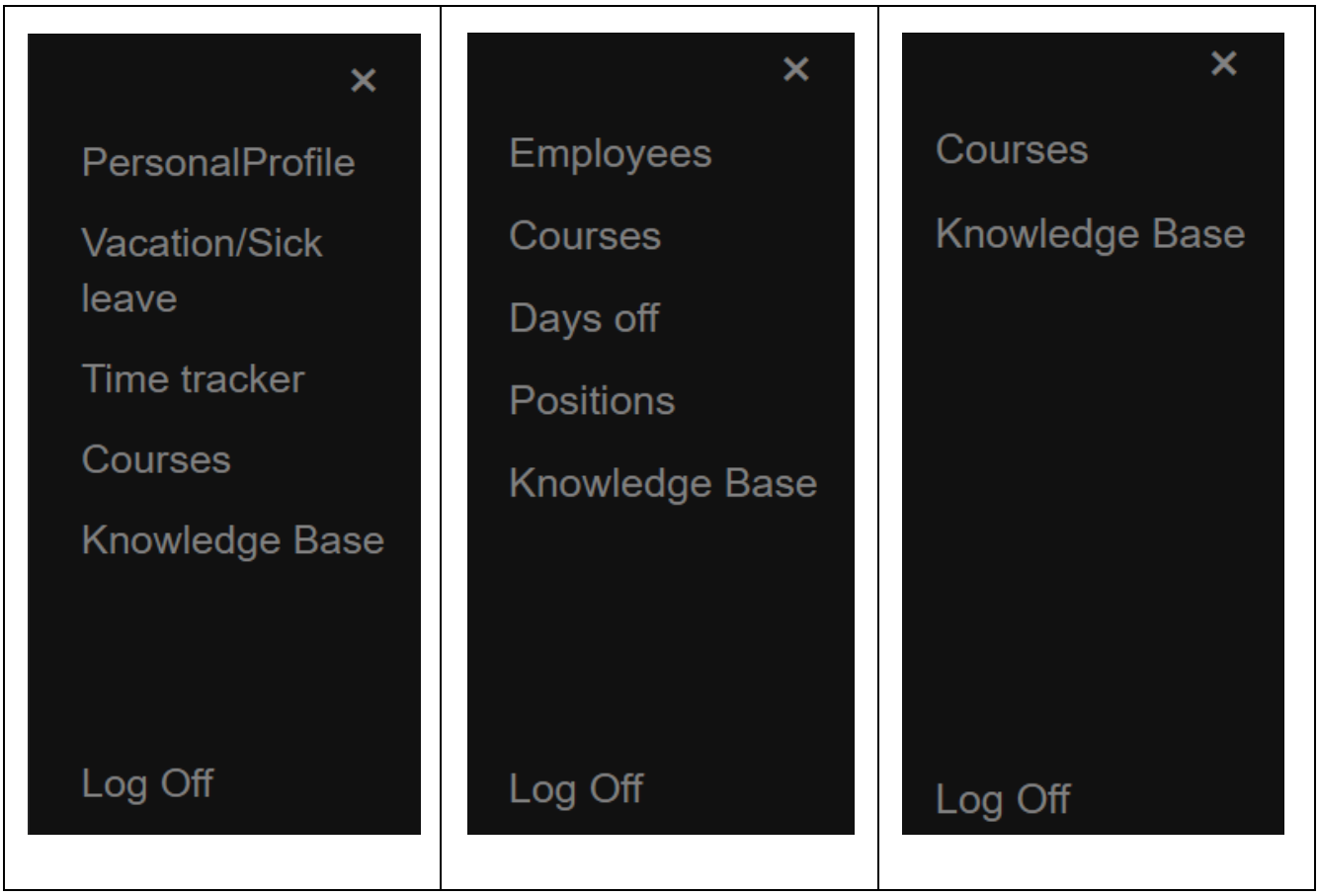
Іншим компонентом застосунку є база знань, що містить засоби для зберігання, пошуку, маніпуляції та видачі знань і забезпечує зручне їх представлення. В даній інформаційній системі база знань зберігає дані про організацію, наприклад, перелік

проектів, вакансій, співробітників та нормативних документів. Це допомагає новим співробітникам знайти спосіб вирішення будь-якої проблеми, що пов’язана з робочим процесом, та прискорити процес адаптації. Сторінка БЗ містить зручну навігацію та пошук, що дозволяє знайти інформацію у будь-якій категорії, а запити, які виконуються під час цього процесу, звертаються та змінюють дані в БД організації.

Кожен користувач системи має перелік дозволених дій (див. 3.2), і для зручної навігації необхідно створити меню, яке дозволить виконувати переходи на них. Пункти меню мають містити посилання лише на дозволені сторінки та сторінки загального користування (посилання на базу знань та на вихід із системи, що переадресовує на сторінку Login). Різницю між зовнішнім виглядом меню наведено у таблиці 3.2.

Таблиця 3.2 – Навігаційні меню підсистем

Підсистеми		
Employee	HrManager	LearningManager



Отже, після створення застосунку із надійним зберіганням даних та зручним інтерфейсом, що відповідає усім вимогам предметної області та використовує найбільш підходящі програмні засоби, необхідно провести його тестування. База має бути заповнена тестовими даними у достатній кількості, щоб перевірити усі можливі випадки при роботі функціоналу та навантаження на систему.

3.7 Тестування програми і результати її виконання

Перед вводом у експлуатацію системи, що використовує бази даних та знань, в першу чергу необхідно заповнити її пробними даними різних типів та провести її переконливе тестування із перевіркою усіх сценаріїв системи.

Дана інформаційна система містить механізм автентифікації на основі ролей, тому необхідно перевірити можливі варіанти входу для кожного користувача, можливість несанкціонованого доступу в іншу систему за допомогою вводу URL та дії системи при введені неправильних даних. Стартовою сторінкою системи є сторінка входу (додаток Е), на якій користувач має ввести свій логін та пароль. При натисканні на кнопку «Log in» з незаповненими полями або з неправильними даними відображаються попереджувальні повідомлення і користувач має ввести дані ще раз.

Користувач, що має виключні права у системі – адміністратор, який при переході на основну сторінку бачить список усіх зареєстрованих користувачів (див. рисунок В.1). Він може виконати щодо кожного користувача наступні дії:

а) змінити основну інформацію;
б) змінити належність користувача до ролі;
в) видалити із системи
г) ввійти у систему замість нього.

Також адміністратор може створити нового користувача, перейшовши на сторінку реєстрації (див. додаток Б), та керувати базою знань, до якої містять доступ всі інші користувачі. Учасники ролі LearningManager також можуть доповнювати БЗ, всі інші мають лише право на перегляд.

Для звичайного користувача база знань виглядає як сторінка із переліком інформації певної категорії, навігаційним меню та формою пошуку (додаток Є). Вона містить перелік проектів, користувачів, вакансій та документів. При бажанні користувача знайти будь-які знання за ключовим словом він може ввести його у форму пошуку та отримати бажаний результат (додаток Ж). Якщо необхідно вийти із системи, користувач має натиснути на знак у правому верхньому куті.

Якщо поточний користувач – HR-менеджер, стартовою сторінкою буде сторінка, показана на рисунку З.1. Менеджер бачить всіх зареєстрованих користувачів, окрім адміністраторів, та може додавати нових користувачів, змінювати інформацію про них, видаляти та призначати навчання. Також є можливість оброблювати заявки на оплачувані вихідні (рисунок З.2), переглядати курси та змінювати інформацію про поточні вакансії, включаючи ЗП та необхідні навики.

