

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ	10
ВСТУП	11
1 АНАЛІЗ РИНКУ І ПОТОЧНИХ РІШЕНЬ	13
1.1 Переваги та недоліки поточних рішень	13
1.1.1 Ліцензування і відкритість API	13
1.2 Технології розробки	16
1.3 Технічний огляд продукту	16
1.4 Історичний огляд корпоративної сфери	17
1.5 Портлети	17
1.5.1 Apache Pluto	18
1.6 Система керування вмістом	19
1.6.1 Головні функції CMS	19
1.6.2 Типи даних та їх використанням	20
1.6.3 Управління корпоративною інформацією	20
1.7 Система управління документами	21
1.7.1 Метадані	21
1.7.2 Інтеграція	22
1.7.3 Захоплення тексту	22
1.7.4 Індексування	22
1.7.5 Сховище	23
1.8 Програмне забезпечення для спільної роботи	23
1.8.1 Огляд ПЗ для спільної роботи	23
1.8.2 Види взаємодії	23
1.8.3 Рівні взаємодії	25

					<i>ДП.ПЗ 04.00.00.000 ПЗ</i>			
Зм.	Арк.	№ докум.	Підпис	Дата				
Розробив		Бойчук Я.В.			Розробка програмного алгоритмічного забезпечення багатофункціональної корпоративної системи для сумісної роботи, управління документами і проектами <i>ПОЯСНЮВАЛЬНА ЗАПИСКА</i>	Літ.	Аркуш	Аркушів
Перевірів		Випасняк Л.І.					6	122
Т.контр						<i>ІНФТУНГ ПЗ-07-1</i>		
Н.контр		Вовк Р.Б.						
Затв.		Юрчишин В.М.						

1.9	Інтранет	25
1.9.1	Особливості, переваги та недоліки Інтранет	26
1.10	Корпоративна Wiki	28
1.11	Онлайн офіс	29
1.12	Корпоративний блог	29
1.12.1	Внутрішньокорпоративний блог	29
1.12.2	Публічний блог	30
2	АЛГОРИТМІЧНЕ ЗАБЕЗПЕЧЕННЯ ПРОЦЕСУ СТВОРЕННЯ КОРПОРАТИВНОЇ СИСТЕМИ	31
2.1	Проектування системи	31
2.1.1	Модель архітектури системи	31
2.2	Проектування бази даних	33
2.2.1	InnoDB механізм	34
2.2.2	Відношення в таблицях	34
2.2.3	Архітектура бази даних	39
2.3	Зовнішній макет сайту	39
2.4	Структура проекту	42
2.4.1	Алгоритм роботи надбудови Java EE – Spring Framework	42
2.4.2	Діаграма відношень	44
2.5	Авторизація і аутентифікація	46
2.5.1	Алгоритм авторизації користувачів	46
2.6	Діаграма класів	47
3	ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ДЛЯ УПРАВЛІННЯ КОРИСТУВАЧАМИ, ДОКУМЕНТАМИ, ЗАВДАННЯМИ І МОЖЛИВОСТІ СПІЛЬНОЇ РОБОТИ	50
3.1	Реалізація роботи бази даних	50
3.2	Реалізація веб інтерфейсу	51
3.2.1	Навігаційна панель	53

3.2.2	Головне меню	54
3.3	Робота із даними	56
3.4	Категорії portalу	58
3.4.1	Авторизація	58
3.4.2	Корпоративна пошта	59
3.4.3	Календар	60
3.4.4	Завдання і задачі	61
3.4.5	Користувачі	61
3.4.6	Документи	63
3.4.7	Корпоративна wiki	64
3.4.8	Корпоративний блог	64
4	ЕКОНОМІЧНА ДОЦІЛЬНІСТЬ ВИКОРИСТАННЯ ПРОГРАМНО-ГО ЗАБЕЗПЕЧЕННЯ	65
4.1	Економічна доцільність розробки програмного забезпечення та його впровадження	65
4.2	Побудова мережевого графа	66
4.3	Економічне обґрунтування розробки та впровадження програми . .	71
4.3.1	Розрахунок витрат на розробку програмного забезпечення .	71
4.3.2	Розрахунок можливого прибутку	72
4.3.3	Розрахунок зведених економічних показників	73
5	ОХОРОНА ПРАЦІ	75
5.1	Значення охорони праці для забезпечення безпечних і здорових умов праці людей	75
5.2	Аналіз потенційних небезпек та шкідливих факторів виробничого середовища	76
5.3	Забезпечення нормальних умов праці при роботі з ЕОМ	78
5.4	Забезпечення безпеки монтажу, пусконаладжувальних, ремонтних робіт та експлуатації ЕОМ і комп'ютерних мереж	84

5.5 Пожежна безпека та безпека в НС	86
ВИСНОВКИ	90
СПИСОК ПОСИЛАНЬ НА ДЖЕРЕЛА	91
ДОДАТКИ	92
БІБЛІОГРАФІЧНА ДОВІДКА	122

					<i>ДП.ПЗ 04.00.00.000 ПЗ</i>	Аркуш
						9
Зм.	Арк.	№ докум.	Підпис	Дата		

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

СКБД – Система керування базою даних

API – Application programming interface

JPA – Java Persistence API

SQL – Structured Query Language

GUI – Graphical User Interface

UML – Unified Modelling Language

MCC(MVCC) – Multiversion concurrency control

					<i>ДП.ПЗ 04.00.00.000 ПЗ</i>	Аркуш
						10
Зм.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

В нас час важливою економічною точкою опори для будь-якої комерційної організації є наявність певних факторів, які визнають чітку позицію компанії на ринку. До всіх цих чинників можна віднести багато варіантів, зокрема:

- капітал підприємства;
- матеріальна база;
- технічна база;
- кваліфікований персонал;
- місце підприємства на внутрішньому і зовнішньому ринку;
- наявність сучасних засобів виробництва та ведення бізнесу;
- та інші.

Тому кожний розділ ведення бізнесу повинний бути детально розглянутий та впроваджений у життя. Але якщо подивитися із точки зору програмного забезпечення, то на даному етапі розвитку цивілізації, якісне ПЗ відіграє напевно найбільш важливу роль. Адже не можливо зараз утримувати всі дані в паперовому вигляді, не можливо відсилати друковані листи, чи спілкуватися тільки по телефоні і взнавати новини компанії тільки при зустрічі. У наш стрімкий час розвитку, новини міняються із колосальною швидкістю, тому встигнути за всім просто не можливо без певного програмного продукту. Уявіть собі інформатор, який сповіщає будь-які для Вас новини чи корисну інформацію в зручний для Вас час, при цьому вміє фільтрувати і аналізувати дані із попередніх запитів. Також на даний момент важко уявити не можливість спільної роботи над документами, над електронними таблицями. Дані технології вже давно використовуються людьми і підприємствами, починаючи від найменших де працює двоє людей, до величезних із кількістю працівників більше ста тисяч. Але для цього всього використовуються дуже багато технологій, які важко налаштувати і потребують великих витрат на підтримку. Тому було розроблено багато сервісів і додатків, які полегшують роботу в мережі для підприємств.

					<i>ДП.ПЗ 04.00.00.000 ПЗ</i>	Аркуш
Зм.	Арк.	№ докум.	Підпис	Дата		11

Дане ПЗ використовується у всіх нішах нашого життя, починаючи від шкіл і лікарень, закінчуючи величезними корпораціями з будівництва космічних кораблів. Тому розробити універсальний продукт, який забезпечить всі вимоги, просто не можливо. Для кожної сфери існують свої нюанси.

Цікавою нішею для дослідження стало корпоративне програмне забезпечення для малого і середнього бізнесу. На даний момент існує багато програмних продуктів для комерційних цілей, проте вони здебільшого розраховані на великі корпорації і підприємства. Тому використання їх для менших фірм просто не доцільно, або дуже складно із фінансової сторони (витрати на підтримку передують вигоді). Як відомо, на ринку до цих пір зберігається тенденція на попит на корпоративне програмне забезпечення, яке б відповідало вимогам малого і середнього бізнесу, і в той же час було практично придатним для використання у великих корпоративних цілях

					<i>ДП.ПЗ 04.00.00.000 ПЗ</i>	Аркуш
						12
Зм.	Арк.	№ докум.	Підпис	Дата		

1 АНАЛІЗ РИНКУ І ПОТОЧНИХ РІШЕНЬ

На даний момент існує досить багато готових рішень корпоративних порталів. Вони можуть забезпечувати підприємства всіма необхідними функціями і додатками, починаючи від системи обліку працівників, і завершуючи системою аналітики і збору даних, все це залежить від потреб ринку і певної компанії. Проте майже всі вони здебільшого призначені для великих компаній, і невеличкі компанії або повинні витратити величезні гроші на покупку ліцензій, або ж не користуватися усіма перевагами корпоративного порталу. Тому було проведено загальний огляд продуктів і виділено основні переваги і недоліки, також виділено поточну використовувану ліцензію розповсюдження ПЗ.

1.1 Переваги та недоліки поточних рішень

Зробивши аналіз даної сфери, можна виділити декілька основних аспектів, які будуть використані для розробки подальшого програмного продукту. Левова частка програмного забезпечення корпоративних порталів розроблено згідно стандартів[1]. Всі додатки і аплікації можуть без проблем взаємодіяти між собою. Проте великою їх нестачею для малої сфери бізнесу є закритість програмного коду і величезна вартість ліцензій. Тому невеличкі компанії (до 100 людей) просто не мають змоги собі дозволити таку «розкіш».

1.1.1 Ліцензіювання і відкритість API

Було проведено аналіз поточних продуктів і їх ліцензій, і виявлено, що майже 90% використовує проприетарні рішення. Більш детально розглянуто у таблиці 1.1:

Табл. 1.1 – Список корпоративних порталів і використовувана ліцензія

Назва продукту	Технологія	Ліцензія
Jetspeed	Java EE	Apache License v2.0
ATG Portal	Java EE	Proprietary

					ДП.ПЗ 04.00.00.000 ПЗ	Аркуш
						13
Зм.	Арк.	№ докум.	Підпис	Дата		

Назва продукту	Технологія	Ліцензія
Backbase Portal Software	Java EE, .NET	Proprietary
Broadvision Portal	Java EE	Proprietary
Bluenog ICE	Java EE	Proprietary
enPortal	Java EE	Proprietary
CommunityManager.NET	.NET	Proprietary
eXo Portal	Java EE	Affero General Public License
eXo Platform	Java EE	Proprietary
GateIn Portal	Java EE	LGPL
Hippo CMS	Java EE	Open Source and Proprietary Licenses
WebSphere Portal	Java EE	Proprietary
TeamPortal	Java EE	Proprietary
JBoss Enterprise Portal	Java EE	LGPL
IntraNet	ASP.NET	Proprietary
Liferay Portal	Java EE	Proprietary Licenses
TeamWox Groupware	C++	Proprietary
SharePoint Server	ASP.NET	Proprietary
Vignette Portal 8.0	Java EE	Proprietary
Oracle WebCenter Suite 11g	Java EE	Proprietary
Oracle WebLogic Portal 10g	Java EE	Proprietary
Oracle WebCenter Interaction 10g	ASP.NET	Proprietary
Oracle IAS Portal 10g	Java EE	Proprietary
Regroup	Ruby	Proprietary
ACUBE Portal 5.0	Java EE	Proprietary
SAP NetWeaver 7.0	Java EE	Proprietary
SORCE V9	ASP.NET	Proprietary
Sun Java System Portal Server	Java EE	Open Source, licensing & support plans
Sun GlassFish Web Space Server	Java EE	Open Source, licensing & support plans
tmsEKP 1.52	Java EE	Proprietary
PortalBuilder 5.2	Java EE	Proprietary
ProPortal 4.0	Java EE	Proprietary
Intrexx	Java EE	Proprietary
uPortal	Java EE	Apache License v2.0

Зробивши певний аналіз, можна дійти до висновку, якби невеликим компаніям давати можливість використовувати продукт на підставах вільного розповсюдження коду, то вони б охотніше пробували, і з часом інвестували в нього гроші, або просто «купляти» підтримку. Тобто використати одну із відкритих ліцензій типу GNU Public License.