Для звичайного користувача стартова сторінка це персональний профіль (рисунок І.1), який він може редагувати та змінити його фотографію. Також користувач може переглянути баланс відпусток та лікарняних (рисунок І.2), попередні заявки (рисунок І.3) та створити нову (рисунок І.4). Іншим функціоналом є ведення звітності – користувач може переглянути попередні записи та створити новий (додаток Ї) та проходження навчальних курсів.

Менеджер з навчання персоналу може окрім управління базою знань створювати та редагувати навчальні курси (рисунок Й.1). Також є можливість переглядати та оновлювати список модулів для кожного курсу, завантажувати матеріали до них (рисунок Й.2).

У випадку спроби доступу до ресурсів іншої підсистеми через ввід URL користувач буде переадресований на сторінку входу і при вводі правильних даний зможе одразу на неї перейти. На додатку К показано сценарій спроби переходу з ролі LearningManager на ресурс ролі Employee.

Отже, розроблена система реалізує всі зазначені вимоги, проходить тестування при виконанні різних сценаріїв, відповідає задачі предметної області та розроблена за допомогою програмних засобів, які забезпечують її надійність, швидкодію та можливість до масштабування.

Висновки

Мета даної курсової роботи полягала у розробці інформаційної системи керування персоналом із застосуванням баз даних і знань. У ході роботи був проведений аналіз існуючих аналогів, визначені їх недоліки та типові проблеми та можливий функціонал. З урахуванням всіх нюансів ситуації на ринку була розроблена структура програми та визначені методи для її реалізації.

Після визначення технічних вимог, архітектури та алгоритму роботи програми, було проведено моделювання основних її об’єктів, обрані одні із найоптимальніших засобів розробки та зберігання даних, що в кінцевому результаті привело до створення зручного веб-застосунку із легко масштабованою базою даних для даної інформаційною системи для користувачів з різними правами доступу та функціоналом.

Дана система, незважаючи на відповідність зазначеним вимогам, має безліч способів для вдосконалення. Є можливим додавати інші групи користувачі для централізації даних цілого підприємства в єдиній системи та зручній взаємодії її користувачів. Можливо додати фінансовий відділ для нарахування заробітної плати та пов’язати його із платіжною системою чи банком. Корисною функцією буде публікація вакансій та обробка відгуків прямо з системи, що дозволить зберігати інформацію про кар’єрний шлях робітника з моменту першої співбесіди. Адміністратор, крім вже існуючих дій, може також активувати чи дезактивувати певний функціонал або цілі системи в залежності від потреб компанії. Для великої мережі компаній можливо додати адміністраторів різних рівнів, які матимуть доступ до свого філіалу або цілої системи.

Список літератури

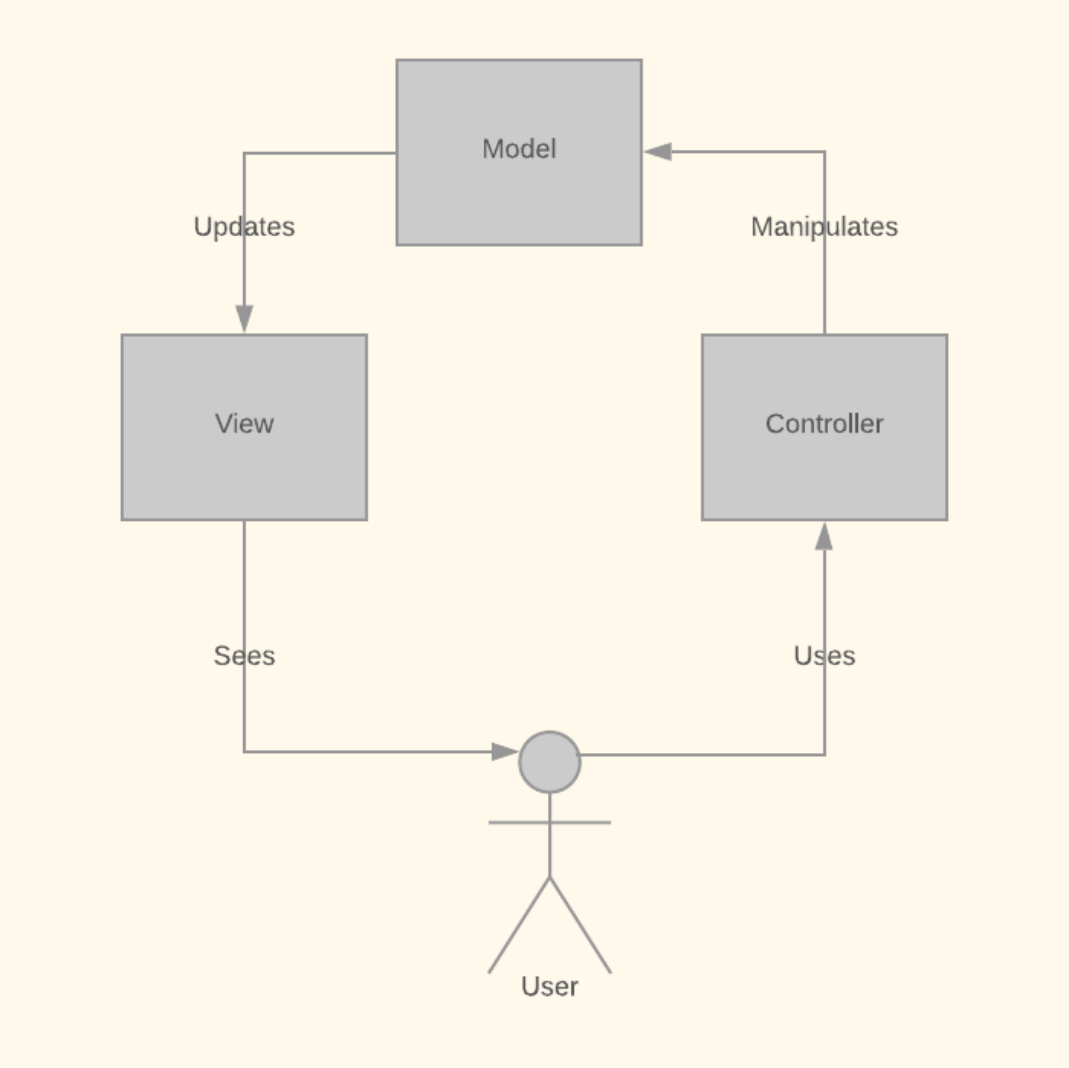
1. Kipkemboi Jacob Rotich. HISTORY, EVOLUTION AND DEVELOPMENT OF HUMAN RESOURCE MANAGEMENT: A CONTEMPORARY PERSPECTIVE [Електронний ресурс] : стаття з журналу Global Journal of Human Resource Management, 2015, с.58
2. Human resource management : a contemporary approach Julie Beardwell, Tim Claydon Published in 2010 in New York NY by Pearson Financial Times/Prentice Hall
3. Організація промислового виробництва. Основні поняття. Терміни та визначення: ДСТУ 2960-94. – [Чинний від 1996-01-01. – К.: Державний інститут праці та соціально-економічних досліджень, 1994. – VII, 15с. – (Державний стандарт України).
4. BambooHR Packaging [Електронний ресурс] – Режим доступу до ресурсу: https://www.bamboohr.com/packaging/
5. FingerCheck About [Електронний ресурс] – Режим доступу до ресурсу: https://fingercheck.com/about/
6. UltiPro Solution Features [Електронний ресурс] – Режим доступу до ресурсу: https://www.ultimatesoftware.com/UltiPro-Solution-Features
7. Management Information System Text and Cases A Digital-Firm Perscpective, fourth edition, Nanda, V.K, Anmol Publications Pvt. Limited
8. Management Information System, Hitesh Gupta, HITESH GUPTA, 2011, с. 15
9. Комп'ютерні мережі: [навчальний посібник] / А. Г. Микитишин, М. М. Митник, П. Д. Стухляк, В. В. Пасічник. — Львів: «Магнолія 2006», 2013. — 256 с.
10. Understanding of the management information system based on MVC pattern, Sida Chen, AIP Conference Proceedings(2018) [Електронний ресурс] – Режим доступу до ресурсу: https://doi.org/10.1063/1.5033678

11. MSDN [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-gb/dotnet/api/system.security.cryptography?view=dotnet-plat-ext-3.1
12. Azure Portal [Електронний ресурс] – Режим доступу до ресурсу: https://azure.microsoft.com/mediahandler/files/resourcefiles/sql-transactional-processing-price-performance-testing/GigaOm%20Azure%20SQL%20DB-AWS%20RDS%20Price-Performance%20Benchmark%20Report.pdf

Додаток А

(обов’язковий)

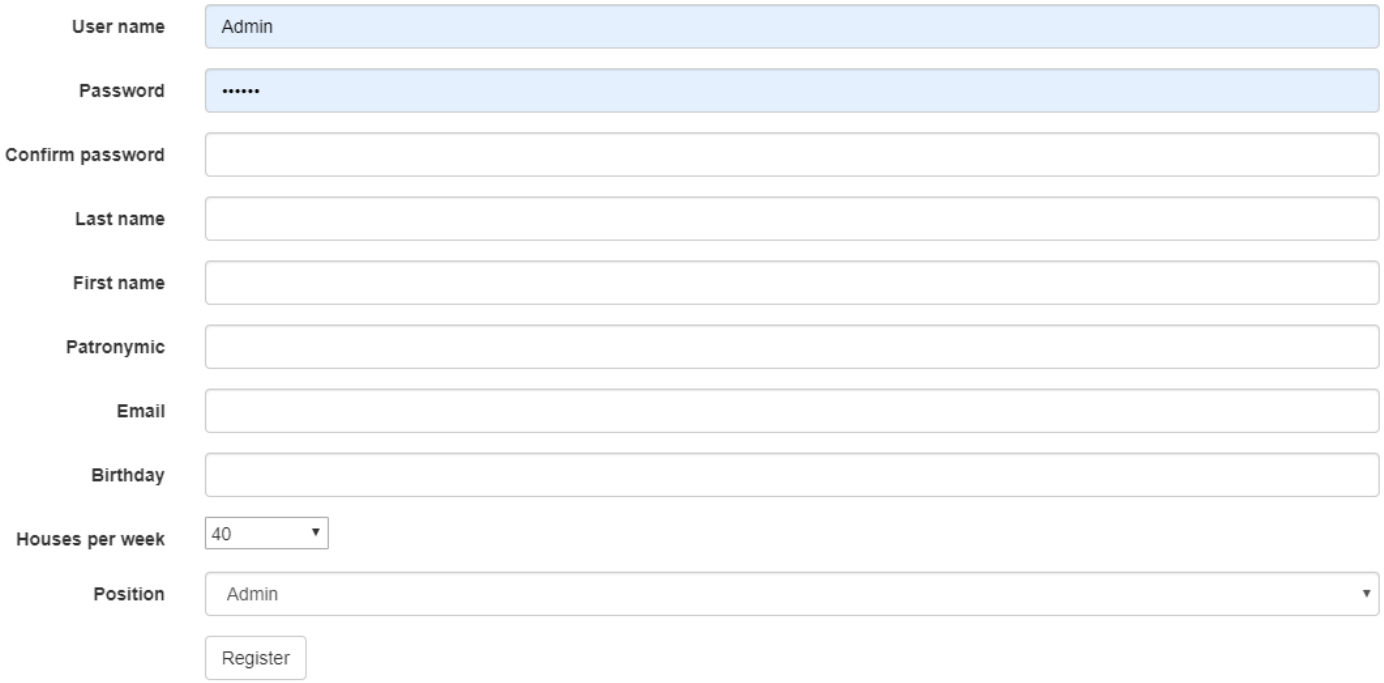
Схема паттерну MVC



Додаток Б

(обов’язковий)

Реєстрація користувача адміністратором



Додаток В

(обов'язковий)