Економічна вигода продукту буде базуватися на корпоративній платній підтримці, типу все починаючи від підбору серверів – до налаштування і підтримки продукту. Зате розробники зможуть у загальній репозиторії додавати свої зміни та виправляти помилки, що значною мірою пришвидшить процес розробки. Основна стратегія розробки буде націлена на швидкий вихід на ринок і пошук потенційних клієнтів. Також велика увага буде прикута для ринку пост радянських республік, адже на даний момент ринок бізнесу стрімко розвивається, тобто попит є, а пропозиція не повній мірі відповідає потребі. Портали які розробляються переважно націлені на Європейський та Американський ринок, а також Азію. Тому базуючись на цьому було виділено такі основні вимоги, як врахування нашого законодавства (для прикладу по працевлаштуванню працівників, ведення документації, конвертації валют і тому подібне) та локалізацію сервісу. Тим більше підтримка користувачів буде набагато легше і ефективніше всередині країни, ніж з-за кордону, що дасть нам перевагу над іншими існуючими продуктами.

Також велику увагу буде приділено відкритості API для взаємодії із вже існуючими додатками. Адже існуючі рішення в основному базуються на закритих протоколах, чим саме змушують користувачів прив'язуватися до їхньої системи і залежати від них

Інтерфейс і система всіх сучасних продуктів дуже «важка» і мультифункціональна, що потребує значних ресурсів як у користувачів так і на стороні сервера. Цей аспект також буде максимально спрощений, що в свою чергу дозволить виділитися продукту на ринку малого бізнесу.

					ДП.ПЗ 04.00.00.000 ПЗ	Аркуш
Зм.	Арк.	№ докум.	Підпис	Дата		15

1.2 Технології розробки

Базуючись на поточних стандартах [1] та використовуваних базових технологій (таблиця 1.1) було прийнято рішення впровадити розробку на базі Java. Адже саме Sun (зараз Oracle) «диктує» моду на ринку стандартизації портлетів, тому буде просто пристосуватися до поточних рішень.

Звичайно ж буде використано всі переваги Java EE. Для гнучкої і швидкої розробки буде застосовано Spring Framework із ORM обгорткою Hibernate поверх бази даних MySQL. Для фронтенд логіки UI в основному буде використовуватися jQuery фреймворк. Пошук забезпечить Apache Lucene. Сервер бекенду буде працювати на Apache Tomcat.

На даний момент не планується стандартизація щодо портлетів, просто в майбутньому можливо буде виділено цей пункт для реалізації в системі.

1.3 Технічний огляд продукту

Розглянемо більш детально пункт про наявність сучасних засобів ведення бізнесу. Кожна компанія, завжди стикається із проблемою ведення обліку працівників, ведення обліку фінансів, спільної роботи над документами та іншим. Також є величезна і невід’ємна потреба у спільному доступі до документів, до корпоративного календаря, до блогу користувачів, до електронних таблиць та інформаційної дошки.

Портал підприємства (також відомий як enterprise information portal (EIP) або корпоративний портал) є основою для інтеграції інформації, людей і процесів в рамках організації. Це дає змогу забезпечити єдину точку доступу, часто у вигляді веб-інтерфейсу і призначеної для агрегування та персоналізації інформації за допомогою конкретних програмних додатків. Однією відмінною рисою корпоративних порталів є децентралізоване внесення контенту та управління, яка зберігається на віддаленому сервері та постійно оновлюється.

					<i>ДП.ПЗ 04.00.00.000 ПЗ</i>	Аркуш
						16
Зм.	Арк.	№ докум.	Підпис	Дата		

1.4 Історичний огляд корпоративної сфери

В середині 1990-х років появилися громадські такі веб-портали як AltaVista, AOL, Excite і Yahoo!. Вони забезпечували користувачів певним набором функцій (наприклад новини, електронна пошта, погода, котирування акцій і пошук), які часто були представлені у вигляді автономного порталу. Незабаром підприємства усіх типів і форм почали бачити необхідність аналогічного функціоналу для їх різноманітних потреб, проте із єдиною точкою доступу.

До кінця 1990-х років, виробники програмного забезпечення почали розробляти веб-портали для різних підприємств. Ці програмні пакети були розроблені таким чином, щоб підприємства могли легко розгорнути свої власні налаштування корпоративного порталу та доповнювати його своїми додатками. Перші постачальники комерційних веб порталів з'явилися в 1998 році, це були такі фірми як: Epicentric, Plumtree та Viador. Ці фірми були основними гравцями на ринку, проте ситуація змінилася в 2002 року, коли на ринок почали виходити постачальники серверних аплікацій, такі як BEA, IBM, Passageways, Oracle Corporation and Sun Microsystems. Підприємства можуть вибрати для своїх цілей декілька порталів, що базується на основі їх бізнес-структури та стратегічної спрямованості.

У 2003 році розробники Java-порталів випустили стандарт, відомий як JSR-168. Він повинен був визначити API для взаємодії між корпоративних порталів та портлетів. Постачальники програмного забезпечення почали розробляти JSR-168 сумісні портлети, які можуть бути розгорнуті на будь-якому JSR-168 сумісному корпоративному порталі. Другий ітераційний стандарт JSR-286 є остаточним на даний момент і випущений 12 червня 2008 року.

1.5 Портлети

Портлет – це змінний компонент інтерфейсу веб-порталу (елемент веб-сторінки), який можливо певним чином підключити до порталу. Портлет містить в собі фрагменти розмітки, які вбудовуються в сторінку порталу. Найча-

					<i>ДП.ПЗ 04.00.00.000 ПЗ</i>	Аркуш
Зм.	Арк.	№ докум.	Підпис	Дата		17

стіше сторінка portalу представляється у вигляді набору портлетів, які взаємодіють між собою. Таким чином, портлет (або сукупність портлетів) представляється у вигляді єдиного веб-додатку, розміщеного на порталі. Приклади портлетів можуть бути наступними: електронна пошта, повідомлення про погоду, фінансовий стан, останні новини і тому подібне. Завдяки існуючим стандартам розробники можуть створювати портлети, що легко вбудовуються в будь-який портал, який слідує стандартам і правилам.

Існує протокол WSRP, що забезпечує стандарт веб-сервісів, що дозволяє автоматично вбудовувати віддалено запуснені портлети з різних джерел. Специфікація Java-портлетів JSR168 дає можливість взаємодіяти між собою портлети з різних веб-порталів. Ця специфікація визначає безліч API для взаємодії контейнерів портлетів і дає різні адреси областей персоналізації, подання та безпеки. Існує безліч постачальників комерційних контейнерів портлетів. Як відомо лідирують у цій галузі IBM, Oracle, Vignette. Реалізації від цих постачальників мають додаткові розширення і налаштування, проте деякі із ним можуть бути не затверджені стандартами. Крім того, є портали з відкритим вихідним кодом, що підтримують JSR168, такі як корпоративний портал Apache Jetspeed-2 або eXo Portal.

1.5.1 *Apache Pluto*

Розглянемо на прикладі один з найбільш вдалої реалізації стандарту портлетів JSR168 – це Apache Pluto. Портлет працює всередині контейнера портлетів (Pluto). Цей контейнер містить портлет з необхідним середовищем для подальшого виконання. Контейнер портлетів керує життєвим циклом всіх вікон portalу та надає інтерфейси для портлетів, котрі викликаються всередині нього. Контейнер також запускає методи на виконання із доступних цільових користувацьких сторінок, і взаємодіє із сторінками portalу. Принцип роботи і архітектурні компоненти аплікації Pluto 2.0. зображено на рисунку 1.1.

В даному випадку, Pluto вбудований безпосередньо в корпоративний портал. Потім через перехресний запит (через веб-додатки) відбувається відправлен-

					ДП.ПЗ 04.00.00.000 ПЗ	Архив
						18
Зм.	Арк.	№ докум.	Підпис	Дата		

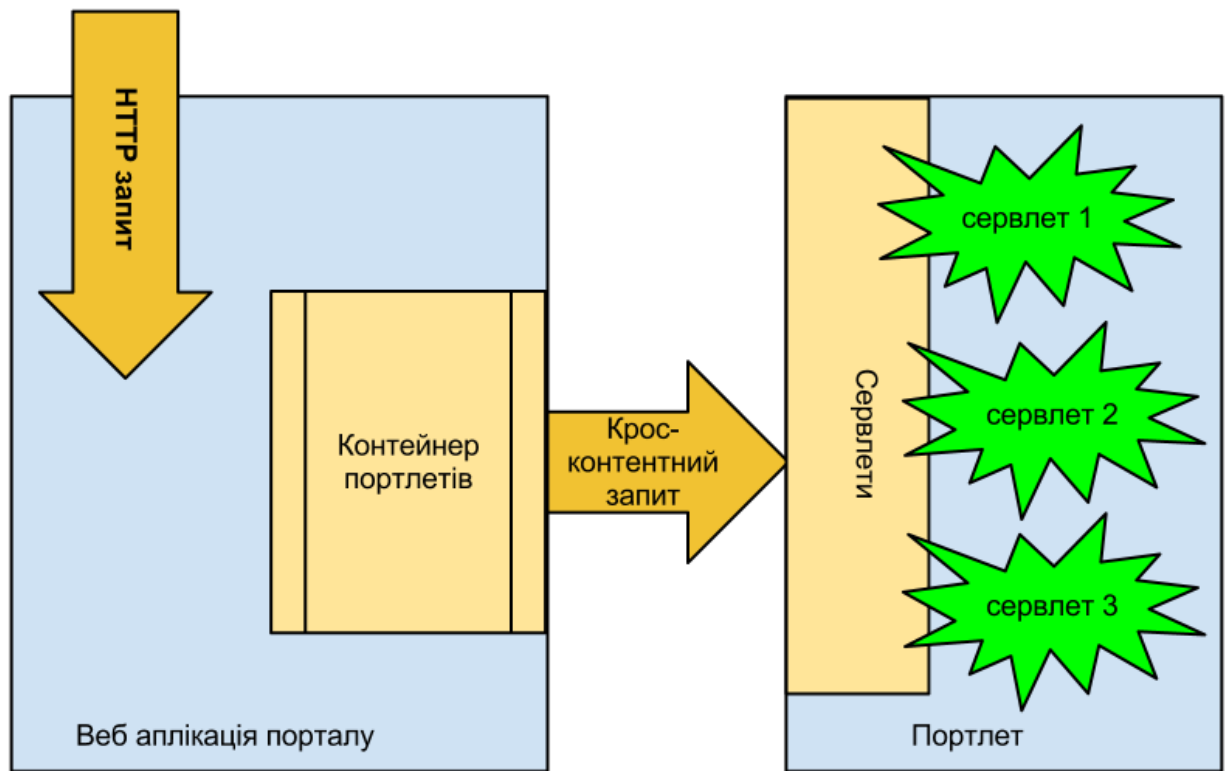


Рисунок 1.1 – Принцип роботи Pluto

ня запиту для відображення вмісту портлету, який як правило знаходяться в різних додатках на порталі і в контейнерах.

1.6 Система керування вмістом

Система управління контентом (content management system - CMS) дозволяє публікувати, редагувати і змінювати вміст веб-сторінок, а також обслуговувати портал з центральної сторінки. При цьому надається набір процедур, що використовуються для управління робочим процесом у середовищі для спільної роботи. Вони можуть бути ручні або комп'ютеризовані (в автоматичному режимі).