Управління ролями користувача

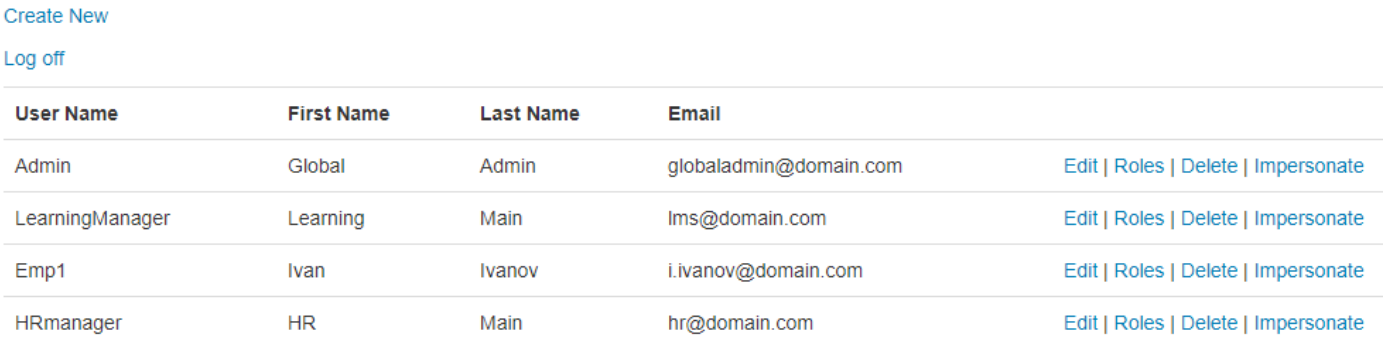


Рисунок В.1 – Відображення створених користувачів та дій над ними

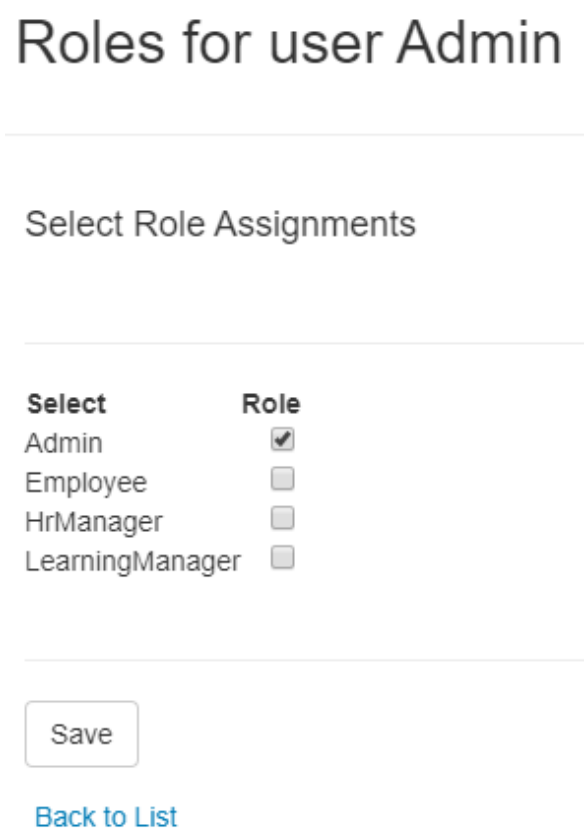
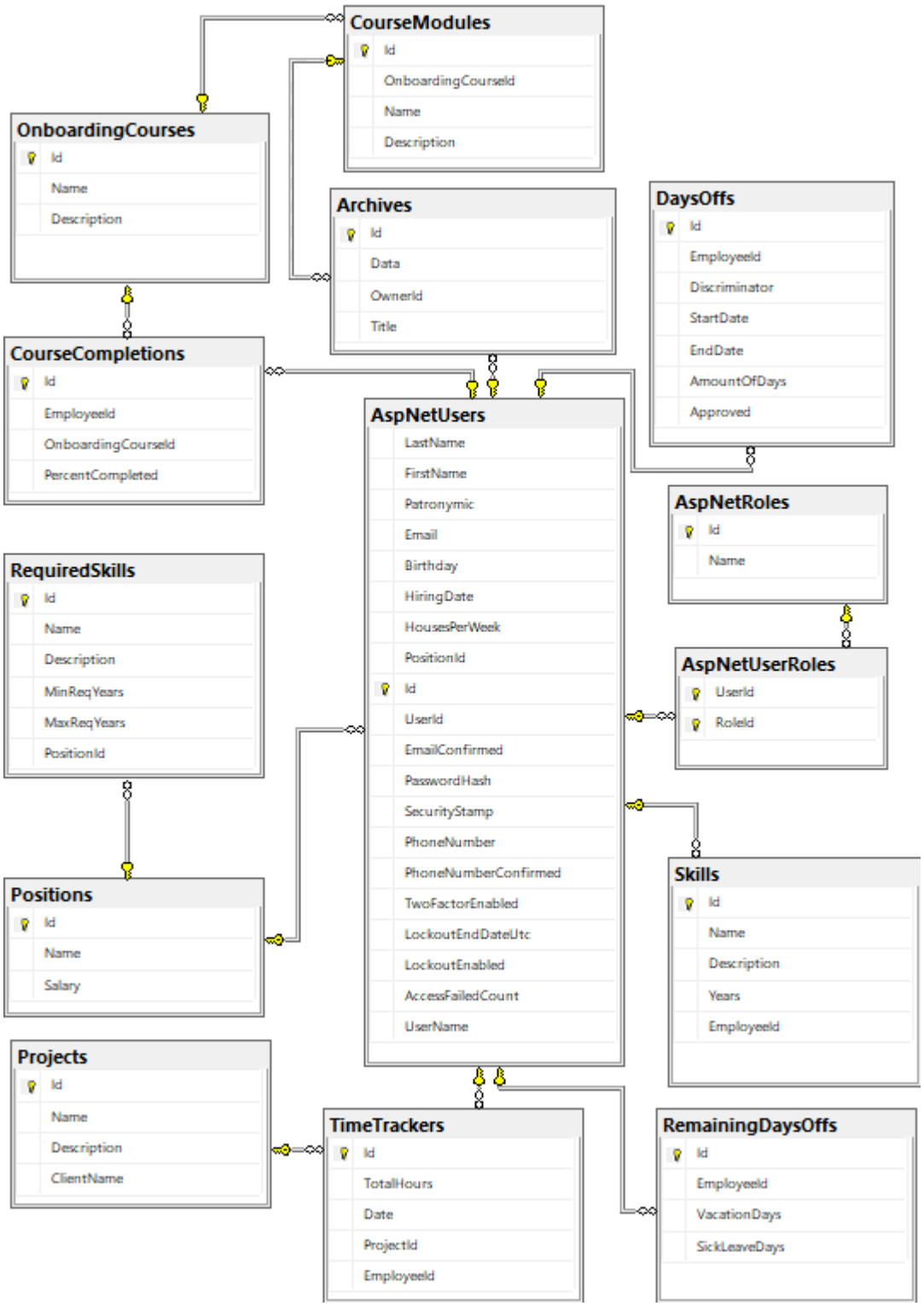


Рисунок В.2 – Вибір ролей для користувача

Додаток Г

(обов'язковий)

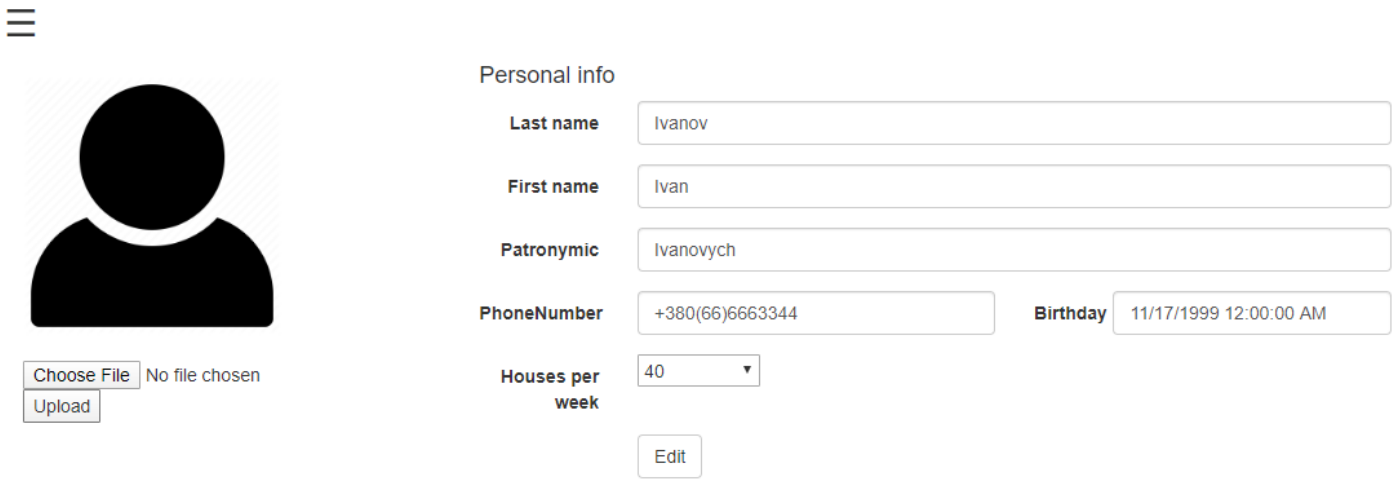
Реляційна модель інформаційної системи



Додаток Д

(обов’язковий)

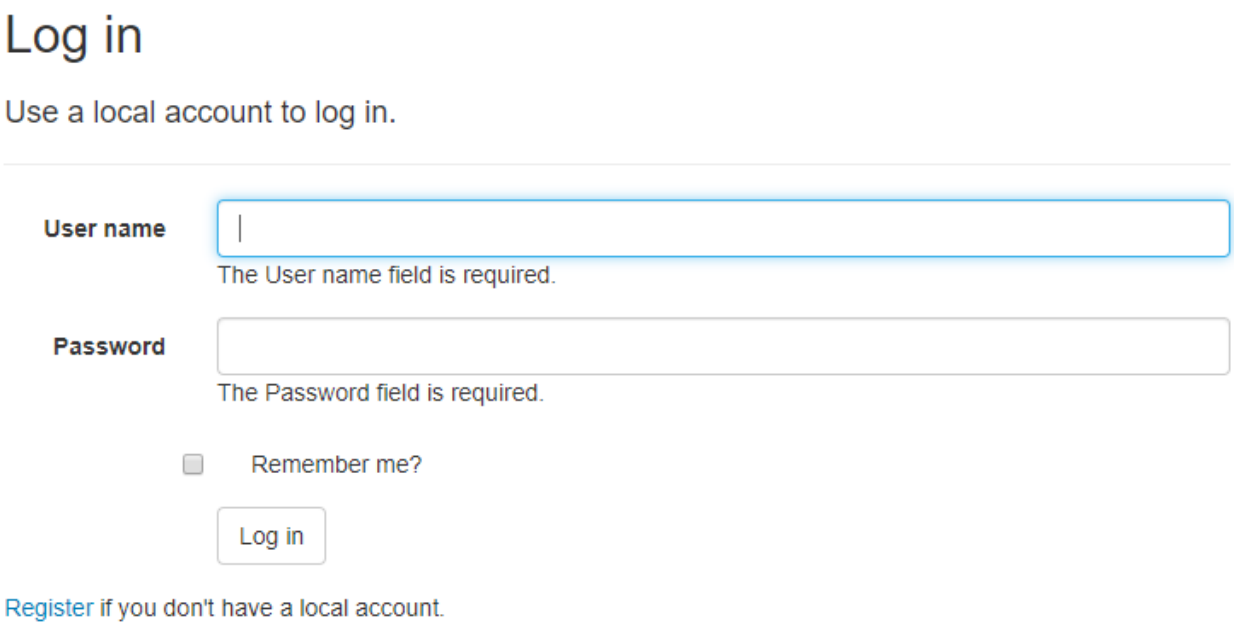
Сторінка редагування користувача



Додаток Е

(обов’язковий)

Сторінка входу



Додаток Є

(обов'язковий)

Інтерфейс бази знань

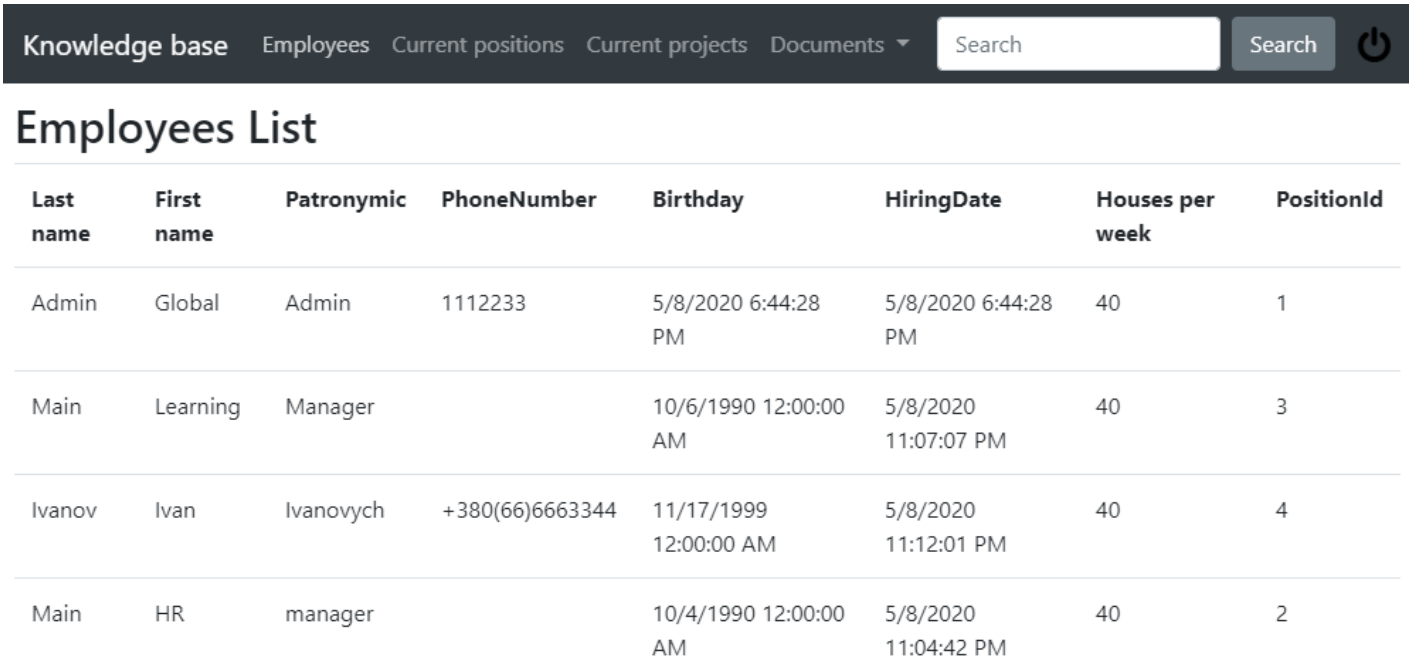


Рисунок Є.1 – Список користувачів системи

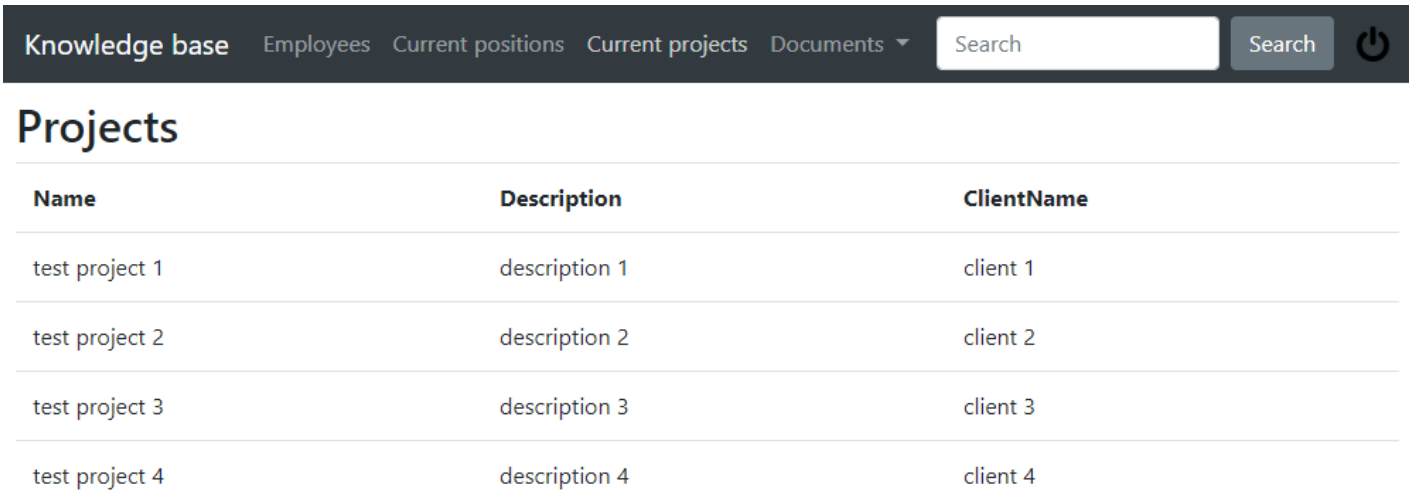
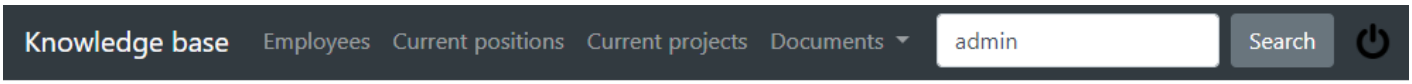


Рисунок Є.2 – Список поточних проектів

Додаток Ж

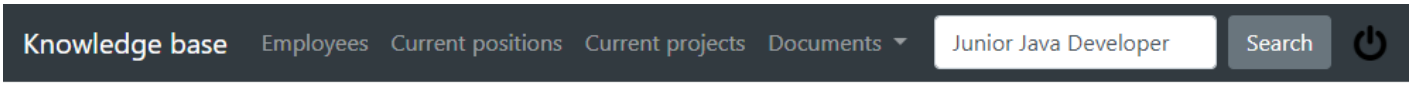
(обов'язковий)