1.6.1 Головні функції CMS

До основних функцій можна віднести наступні пункти:

- можливість великій кількості людей ділитися інформацією і робити свій вклад в розвиток порталу;

Зм.	Арк.	№ докум.	Підпис	Дата

ДП.ПЗ 04.00.00.000 ПЗ

Архив

- контроль доступу до даних на основі ролей користувачів (наприклад визначити роль, яка має тільки права на перегляд інформації, або ж редагування, публікацію тощо);
- пошук і поширення інформації між користувачами;
- зменшення дублікацій на вході;
- спрощене керування корпоративними додатками;
- відносно легка комунікація між користувачами.

1.6.2 Типи даних та їх використання

У CMS дані можуть бути представлені як правило у будь-якій формі: документи, відео, тексти, фотографії, номери телефонів, наукові дані і тому подібне. CMS часто використовуються для зберігання, управління, перегляду і публікації документів. Також досить поширене використання в якості центрального сховища у зв'язці із централізованою системою контролю версій, що є однією із переваг CMS.

1.6.3 Управління корпоративною інформацією

Enterprise Content Management (ECM) - управління інформаційними ресурсами підприємства або управління корпоративною інформацією. В даному контексті інформація (контент) передбачається як слабо структурована одиниця - це можуть бути файли різних форматів, електронні документи з різними наборами полів і т. п. За визначенням ECM - це стратегічна інфраструктура і технічна архітектура для підтримки єдиного життєвого циклу неструктурованої інформації різних типів і форматів. ECM-системи складаються з додатків, які можуть взаємодіяти між собою, а також використовуватися і продаватися самостійно.

Всі сучасні ECM-системи визначають такі ключові компоненти:

- управління документами — довгострокове архівування, автоматизація політик зберігання та відповідності нормам регулюючих органів, забезпечення відповідності законодавчим та галузевим нормам;

					<i>ДП.ПЗ 04.00.00.000 ПЗ</i>	Аркуш
						20
<i>Зм.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>		

- управління веб-контентом (WCM) — автоматизація ролі веб-майстра, управління динамічним контентом і взаємодією між користувачами;
- управління мультимедіаконтентом (DAM) — управління графічними, відео та аудіофайлами, різними маркетинговими матеріалами, наприклад, флеш-банерами, рекламними роликами;
- управління знаннями (Knowledge Management) — підтримка систем для накопичення та доставки релевантної для бізнесу інформації;
- документо-орієнтоване взаємодія (співробітництво) — спільне використання документів користувачами та підтримка проектних команд.

1.7 Система управління документами

Система управління документами (DMS - Document management system) - комп'ютерна система (або набір комп'ютерних програм), що використовується для відстеження та зберігання електронних документів і / або образів (зображень та інших артефактів) паперових документів. Дане поняття тісно пов'язане з концепцією Content Management System (система керування вмістом) і зазвичай розглядається як компонент Enterprise Content Management System (CMS рівня підприємства). У загальному випадку системи управління документами (DMS) надають можливість зберігання, ведення контролю версій, позначення метаданими і безпеку по відношенню до документів, а також індексування і розвинені можливості пошуку документів.

1.7.1 Метадані

Метадані зазвичай зберігаються для кожного документа. Метадані, наприклад, можуть включати дату занесення документа в сховище і код користувача, котрий виконав зміни до файлу. Система управління документами також може витягувати метадані з документа автоматично або запитувати їх у користувача. Деякі системи надають сервіс оптичного розпізнавання тексту відсканованих документів, або можливість витягувати текст з електронних документів. Викори-

					ДП.ПЗ 04.00.00.000 ПЗ	Аркуш
Зм.	Арк.	№ докум.	Підпис	Дата		21

стовуючи опрацьований текст система дозволяє здійснювати пошук документа за ключовими словами всередині самого документа.

1.7.2 Інтеграція

Багато систем управління документами намагаються інтегрувати функцію управління документами безпосередньо в різні додатки, дозволяючи користувачеві отримувати документ відразу зі сховища системи управління документами, робити які-небудь модифікації, і зберігати його назад в сховище в якості нової версії, і все це проробляти в одному додатку, не виходячи з нього. Дана інтеграція в основному доступна для офісних пакетів і поштових клієнтів або для програмного забезпечення, призначеного для групової або колективної роботи. Інтеграція зазвичай має на увазі використання таких відкритих стандартів як: ODMA, LDAP, WebDAV і SOAP.

1.7.3 Захоплення тексту

Під захопленням тексту мається на увазі переведення паперових документів в цифровий варіант за сканерів та МФУ. Також часто використовується програмне забезпечення для оптичного розпізнавання тексту, щоб конвертувати цифрові зображення в текст.

1.7.4 Індексування

Індексування надає можливість класифікувати документи за допомогою метаданих і індексування словникового тексту, який було витягнутого з документа. Індексція існує для підтримки розвинених можливостей пошуку документів. Одна з головних умов швидкого та якісного пошуку - це створення індексу документа.

					<i>ДП.ПЗ 04.00.00.000 ПЗ</i>	Аркуш
Зм.	Арк.	№ докум.	Підпис	Дата		22

1.7.5 Сховище

Основне призначення це для зберігання електронних версіях документів. Сховище документів також включає в себе і керування тими ж документами, котрі воно зберігає. Також сховище забезпечує міграцію з одного носія на інший і забезпечує цілісність даних. Сховище документів може бути як файлове, так і сховище у вигляді СУБД (бази даних). У свою чергу, сховище документів в СУБД може бути як в одній базі даних, так і в окремо розподілених базах даних.

1.8 Програмне забезпечення для спільної роботи

Програмне забезпечення для спільної роботи (англ. collaborative software, groupware, workgroup support systems, group support systems) - програмне забезпечення створене з метою підтримки взаємодії між людьми, котрі спільно працюють над вирішенням деяких спільних завдань.

1.8.1 Огляд ПЗ для спільної роботи

Програмне забезпечення для спільної роботи – це область, яка в значній мірі перекривається з областю CSCW (англ. computer-supported cooperative work (CSCW)). Часто вважається що ці області еквівалентні, хоча з іншого боку програмне забезпечення для спільної роботи є підчастиною CSCW. Сюди відносяться такі системи як: електронна пошта, календарі, текстовий чат, вікі сторінки, корпоративні закладки, блог. Оскільки ПО спільної роботи відноситься до технологічних елементів CSCW, системи спільної роботи стають корисним аналітичним інструментом у вивченні поведінкових і організаційних параметрів, пов'язаних з більш широкою сферою CSCW.

1.8.2 Види взаємодії

В літературі можна зустріти кілька різних визначень спільної роботи (англ. - collaboration) в застосуванні до інформаційних технологій. Деякі з них виправда-

					ДП.ПЗ 04.00.00.000 ПЗ	Аркуш
						23
Зм.	Арк.	№ докум.	Підпис	Дата		

ні, інші ж настільки великі, що починають втрачати будь-який сенс. Для того щоб бути впевненим що обрані технології підходять для конкретних потреб, необхідно розуміти відмінності в способах взаємодії людей один з одним. Є три основні шляхи, по яких здійснюється взаємодія між людьми:

- діалог;
- здійснення угоди;
- співробітництво.

Діалог - це обмін інформацією між одним або кількома учасниками, основна мета якого полягає у з'ясуванні їх позицій і встановлення взаємин. Відбувається вільний обмін інформацією без будь-яких обмежень. Для підтримання діалогу цілком підходять звичайні комунікаційні технології, такі як телефон, миттєві повідомлення та електронна пошта.

Укладення угоди передбачає обмін якимись сутностями, і ця процедура зазвичай проводиться за добре певними правилами і передбачає зміну відносин між учасниками. Наприклад, один з учасників угоди обмінює гроші на товари і стає покупцем. Новий статус учасників операції та обмінюваних сутностей потрібно зберегти в будь-якому надійному сховищі. Такі операції добре обслуговуються системами управління транзакціями.

Співпраця полягає в тому, що його учасники обмінюються якимись загальними сутностями (на противагу угоді, коли предмет обміну належить лише одному учаснику). Як приклад можна привести просування нової ідеї, створення нової конструкції, досягнення спільних цілей. При цьому самі сутності досить розпливчасті і невизначені. Таким чином, технології для забезпечення спільної роботи теж повинні бути достатньо гнучкими. Вони повинні включати в себе управління документами, кошти для ведення обговорень з можливістю сортування за темами, можливість відновити історію внесених змін та багато іншого.

1.8.3 Рівні взаємодії

Рівні взаємодії можна поділити на три категорії по рівню забезпечення взаємодії: засоби зв'язку, засоби для організації конференцій та засоби управління.

Електронні засоби зв'язку використовуються для пересилання повідомлень, файлів, даних чи документів між людьми і таким чином дають можливість для обміну інформацією:

- електронна пошта;
- факс;
- голосова пошта;
- веб-публікації.

Електронні конференції також дають змогу для обміну інформацією, проте в інтерактивній формі це є:

- телефонні конференції;
- відео і аудіо конференції;
- інтернет форуми;
- чати.

Засоби управління діяльність групи:

- електронні календарі (створення щоденників, системи автоматичного нагадування);
- системи управління проектами (складання розкладу робіт, відслідковування їх виконання);
- управління документообігом;
- бази знань - збір, сортування, зберігання і організація доступу до різних форм інформації.

1.9 Інтранет

Інтранет (англ. Intranet, також вживається термін інтрамережа) - на відміну від мережі Інтернет, це внутрішня приватна мережа організації. Як правило,

					<i>ДП.ПЗ 04.00.00.000 ПЗ</i>	Аркуш
						25
Зм.	Арк.	№ докум.	Підпис	Дата		

Інтранет - це Інтернет в «мініатюрі», який побудований на використанні протоколу IP для обміну і спільного використання деякої частини інформації всередині певної організації. Це можуть бути списки співробітників, списки телефонів партнерів і замовників. Найчастіше під цим терміном мають на увазі тільки видиму частину Інтранет - внутрішній веб-сайт організації. Заснований на базових протоколах HTTP і HTTPS і організований за принципом клієнт-сервер, інтранет-сайт доступний з будь-якого комп'ютера через браузер.

Таким чином, Інтранет - це «приватний» Інтернет, обмежений віртуальним простором окремо взятої організації. Intranet допускає використання публічних каналів зв'язку, що входять в Інтернет, (VPN), але при цьому забезпечується захист переданих даних і мають набір заходів щодо припинення проникнення ззовні на корпоративні вузли.

Програми в Intranet засновані на застосуванні Інтернет-технологій і особливо Web-технології: гіпертекст у форматі HTML, протокол передачі гіпертексту HTTP і інтерфейс серверних додатків CGI. Складовими частинами Intranet є Web-сервери для статичної або динамічної публікації інформації і браузери для перегляду й інтерпретації гіпертексту.

1.9.1 Особливості, переваги та недоліки Інтранет

Інтранет побудований на базі тих же понять і технологій, які використовуються для Інтернету, такі як архітектура клієнт-сервер і стек протоколів Інтернету (TCP / IP). В Інтранет зустрічаються все з відомих інтернет-протоколів, наприклад, протоколи HTTP (веб-служби), SMTP (електронна пошта) і FTP (передача файлів). Інтернет-технології часто використовуються для забезпечення сучасними інтерфейсами функції інформаційних систем, які розміщують корпоративні дані.

Інтранет можна представити як приватну версію Інтернету, або як приватне розширення Інтернету, обмеженого організацією за допомогою брандмауера.

Перші інтранет-веб-сайти і домашні сторінки почали з'являтися в органі-

					<i>ДП.ПЗ 04.00.00.000 ПЗ</i>	Аркуш
Зм.	Арк.	№ докум.	Підпис	Дата		26

заціях у 1990-1991 роках. Проте за неофіційними даними, термін Інtranет вперше почав використовуватися в 1992 році в таких закладах, як університети і корпорації, що працюють у технічній сфері.

Інtranет також протиставляють Екстрaнет, доступ до Інtranету надано тільки службовцям організації, в той час як до Екстрaнет можуть отримати доступ клієнти, постачальники, або інші затверджені керівництвом особи. В Екстрaнет-технології крім приватної мережі, користувачі мають доступ до Інтернет ресурсів, але при цьому здійснюються спеціальні заходи для безпечного доступу, авторизації, і аутентифікації.

Інtranет компанії не обов'язково повинен забезпечувати доступ до Інтернету. Коли такий доступ все ж забезпечується, зазвичай це відбувається через мережевий шлюз з брандмауером, захищаючи Інtranет від несанкціонованого зовнішнього доступу. Мережевий шлюз часто також здійснює аутентифікацію користувачів, шифрування даних, і часто - можливість з'єднання по віртуальній приватній мережі (VPN) що знаходяться за межами підприємства.

Переваги використання Інtranет:

- висока продуктивність при спільній роботі над якимись загальними проектами;
- легкий доступ персоналу до даних;
- гнучкий рівень взаємодії: можна міняти бізнес-схеми взаємодії як по вертикалі, так і по горизонталі;
- миттєва публікація даних на ресурсах Інtranет дозволяє специфічні корпоративні знання завжди підтримувати у формі і легко отримувати звідусіль в компанії, використовуючи технології Мережі та гіпермедіа;
- дозволяє проводити в життя загальну корпоративну культуру і використовувати гнучкість і універсальність сучасних інформаційних технологій для управління корпоративними роботами.

Переваги веб-сайту в Інtranет перед клієнтськими програмами архітектури клієнт-сервер:

					<i>ДП.ПЗ 04.00.00.000 ПЗ</i>	Архив
Зм.	Арк.	№ докум.	Підпис	Дата		27

- Не потрібно інсталяція програми-клієнта на комп'ютерах користувачів (як неї використовується браузер).
- Відповідно, при змінах функціональності корпоративної інформаційної системи оновлення клієнтського ПЗ також не потрібно.
- Скорочення тимчасових витрат на рутинних операціях по вводу різних даних, завдяки використанню веб-форм замість обміну даними по електронній пошті
- Крос-платформна сумісність.

Основні недоліки Інтранет:

- мережа може бути зламана і використана в хакерських цілях;
- неперевірена або неточна інформація, опублікована в Інтранет, призводить до плутанини і непорозумінь;
- легкий доступ до корпоративних даних може спровокувати їх витік до конкурентів через несумлінного працівника;
- працездатність і гнучкість Інтранет вимагають значних накладних витрат на розробку і адміністрування.

1.10 Корпоративна Wiki

Корпоративна вікі — це програмне забезпечення яке призначене для використання в корпоративній сфері і служить особливим чином для підвищення внутрішнього обміну знаннями, з великим акцентом на такі функції, як контроль доступу, інтеграція з іншими програмними продуктами та управління документами.

В організаціях вікі може або додати або замінити централізовану систему керування контентом. Її децентралізований характер дозволяє швидкому поширенню необхідної інформації в межах організації. Вікі являється швидшим організаційним продуктом ніж централізований репозиторій знань. Вікі може використовуватися для управління проектами, взаємодією з клієнтами, планування ресурсів підприємства а також інші види управління даними.

					<i>ДП.ПЗ 04.00.00.000 ПЗ</i>	Аркуш
Зм.	Арк.	№ докум.	Підпис	Дата		28

Особливості вікі для корпорації включають в себе такі основні аспекти як:

- швидкий і простий доступ для створення сторінок, які містять посилання на інші корпоративні системи;
- дозволяє розвантажити електронну пошту за рахунок зберігання всієї необхідної інформації із можливістю спільного доступу людьми які є на даному проекті.
- гнучка організація інформації;
- швидкий і розширений пошук.

1.11 Онлайн офіс

Онлайн офіс — це набір веб-сервісів у формі програмного забезпечення яке подану кінцевому користувачеві як послуга. Набір наданих веб-служб зазвичай включає всі основні можливості традиційних офісних пакетів, такі як текстовий редактор, електронні таблиці, додаток для створення презентацій, органайзер справ і навіть аналоги СУБД. Онлайн офіс може бути доступний з будь-якого комп'ютера, у якого є доступ в Інтернет, незалежно від того, яку операційну систему користувач використовує. Це дозволяє людям працювати разом по всьому світу і в будь-який час, що веде до створення міжнародних віртуальних команд для спільної роботи над проектами.

1.12 Корпоративний блог

Корпоративний блог — це блог, що видається організацією і використовується як для зв'язків з громадськістю, так і для внутрішньої організації. Або повністю підконтрольний організації, координований і наповнюється нею контентом, але формально з нею не пов'язаний.

1.12.1 Внутрішньокорпоративний блог

Внутрішній корпоративний блог – це важливий засіб комунікації, особливо у великих компаніях. Можна навести деякі явні переваги:

					ДП.ПЗ 04.00.00.000 ПЗ	Аркуш
Зм.	Арк.	№ докум.	Підпис	Дата		29

- блог допомагає поліпшити взаємодію співробітників, надає можливості для навчання. Він добре підходить для запуску нових проектів, для роботи в неоднорідних, великих колективах;
- блог допомагає виявити різні погляди на будь-яке питання. Відкритість для публікації постів і коментарів – хороша можливість висловитися всім членам колективу;
- шляхом дискусій на задану тему блог допомагає знайти компроміс при наявності різних точок зору. Для керівників блог – можливість налагодити взаємодію з співробітниками;
- блог – це своєрідна «історія фірми», архів ідей і обговорень.
- найчастіше кожен співробітник може залишити коментар до будь-якого посту. Коло авторів блогу визначається політикою компанії, часто написати пост може будь-який співробітник.

Блог має певні переваги перед такими внутрішньокорпоративними комунікаціями, як, наприклад, листування по електронній пошті, зокрема:

- коли листів стає занадто багато, це ускладнює спілкування;
- не всі співробітники вміють правильно архівувати листи, в результаті чого вони не зможуть згодом знайти необхідну інформацію.

Внутрішній блог – альтернатива чи доповнення до корпоративних зборів, нарад. Співробітники великих компаній часто не мають можливості проводити наради (наприклад, через велику відстань між філіями або зайнятості).

1.12.2 Публічний блог

Одна з основних цілей компаній – це налагодження комунікацій з клієнтами. Завдяки оперативності публікації постів і можливості коментування публічний корпоративний блог дуже важливий для досягнення цієї мети. Блоги є цінним доповненням до корпоративного сайту, так як в них може бути представлена альтернативна точка зору на те чи інше питання, ті чи інші продукти компанії можуть бути описані більш простою і доступною мовою.

					<i>ДП.ПЗ 04.00.00.000 ПЗ</i>	Аркуш
						30
<i>Зм.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>		

2 АЛГОРИТМІЧНЕ ЗАБЕЗПЕЧЕННЯ ПРОЦЕСУ СТВОРЕННЯ КОРПОРАТИВНОЇ СИСТЕМИ

2.1 Проектування системи

На початку розробки будь-якого програмного продукту слід значну увагу приділити проектуванню системи, адже саме від цього буде залежати легкість і правильність подальшої розробки системи, її підтримка і удосконалення. Саме тому проектування визначає основну складову програмного забезпечення. Необхідні пункти для успішного запуску продукту:

- побудова UML діаграми класів;
- побудова діаграми відношень між об'єктами;
- проектування бази даних;
- створення макетів майбутнього інтерфейсу;
- чітке розмежування модулів системи і їх взаємодія і тому подібне.

Як було згадано вище, перш за все слід розробити діаграму класів, показати всі взаємовідношення між об'єктами, їх роль у системі та загальну взаємодію.

2.1.1 Модель архітектури системи

Вся робота системи організована за принципом MVC шаблону.

Модель-вид-контролер – архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення.

Як показана на рисунку 2.1 цей шаблон поділяє систему на три частини: модель даних, вигляд даних та керування. Застосовується для відокремлення даних (модель) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача.

Головна мета використання даного шаблону – гнучкий дизайн програмного забезпечення, який повинен полегшувати подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонент

					ДП.ПЗ 04.00.00.000 ПЗ	Архум
Зм.	Арк.	№ докум.	Підпис	Дата		31

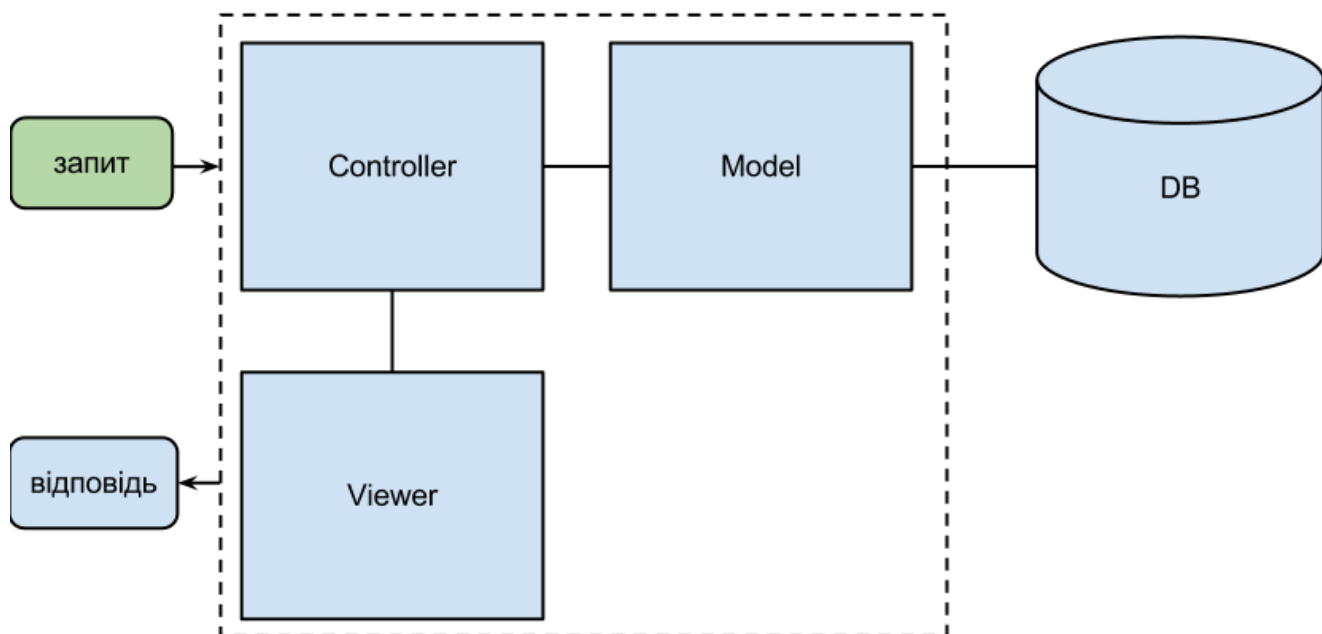


Рисунок 2.1 – Архітектура MVC

програми. Крім того використання цього шаблону у великих системах призводить до певної впорядкованості їх структури і робить їх зрозумілішими завдяки зменшенню складності.