Результати пошуку у базі знань



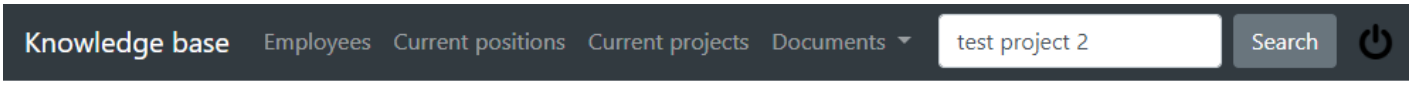
Employees List

Last name	First name	Patronymic	PhoneNumber	Birthday	HiringDate	Houses per week	PositionId
Admin	Global	Admin	1112233	5/8/2020 6:44:28 PM	5/8/2020 6:44:28 PM	40	1



Positions

Name	Salary
Junior Java Developer	$700.00



Projects

Name	Description	ClientName
test project 2	description 2	client 2

Додаток З

(обов'язковий)

Інтерфейс підсистеми для HR-менеджера

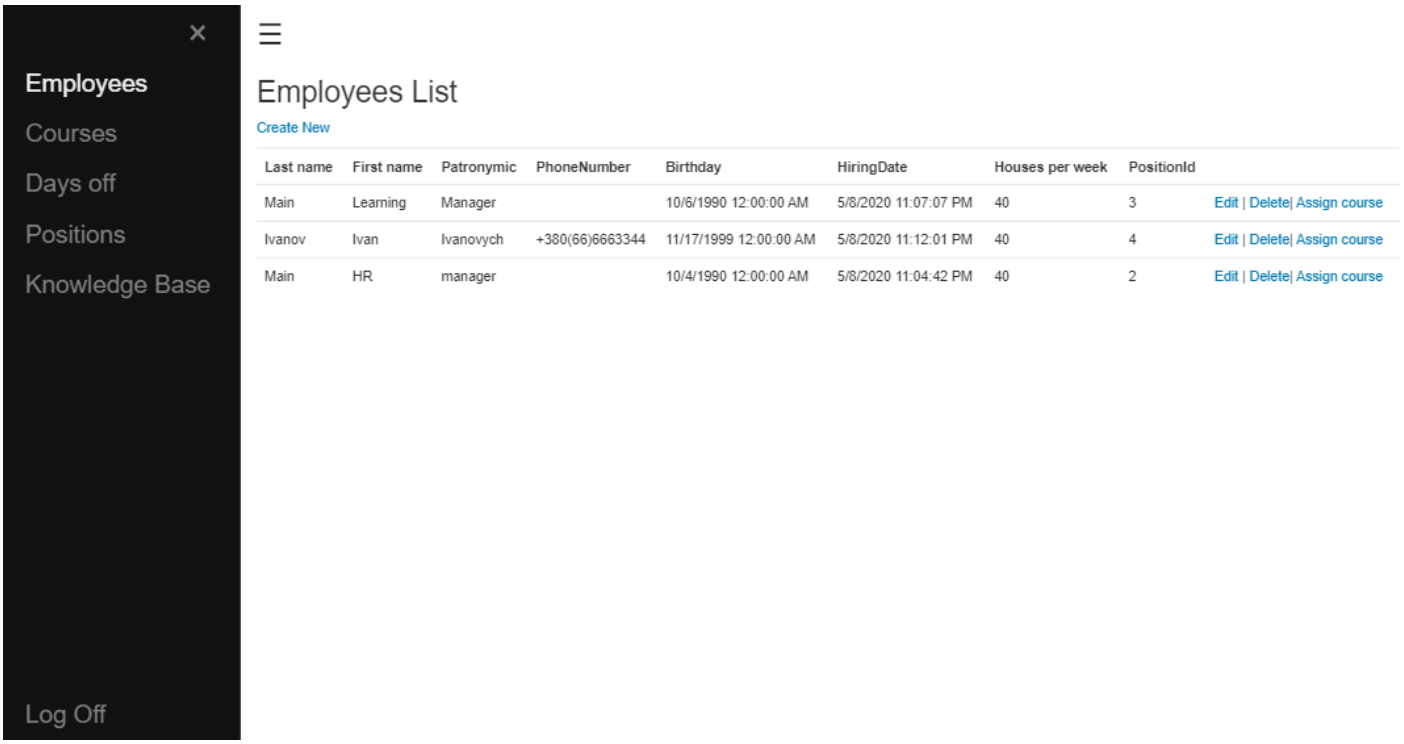


Рисунок 3.1 – Стартова сторінка

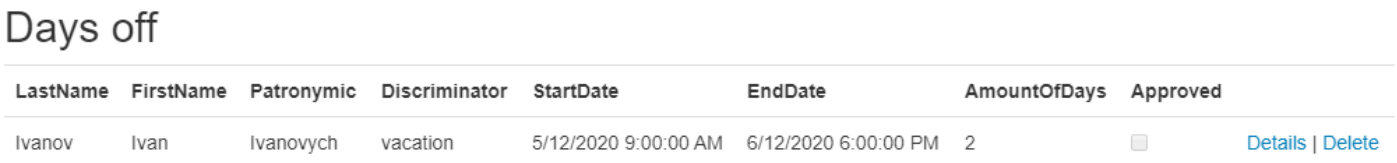


Рисунок 3.2 – Обробка заяв на відпустки та лікарняні

Додаток І

(обов'язковий)