Як згадувалося раніше, архітектурний шаблон Модель Вид Контролер (MVC) поділяє програму на три частини. У тріаді до обов'язків компоненту Модель (Model) входить зберігання даних і забезпечення інтерфейсу до них. Вигляд (View) відповідальний за представлення цих даних користувачеві. Контролер (Controller) керує компонентами, отримує сигнали у вигляді реакції на дії користувача, і повідомляє про зміни компоненту Модель. Така внутрішня структура в цілому поділяє систему на самостійні частини і розподіляє відповідальність між різними компонентами.

MVC поділяє цю частину системи на три самостійні частини: введення даних, компонент обробки даних і виведення інформації. Модель, як вже було відмічено, інкапсулює ядро даних і основний функціонал з їх обробки. Також компонент Модель не залежить від процесу введення або виведення даних. Компонент виводу Вигляд може мати декілька взаємопов'язаних областей, наприклад, різні таблиці і поля форм, в яких відображається інформація. У функції Контролера входить моніторинг за подіями, що виникають в результаті дій користувача

(зміна положення курсора миші, натиснення кнопки або введення даних в текстове поле). Зареєстровані події транслюються в різні запити, що спрямовуються компонентам Моделі або об'єктам, відповідальним за відображення даних. Відокремлення моделі від вигляду даних дозволяє незалежно використовувати різні компоненти для відображення інформації. Таким чином, якщо користувач через Контролер внесе зміни до Моделі даних, то інформація, подана одним або декількома візуальними компонентами, буде автоматично відкоригована відповідно до змін, що відбулися.

2.2 Проектування бази даних

База даних базується на СКБД MySQL - відкритій системі. Взаємозв'язок із користувачами відбувається через Модель системи (див. рис. 2.1). В базі містяться всі необхідні дані. Паролі користувачів зберігаються у шифрованому вигляді.

Реляційна система керування базами даних (РСКБД; інакше Система керування реляційними базами даних, СКРБД) — СКБД, що керує реляційними базами даних.

Поняття реляційний (англ. relation – відношення) пов'язане з розробками відомого англійського спеціаліста в області систем баз даних Едгара Кодда (Edgar Codd).

Ця модель характеризується простотою структури даних, зручним для користувача табличним представленням і можливістю використання формального апарату алгебри відношень і реляційного обчислення для обробки даних.

Реляційна модель орієнтована на організацію у вигляді двовимірних таблиць. Кожна реляційна таблиця являє собою двовимірний масив і має такі властивості:

- кожний елемент таблиці - один елемент даних;
- всі комірки в стовпці таблиці однорідні, тобто всі елементи в стовпці мають однаковий тип;
- кожний стовпець має унікальне ім'я;

- однакові рядки в таблиці відсутні;
- порядок наступності рядків і стовпців може бути довільним.

Базовими поняттями реляційних СКБД є:

- атрибут;
- відношення;
- кортеж.

2.2.1 InnoDB механізм

InnoDB це потужний механізм (рушій) для зберігання даних, розроблений фінською компанією Innobase Oy, яка була придбана в 2006 році концерном Oracle Corporation.

Поширюється за ліцензією GNU General Public License. Є у всіх нових версіях MySQL, і, починаючи з версії 5.5 для MySQL механізм за замовчуванням.

Застосування InnoDB дозволяє використання базою даних таких функцій, як транзакції, зовнішні ключі. Він також сумісний з ACID.

У цьому рушії є два способи для зберігання даних: файл або група файлів, загальних для всіх баз даних і таблиць, або один файл даних для кожної таблиці. Інші важливі особливості InnoDB: блокування на рівні рядків, можливість стиснення даних, і MVCC.

2.2.2 Відношення в таблицях

Головною одиницею бази даних є таблиця яка відповідає за дані користувачів, адже саме від неї залежать більшість таблиць. Для прикладу це повідомлення, чи завдання.

Для зв'язку між таблицями використовуються зовнішні ключі. Зовнішній ключ це – атрибут (набір атрибутів) в деякому відношенні R, який відповідає первинному ключу іншого відношення або того ж таки відношення R.

Ці взаємозв'язки представляються у вигляді відношень. Розділяють три види відношень:

					ДП.ПЗ 04.00.00.000 ПЗ	Аркуш
Зм.	Арк.	№ докум.	Підпис	Дата		34

- Багато-до-багато (n:m)
- Один-до-багато (1:m)
- Один-до-одного (1:1)

Багато-до-багатьох SQL відносин використовується, коли деяка невизначена кількість рядків (n) в таблиці пов'язані з невизначеною кількістю рядків (m), які зберігаються в іншій таблиці. Це називається: (m:m) відносини, тому що (n) рядків у першій таблиці, відносяться до (m) рядків в іншій.

Потрібно бути впевненим, що використовується кількість рядків строго більше ніж одиниця, тому що у випадку із одиницею слід використовувати відношення один-до-багатьох (1:n).

Розглянемо приклад, який зберігає країни в одну таблицю і мови в іншу таблицю. Є країни в світі, де більш ніж одна офіційний мова, і відповідно є випадки коли говорять одною мовою більш ніж в одній країні. Саме для цього призначене відношення багато-до-багатьох.

Приклад створення такої бази даних продемонстрований нище:

```
-----
-- Table `Country`
-----
```

```
CREATE TABLE `Country` (
  `countryId` INT UNSIGNED NOT NULL AUTO_INCREMENT ,
  `countryName` VARCHAR(45) NOT NULL ,
  PRIMARY KEY (`countryId`) );
```

```
-----
-- Table `Language`
-----
```

```
CREATE TABLE `Language` (
  `languageId` INT UNSIGNED NOT NULL AUTO_INCREMENT ,
  `languageName` VARCHAR(45) NOT NULL ,
  PRIMARY KEY (`languageId`) );
```

```
-----
```

```
-- Table `Country2Language`
```

```
-----
CREATE TABLE `Country2Language` (
  `Country_countryId` INT UNSIGNED NOT NULL ,
  `Language_languageId` INT UNSIGNED NOT NULL ,
  PRIMARY KEY (`Country_countryId`, `Language_languageId`) ,
  INDEX `fk_Country_has_Language_Language1` (`Language_languageId`
    `ASC`) ,
  INDEX `fk_Country_has_Language_Country` (`Country_countryId`
    `ASC`) ,
  CONSTRAINT `fk_Country_has_Language_Country`
    FOREIGN KEY (`Country_countryId`)
    REFERENCES `Country` (`countryId`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Country_has_Language_Language1`
    FOREIGN KEY (`Language_languageId`)
    REFERENCES `Language` (`languageId`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION);
```

Реалізація такої структури таблиць зображено на рисунку 2.2.

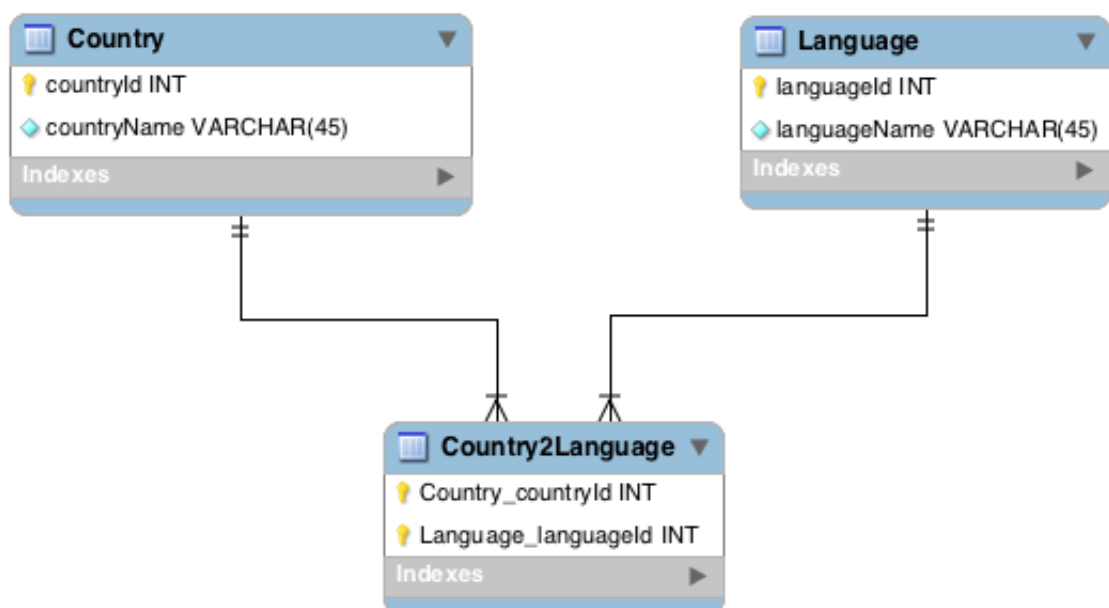


Рисунок 2.2 – Відношення багато-до-багато

Один-до-багато (1: n) відношення є дуже поширеним в SQL. Основна ідея полягає в тому, що кожен рядок зберігається в одній таблиці та пов'язаний з невизначеною кількістю рядків у іншій таблиці. Це може бути будь-яке число між 0 і (n) рядків.

Дуже хорошим прикладом є SQL база даних, яка зберігає замовлення в одній таблиці (перша частина) і порядок позицій в іншій (n-на сторона). Приклад такої структури таблиць зображено на рисунку 2.3.

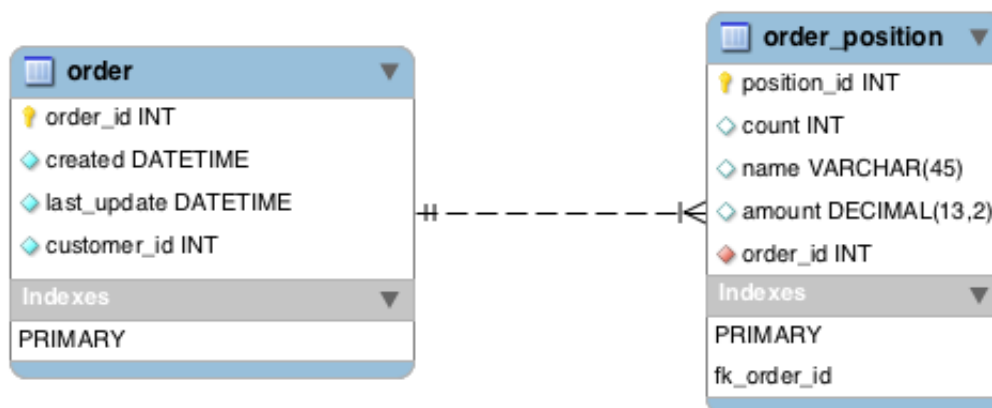


Рисунок 2.3 – Відношення один-до-багато

Реалізувати відношення один-до-багатьох дуже просто. Відносини зберігаються за допомогою спеціальної колонки в n-таблиці. Ця колонка містить один первинний ключ. Якщо присутні ще первинні ключі, то просто потрібно додати ще декілька стовпців.

У наведеному вище прикладі, потрібно додати до основної колонки таблиці «order» (поле «order_id») позицію в таблицю «order_position». Для того щоб забезпечити цілісність даних, використовуються зовнішні ключі.

Основною перевагою один-до-багатьох є доступність нормалізації таблиць і не при цьому не потрібно нагромаджувати таблицю непотрібними даними.

Один-до-одного відношення в SQL використовують для того щоб розділити великі таблиці на менші при цьому без втрат продуктивності. В деяких випадках це краще ніж величезна EAV [7] таблиця.

Ідеальний варіант використання відносин 1:1 для не пов'язаних зв'язків, які поділяють основні ознаки, як для прикладу каталог записів. Приклад такої

структури таблиць зображено на рисунку 2.4.

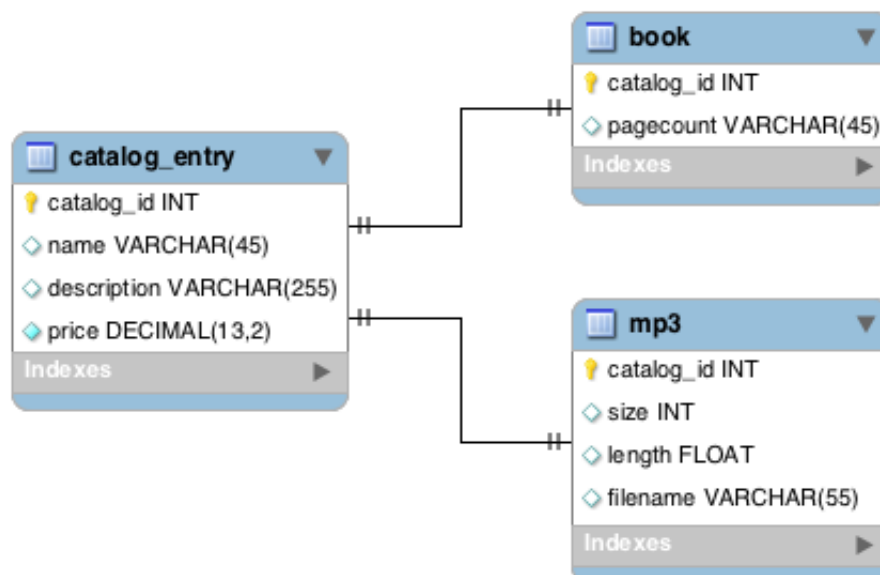


Рисунок 2.4 – Відношення один-до-одного

Ідея цієї концепції полягає в тому, що таблиця, яка містить основні атрибути, які є загальною для всіх суб'єктів. У першому каталозі загальні атрибути містять назва продукту, опис і ціну для прикладу. А самі атрибути, які є специфічними для певних видів продукції зберігаються в окремих таблицях. Зв'язок між цими таблицями зберігається за допомогою тих же первинних ключів в кожній таблиці.

Як ви можете бачити в наведеному вище прикладі, один-до-одного може бути реалізовано за допомогою простого використання тих же первинних ключів для таблиці.

Крім того, можливо використовувати різні первинні ключі і додавати відносини кожній колонці таблиці, але цей підхід має свої недоліки: SQL буде досить складним і заплутаним, і якщо використовувати зовнішні ключі, то це призведе до кругової залежності.

Первинний ключ – атрибут, або набір атрибутів, що однозначно ідентифікує кортеж даного відношення. Первинний ключ обов'язково унікальний, він єдиний і найголовніший із унікальних ключів. В реляційних базах даних первинний ключ задається обмеженням PRIMARY KEY. Для прикладу:

```
CREATE TABLE users(id INTEGER PRIMARY KEY AUTO_INCREMENT, name
CHAR(20), surname CHAR(40));
```

2.2.3 Архітектура бази даних

Як згадувалося вище майже все базується на відношенні до таблиці користувачів. Відношення повинні базуватися на основі моделі один-до-багатьох. Загальна модель бази даних зображена на рисунку 2.5.

Як показано на рисунку 2.5 – всі таблиці у базі даних взаємозв'язані. На таблицю користувачів посилаються таблиці документів, коментарів, блогів, календарів, команд, регіонів, типу робіт, вікі, завдань та повідомлень. Це в свою чергу дає змогу забезпечити відношення користувача до певної категорії. Для прикладу один користувач може мати декілька документів чи коментарів, звідси і слідує використання відношення один-до-багатьох.

Також допоміжні таблиці як категорія документів, кімнати, категорія вікі та категорія завдань мають свої відношення на головні таблиці (для прикладу одна категорія завдань може містити багато завдань).

2.3 Зовнішній макет сайту

Основою для користувача є зовнішній вигляд - UI (user interface) інтерфейс користувача. Інтерфейс користувача – сукупність засобів для обробки та відображення інформації, максимально пристосованих для зручності користувача; у графічних системах інтерфейс користувача реалізовується багатовіконним режимом, змінами кольору, розміру, видимості (прозорість, напівпрозорість, невидимість) вікон, їхнім розташуванням, сортуванням елементів вікон, гнучкими налаштуваннями як самих вікон, так і окремих їхніх елементів (файли, папки, ярлики, шрифти тощо), доступністю багатокористувацьких налаштувань.

Кінцевому користувачу не завжди цікаво що відбувається всередині програми – тому розробляють макет майбутнього проекту, в нашому випадку – це макет веб інтерфейсу.

					ДП.ПЗ 04.00.00.000 ПЗ	Архум
Зм.	Арк.	№ докум.	Підпис	Дата		39

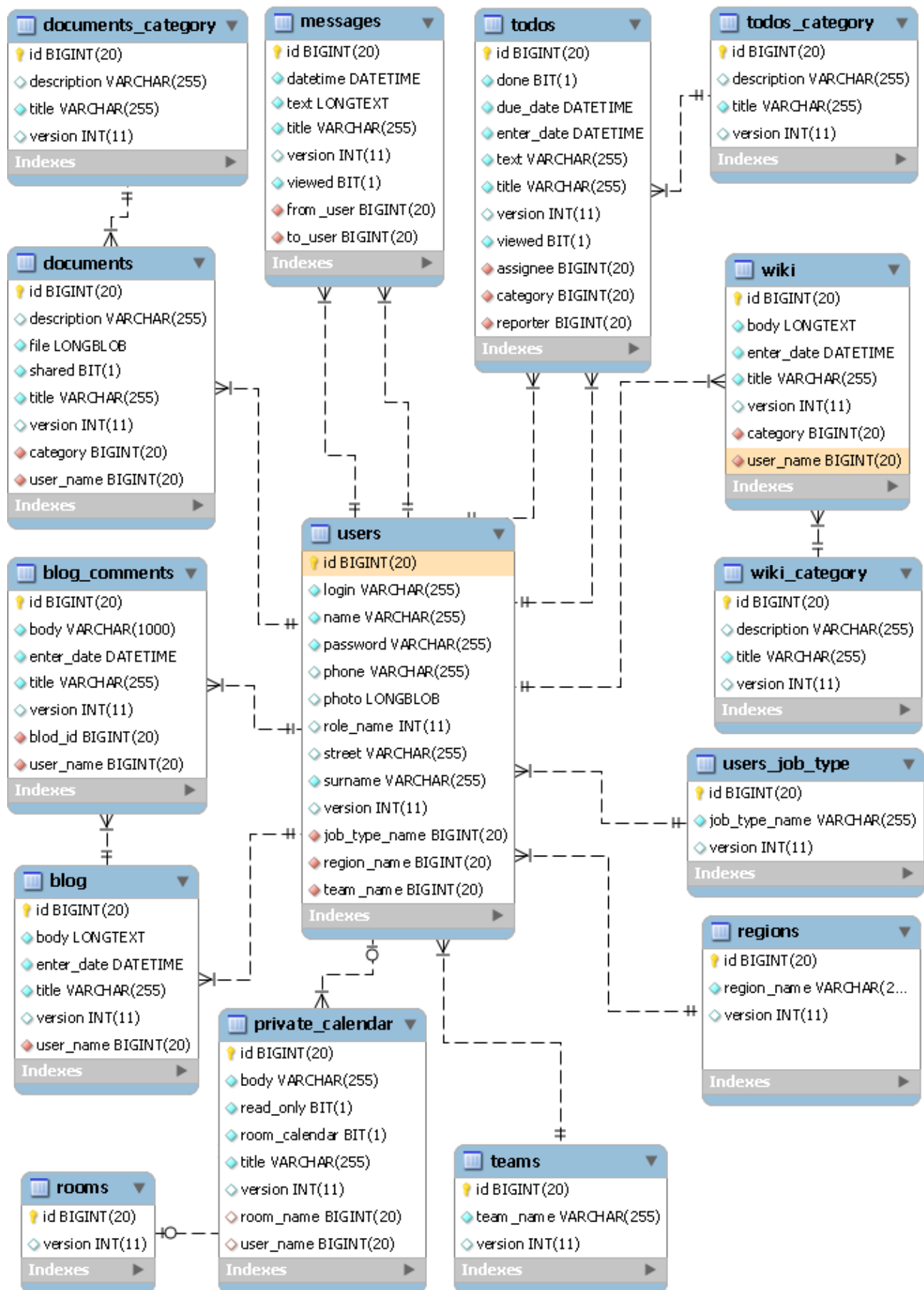


Рисунок 2.5 – Загальна схема структури бази даних

Макет інтерфейсу будується в звичайному графічному редакторі. Головною метою даної розробки є максимальне відображення всіх графічних властивостей майбутнього робочого веб-сайту. Макет майбутнього сайту, зокрема головної сторінки зображено на рисунку 2.6.



Рисунок 2.6 – Макет майбутнього сайту

З рисунку 2.6, де зображено макет сайту добре видно, що навігаційне меню розміщено зверху сайту. Дане меню повинно бути завжди доступне користувачу, тому воно завжди буде рухатися, тобто при будь якому положенні сторінки, не важливо чи внизу чи на верху, ця панель буде завжди доступна і рухати за користувачем. На ній варто розмістити швидкі навігаційні кнопки, такі як: швидке створення завдання чи нотатки.

Зліва від всього розміщено навігаційну панель. Саме на цій панелі розміщуються всі доступні користувачеві на веб-проекті пункти навігації. В залежності від того де в даний момент перебуває користувач, слід підсвічувати поточне меню.

Панель користувача розміщується справа на навігаційній панелі. Тут повинна розмістися вся інформація для керування поточним користувачем: вихід із

сайту, посилання на профіль користувача, приватні налаштування.

2.4 Структура проекту

Після побудови структури бази даних та загального макету сайту, слід приступити до розробки серверної частини – загальної структури сайту. Весь проект базується на роботі Java EE та поверх фреймворку Spring.

2.4.1 Алгоритм роботи надбудови Java EE – Spring Framework

Архітектура Spring Framework зображена на рисунку 2.7.