Інтерфейс звичайного співробітника

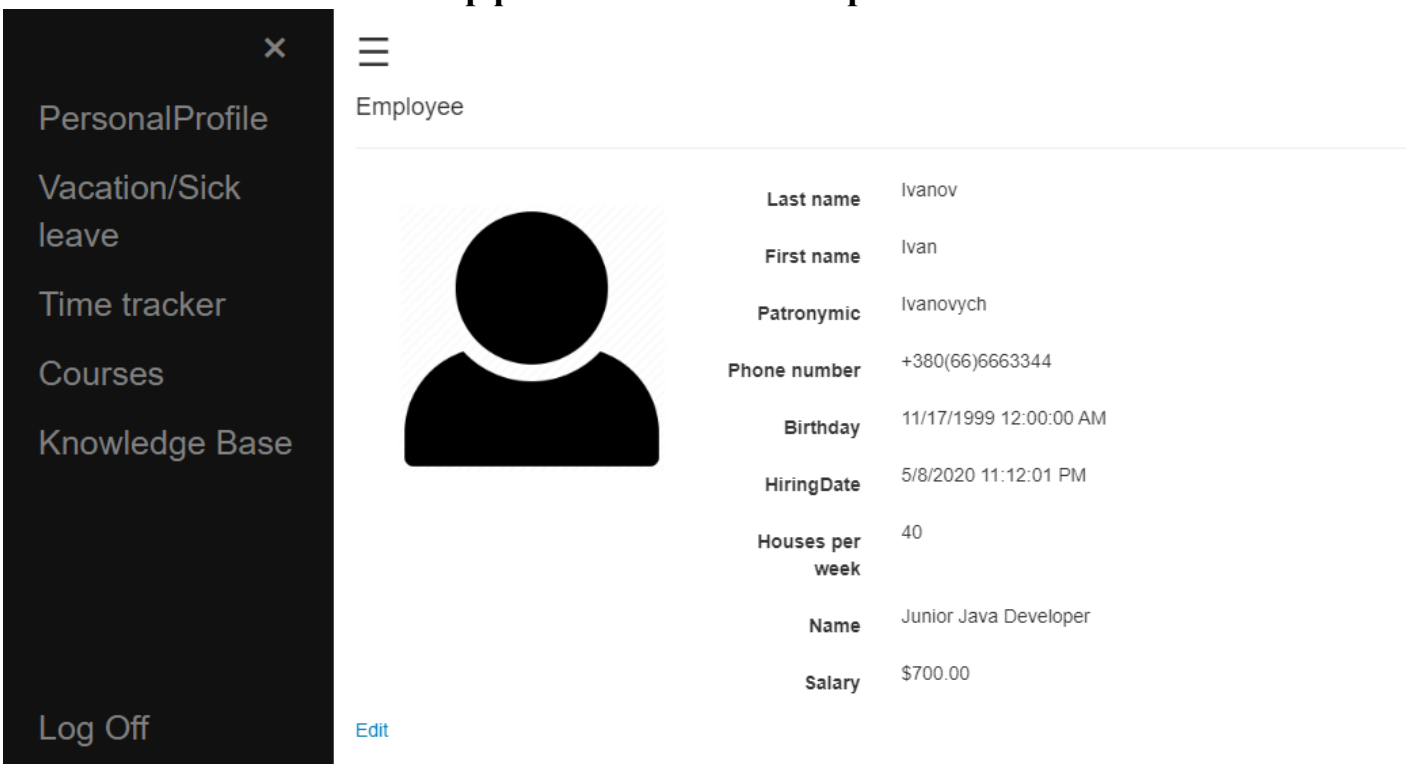


Рисунок І.1 – Стартова сторінка

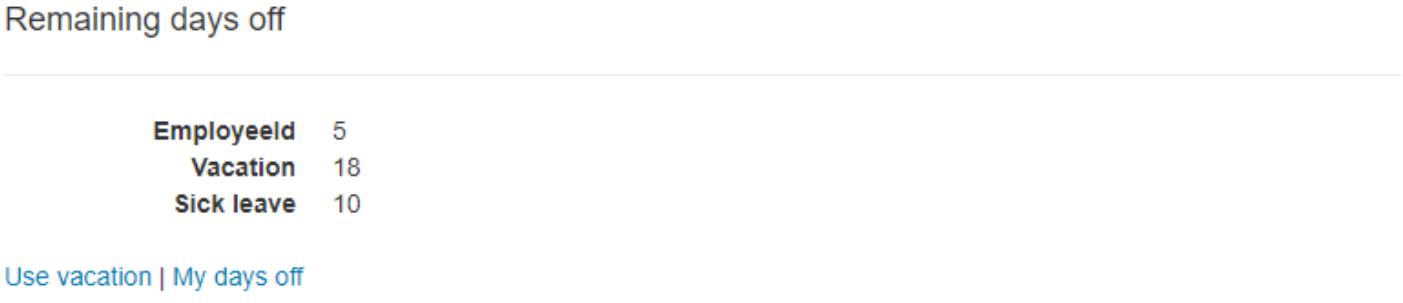


Рисунок І.2 – Баланс вихідних днів

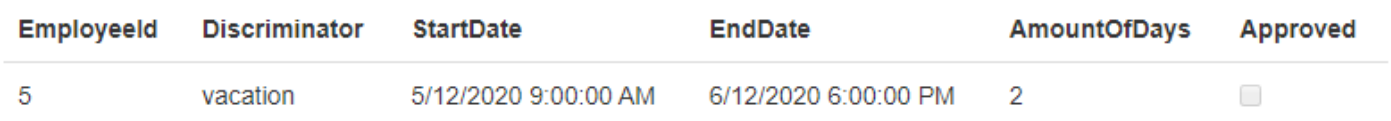


Рисунок І.3 – Перегляд власних заявок

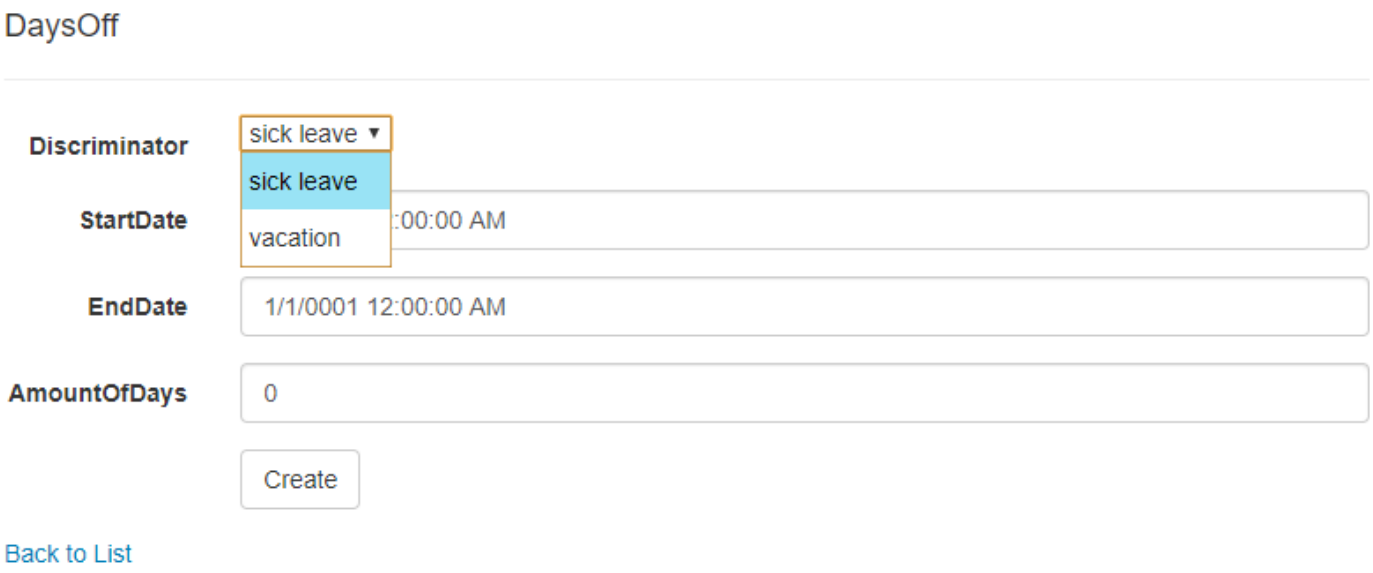
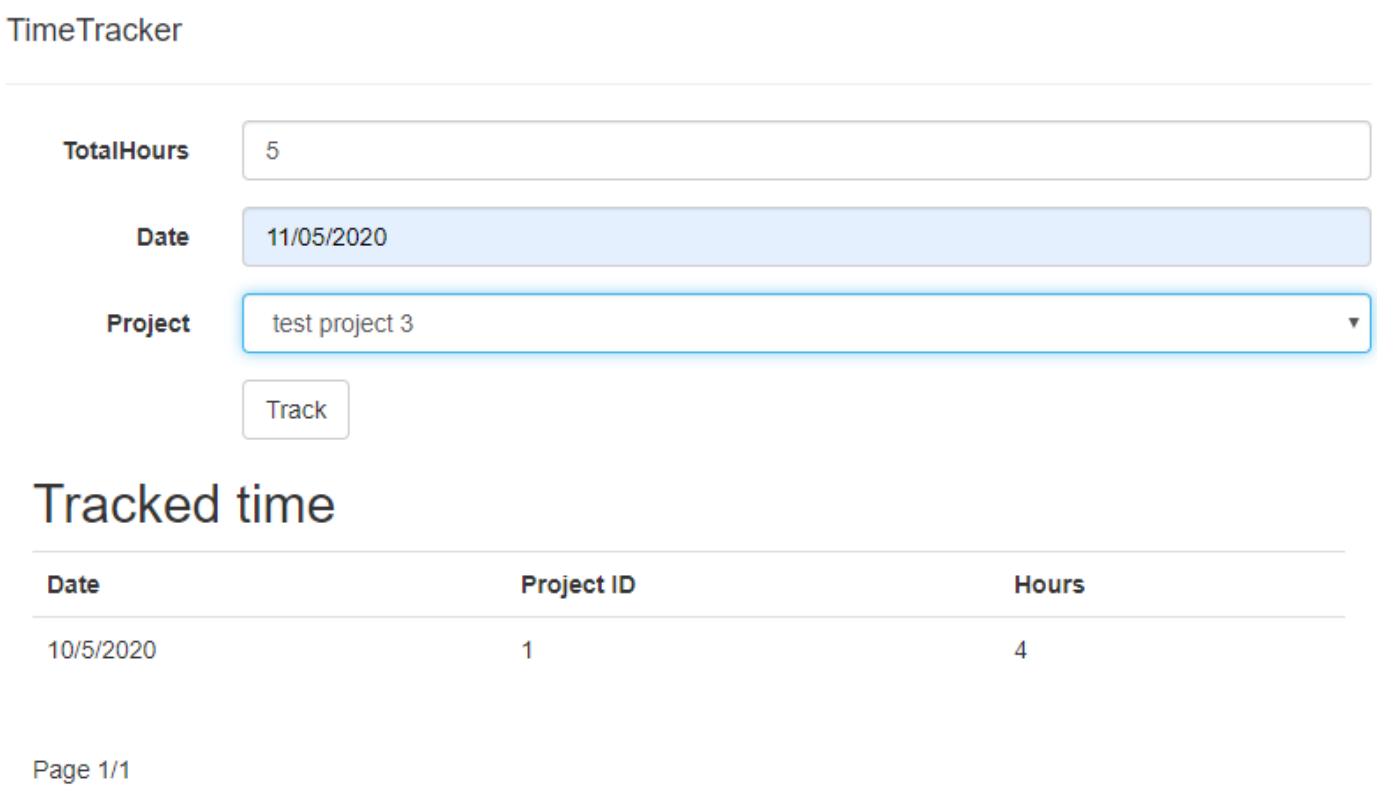


Рисунок І.4 – Створення нової заявки

Додаток Ї

(обов'язковий)

Ведення звітності



Додаток Й

(обов'язковий)

Інтерфейс менеджеру з навчання персоналу

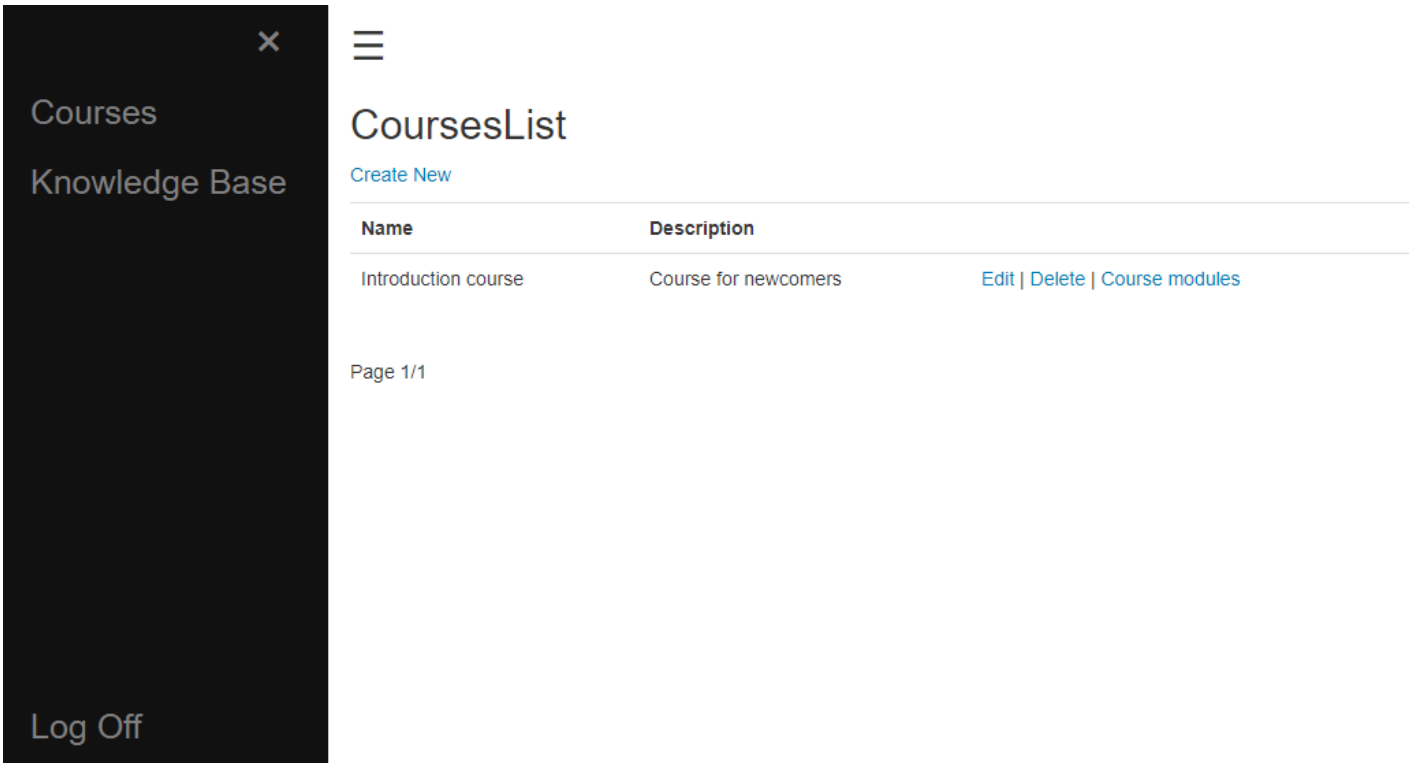


Рисунок Й.1 – Стартова сторінка

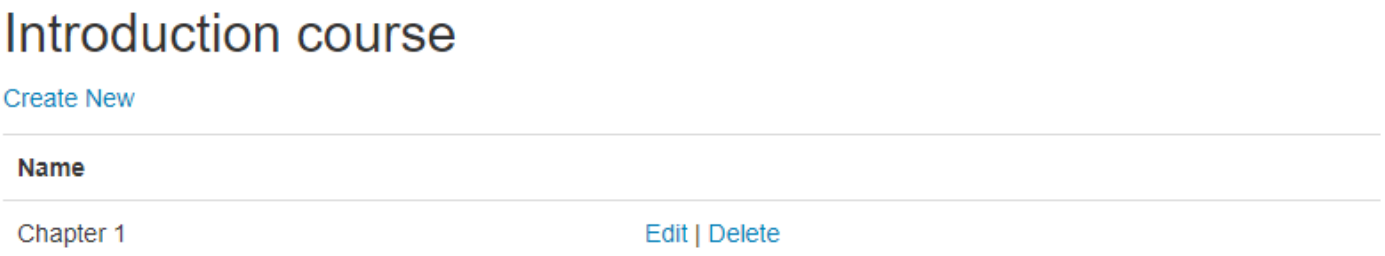


Рисунок Й.2 – Перегляд модулів курсу

Додаток К

(обов'язковий)

Несанкціонований доступ до ресурсів