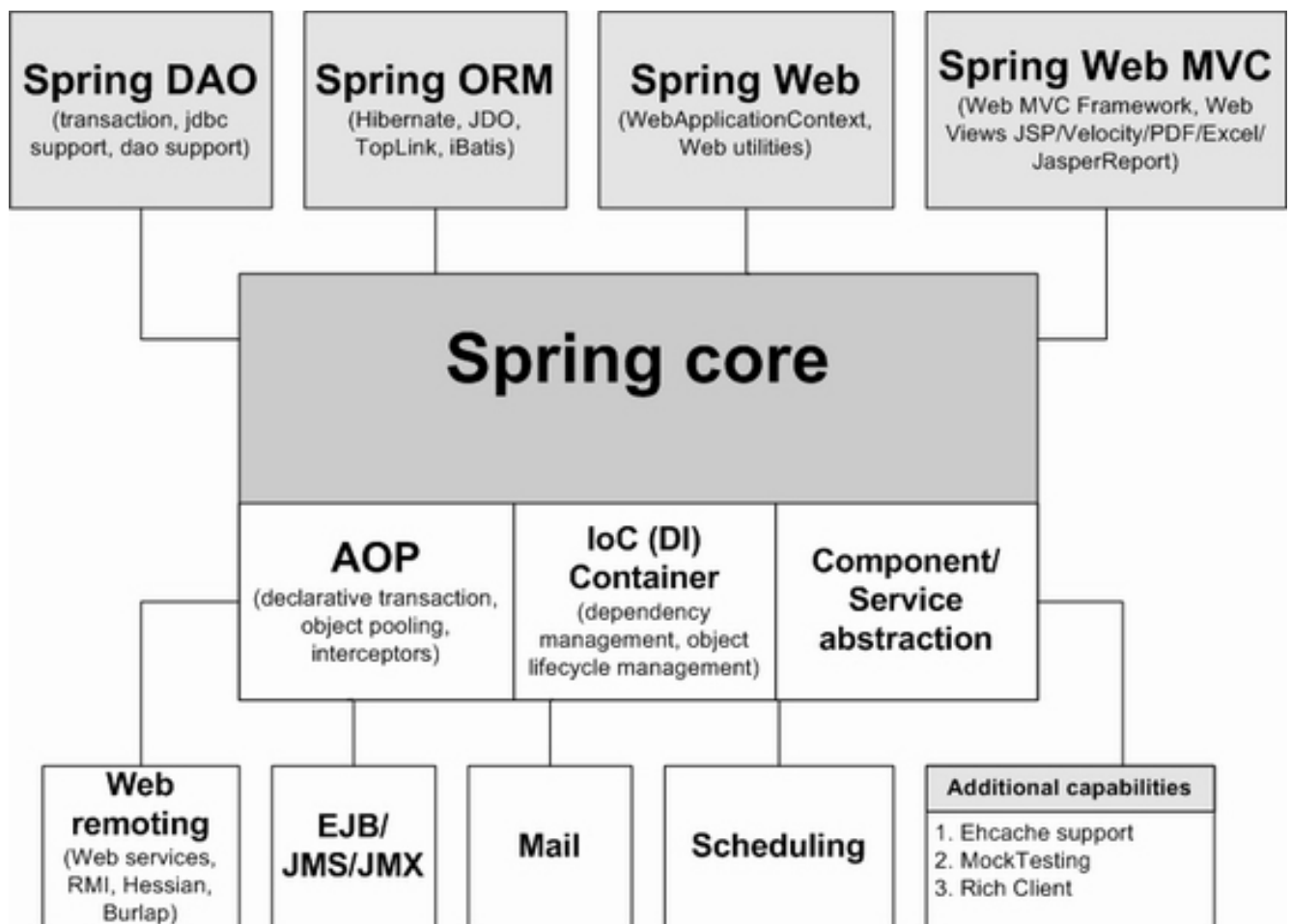


Рисунок 2.7 – Архітектура Spring Framework

Центральною частиною Spring Framework є Inversion of Control контейнер, який надає засоби конфігурування та управління об'єктами Java з допомогою зворотних викликів. Контейнер відповідає за управління життєвим циклом об'єкта:

створення об'єктів, виклик методів ініціалізації та конфігурування об'єктів шляхом зв'язування їх між собою.

Об'єкти створювані контейнером також називаються Керовані об'єкти або Beans. Зазвичай конфігурування контейнера здійснюється шляхом завантаження XML файлів, що містять Визначення Bean-ів і надають інформацію необхідну для створення bean-ів.

Об'єкти можуть бути отримані або за допомогою Пошуку залежності, або Впровадження залежності. Пошук залежності - шаблон проектування, коли зухвалий об'єкт запитує у об'єкта-контейнера екземпляр об'єкта з певним ім'ям або певного типу. Впровадження залежності - шаблон проектування, коли контейнер передає екземпляри об'єктів за їх імені іншим об'єктам або за допомогою конструктора, або властивості, або фабричного методу.

Spring Framework містить в собі наступні програмні модулі:

- Inversion of Control контейнер: конфігурування компонент додатків і управління життєвим циклом Java об'єктів;
- фреймворк аспектно-орієнтованого програмування: працює з функціональністю, яка не може бути реалізована можливостями об'єктно-орієнтованого програмування на Java без втрат;
- фреймворк доступу до даних: працює з системами управління реляційними базами даних на Java платформі використовуючи JDBC і Object-relational mapping кошти забезпечуючи вирішення завдань, які повторюються у великому числі Java-based environments;
- фреймворк управління транзакціями: координація різних API управління транзакціями і інструментарій настроюваного управління транзакціями для об'єктів Java;
- фреймворк Model-view-controller: каркас, заснований на HTTP і сервлетах надає безліч можливостей для розширення і налаштування (customization);
- фреймворк віддаленого доступу: конфігурується передача Java-об'єктів

					ДП.ПЗ 04.00.00.000 ПЗ	Аркуш
						43
Зм.	Арк.	№ докум.	Підпис	Дата		

через мережу в стилі RPC, підтримуюча RMI, CORBA, HTTP-based протоколи, включаючи web-сервіси (SOAP);

- фреймворк аутентифікації та авторизації: конфігурується інструментарій процесів аутентифікації та авторизації, що підтримує багато популярних і стали індустріальними стандартами протоколів, інструментів, практик через дочірній проект Spring Security (раніше відомий як Aсegi);
- фреймворк віддаленого управління: конфігурується уявлення і керування Java об'єктами для локальної або віддаленої конфігурації за допомогою JMX;
- фреймворк роботи з повідомленнями: конфігурується реєстрація об'єктів-слухачів повідомлень для прозорості обробки повідомлень з черги повідомлень за допомогою JMS, поліпшена відправлення повідомлень за стандартом JMS API;
- тестування: каркас, що підтримує класи для написання модульних та інтеграційних тестів.

2.4.2 Діаграма відношень

Діаграма послідовності – в UML, діаграма послідовності котра відображає взаємодії об'єктів впорядкованих за часом. Зокрема, такі діаграми відображають задіяні об'єкти та послідовність відправлених повідомлень.

На діаграмі послідовностей показано у вигляді вертикальних ліній різні процеси або об'єкти, що існують водночас. Надіслані повідомлення зображуються у вигляді горизонтальних ліній, в порядку відправлення.

Визначені стандартом UML 2.0 діаграми послідовностей мають ті ж можливості що і визначені стандартом UML 1.x, та підтримують додаткові можливості зміни стандартного порядку повідомлень.

Діаграма відношень користувача до вибірки із даних бази даних і їх формування зображені на рисунку 2.8

					ДП.ПЗ 04.00.00.000 ПЗ	Аркуш
						44
Зм.	Арк.	№ докум.	Підпис	Дата		

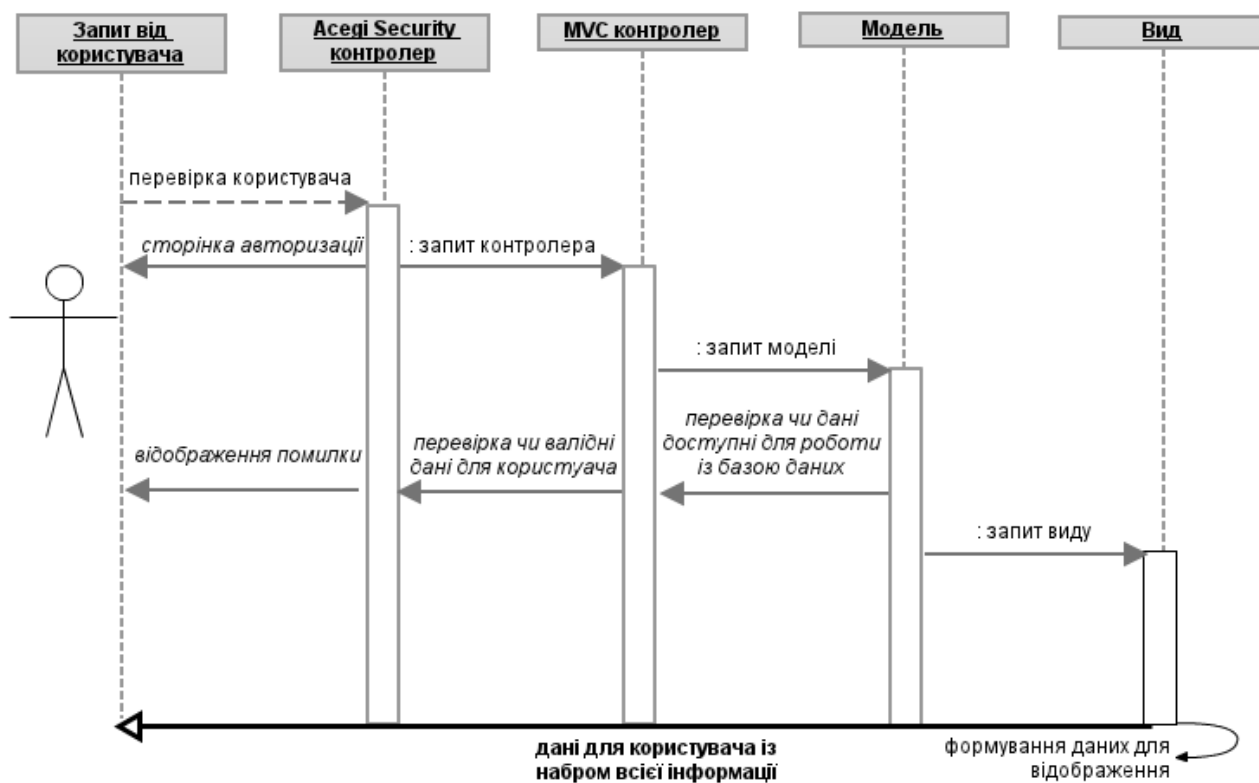


Рисунок 2.8 – Діаграма відношення запиту користувача до роботи сервера

На рисунку 2.8 – діаграми відношень зображено весь процес трансферу даних – починаючи від запиту користувача, і завершуючи поверненням сформованих даних. Спочатку запит попадає на Asegi Security Controller. Після проходження перевірки користувача (детально описано в розділі 2.5) дані від користувача попадають на контролер, звідки всі дані для запиту передаються на модель, яка відповідає за комутацію бази даних та контролера. Після успішних проходжень цих пунктів, на контролері відбувається формування вигляду майбутньої сторінки, яке будується на базі певного виду. Після цих всіх успішних маніпуляцій, сформовані дані повертаються користувачеві, тобто на веб інтерфейс. Якщо на якомусь етапі сталася помилка, чи невірна вибірка, то кожний етап зупиняється і повертається на попередній, аж до користувача із статусом помилки.

2.5 Авторизація і аутентифікація

Основою будь-якої корпоративної системи є можливість використання системи управління користувачами. Тому слід розробити повний цикл взаємодії користувачів. Сюди повинні бути включені наступні важливі аспекти:

- можливість авторизації користувачів;
- ролі користувачів;
- зберігання даних у закодованому вигляді;
- чіткий розподіл прав користувачів;
- легка взаємодія між користувачами;
- можливість інтеграції із іншими сервісами системи.

Кожний працівник (він же користувач системи) повинний мати безперебійний доступ до свого профілю в будь-який час. Авторизація повинна бути реалізована інтуїтивно зрозуміло для кожного користувача і легко доступна. Управління користувачами буде реалізовано через адміністративну панель, доступ до якої будуть мати тільки користувачі певної групи.

2.5.1 Алгоритм авторизації користувачів

Загальний алгоритм авторизації полягає в перевірці даних користувача і повернення від серверу сформованих даних.

Сервер спочатку приймає надіслані від користувача логін та пароль. Логін передається на Acegi Security Controller і перекодовує пароль в шифр за допомогою алгоритму SHA-256. Адже зберігати дані не в шифрованому вигляді досить небезпечно, і у випадку, якщо зловмисник отримає їх, від цих даних не буде користі, оскільки вони шифровані і зворотнього шифру не існує. Шифрування на базі серверу відбувається через менеджер авторизації і налаштовується в конфігураційному файлі із посиланням на Bean із сервісом котрий відповідає за маніпуляцію даних користувача.

```
<authentication-manager alias="authenticationManager">
```

					ДП.ПЗ 04.00.00.000 ПЗ	Архив
						46
Зм.	Арк.	№ докум.	Підпис	Дата		

```

        <authentication-provider user-service-ref="
            UserServiceBean">
            <password-encoder hash="sha-256" />
        </authentication-provider>
    </authentication-manager>
    <beans:bean id="UserServiceBean" class="com.diploma.ccms.service.
        UserService" />

```

Слід виділити наступні групи користувачів:

- Адміністратори;
- Відділ кадрів;
- Користувач без особливих прав доступу.

Кожній групі повинний бути забезпечений доступ до певної категорії. Це можливо реалізувати за допомогою інтрацепторів. В разі доступу користувача до недозволеної категорії, буде повернена від сервера помилка.

```

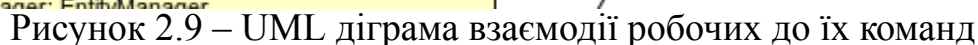
<http auto-config="true" use-expressions="true">
<form-login login-processing-url="/resources/
    j_spring_security_check" login-page="/login" authentication-
    failure-url="/login?login_error=t" />
<logout logout-url="/resources/j_spring_security_logout" />
<!-- Configure these elements to secure URIs in your application
-->
<intercept-url pattern="/admin/**" access="hasRole('ROLE_ADMIN') "
    />
<intercept-url pattern="/login/**" access="permitAll" />
<intercept-url pattern="/resources/**" access="permitAll" />
<intercept-url pattern="/**" access="isAuthenticated()" />
</http>

```

2.6 Діаграма класів

Діаграма класів – статичне представлення структури моделі. Відображає статичні (декларативні) елементи, такі як: класи, типи даних, їх зміст та відношення. Діаграма класів, також, може містити позначення для пакетів та може містити

					ДП.ПЗ 04.00.00.000 ПЗ	Аркуш
Зм.	Арк.	№ докум.	Підпис	Дата		47



ДП.ПЗ 04.00.00.000 ПЗ

чної структури моделі системи в термінології класів об'єктно-орієнтованого програмування. На цій діаграмі показують класи, інтерфейси, об'єкти й кооперації, а також їхні відносини.

Взаємозв'язок класу «Команд» і загальних інтерфейсів зображено на рисунку 2.9, решта класів працює за таким принципом. Кожний клас домену посиляється на контролер управління та взаємозалежить від інших класів. Інтерфейси забезпечують уніфікації і взаємодію класів.

					ДП.ПЗ 04.00.00.000 ПЗ	Аркуш
						49
Зм.	Арк.	№ докум.	Підпис	Дата		

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ДЛЯ УПРАВЛІННЯ КОРИСТУВАЧАМИ, ДОКУМЕНТАМИ, ЗАВДАННЯМИ І МОЖЛИВОСТІ СПІЛЬНОЇ РОБОТИ

3.1 Реалізація роботи бази даних

Загальна структура бази даних зображено на рисунку 2.5. Весь код для створення бази даних реалізовано мовою SQL, за допомогою запитів до БД.

Зовнішні посилання створено за допомогою команди Reference. Для прикладу код для створення таблиці користувачів (worker), котра має чотири зовнішні ключі які посилаються на таблиці команди (team), роль користувача (role_name), тип посади (job_type_name) та регіон (region_name).

Відповідно таким чином і реалізовано всі решта таблиць. Код для створення таблиці користувачів:

```
DROP TABLE IF EXISTS `worker`;  
  
/*!40101 SET @saved_cs_client      = @@character_set_client */;  
/*!40101 SET character_set_client = utf8 */;  
  
CREATE TABLE `worker` (  
    `id` bigint(20) NOT NULL AUTO_INCREMENT,  
    `birthday` date DEFAULT NULL,  
    `date_hire` date DEFAULT NULL,  
    `login` varchar(255) NOT NULL,  
    `name` varchar(255) NOT NULL,  
    `pass` varchar(255) NOT NULL,  
    `phone` varchar(255) DEFAULT NULL,  
    `photo` longblob,  
    `private_mail` varchar(255) DEFAULT NULL,  
    `street` varchar(255) DEFAULT NULL,  
    `surname` varchar(255) NOT NULL,  
    `version` int(11) DEFAULT NULL,  
    `job_type_name` bigint(20) NOT NULL,  
    `region_name` bigint(20) NOT NULL,  
    `role_name` bigint(20) DEFAULT NULL,  
    `team_name` bigint(20) NOT NULL,
```

```

`mobile` varchar(255) DEFAULT NULL,
PRIMARY KEY (`id`),
UNIQUE KEY `login` (`login`),
KEY `FKD162537E30271785` (`team_name`),
KEY `FKD162537EB0DD720C` (`job_type_name`),
KEY `FKD162537E520C2F83` (`role_name`),
KEY `FKD162537EBDECCB25` (`region_name`),
CONSTRAINT `FKD162537E30271785` FOREIGN KEY (`team_name`)
REFERENCES `team` (`id`),
CONSTRAINT `FKD162537E520C2F83` FOREIGN KEY (`role_name`)
REFERENCES `worker_role` (`id`),
CONSTRAINT `FKD162537EB0DD720C` FOREIGN KEY (`job_type_name`)
REFERENCES `worker_job_type` (`id`),
CONSTRAINT `FKD162537EBDECCB25` FOREIGN KEY (`region_name`)
REFERENCES `region` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

```

Посилання на інші таблиці на веб інтерфейсі реалізовано за допомогою випадających списків – це дає можливість забезпечити введення вірних даних і допомагає відобразити вже існуючі в базі даних записи. Для прикладу візьмемо форму для створення нового користувача та вибору регіону, що зображено на рисунку 3.1.

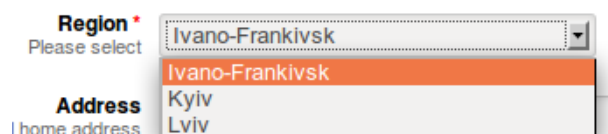


Рисунок 3.1 – Вибір регіону при створенні користувача

3.2 Реалізація веб інтерфейсу

Веб інтерфейс користувача повинний бути зручний та інтуїтивно зрозумілий кожному користувачеві, тому його було реалізовано в легких тонах та зручно розташовано всі навігаційні елементи.

					ДП.ПЗ 04.00.00.000 ПЗ	Архив
Зм.	Арк.	№ докум.	Підпис	Дата		51

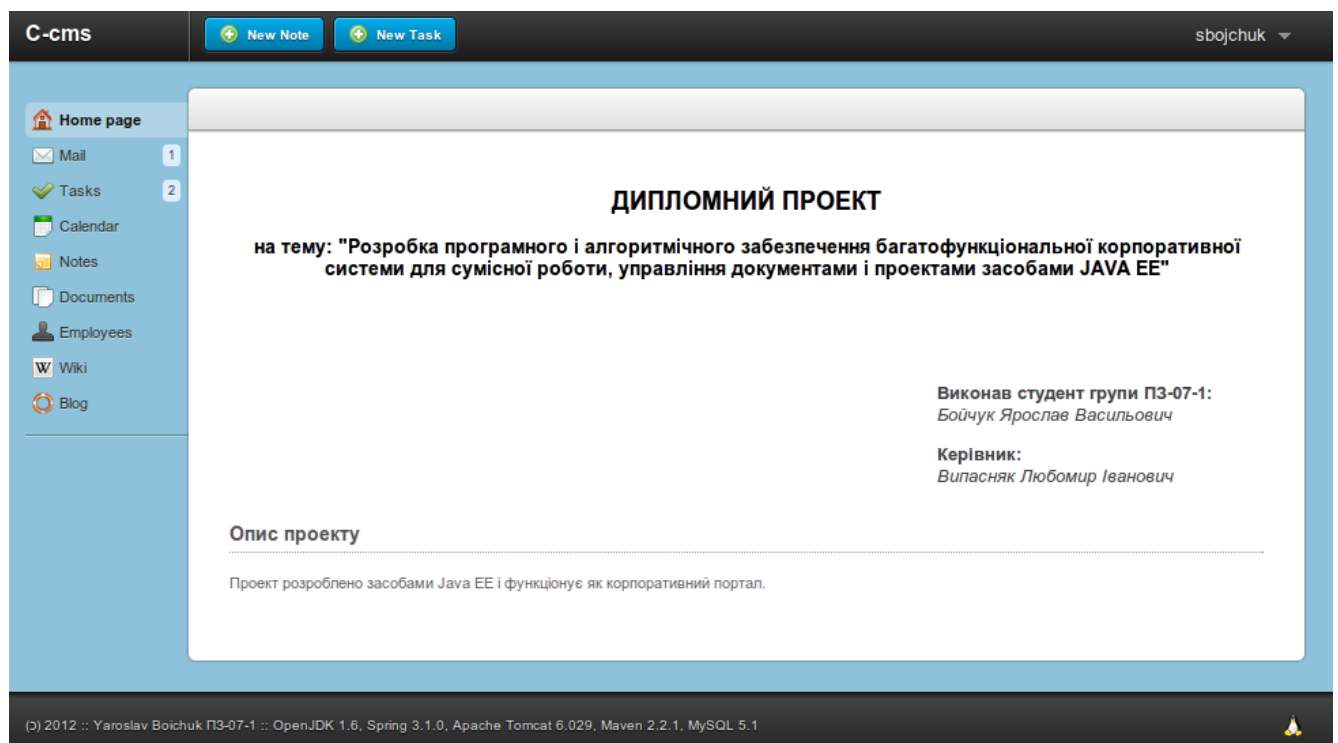


Рисунок 3.2 – Загальний інтерфейс програмного продукту

Весь інтерфейс сайту поділяється на три основні компоненти: головний блок, навігаційна панель, головне меню. Для зручності розробки цих компонентів використано Apache Tiles, що дає змогу поділяти код на логічні одиниці.

```
<body>
<div id="wrapper">
  <header>
    <tiles:insertAttribute name="header" ignore="true" />
  </header>
  <section>
    <div class="container_8 clearfix">
      <tiles:insertAttribute name="menu" ignore="true" />
      <tiles:insertAttribute name="body" />
    </div>
    <div id="push"><!-- --></div>
  </section>
</div>
<footer>
  <tiles:insertAttribute name="footer" ignore="true" />
</footer>
```

					ДП.ПЗ 04.00.00.000 ПЗ	Архив
Зм.	Арк.	№ докум.	Підпис	Дата		52

```

<div class="apple_overlay black" id="overlay">
  <a class="close"></a>
  <iframe class="contentWrap" style="width: 100%; height: 500px
    "></iframe>
</div>
<div style="display: none; position: absolute;" id="calroot">
  <div id="calhead">
    <a id="calprev"></a>
    <div id="caltitle"></div>
    <a id="calnext"></a>
  </div>
  <div id="calbody">
    <div id="caldays">
      <span>Sun</span><span>Mon</span><span>Tue</span><span>Wed</
        span><span>Thu</span><span>Fri</span><span>Sat</span>
    </div>
    <div id="calweeks">
      <!-- -->
    </div>
  </div>
</div>
</body>

```

3.2.1 Навігаційна панель

На навігаційній панелі розташовані елементи швидкого доступу до завдань та задач. З легкістю можна додати будь-яке завдання, при чому вибрати заголовок завдання, детальний опис та кінцевий час виконання (рисунки 3.3);

Для зручності навігаційна панель рухається разом із прокруткою сторінки, тобто якщо навіть користувач перейде в низ сторінки, то йому панель буде завжди доступна – це зроблено для простоти і швидкості доступу до створення нової нотатки та завдання.

Справа на навігаційній панелі розташовано меню користувача. Тут знаходяться кнопки переходу на профіль робочого та кнопка виходу із сайту (рисунки 3.4, 3.5);

					ДП.ПЗ 04.00.00.000 ПЗ	Архив
Зм.	Арк.	№ докум.	Підпис	Дата		53

Рисунок 3.3 – Приклад створення завдання із навігаційної панелі

3.4). Після виходу всі дані, які збережені в сесії будуть видалені.

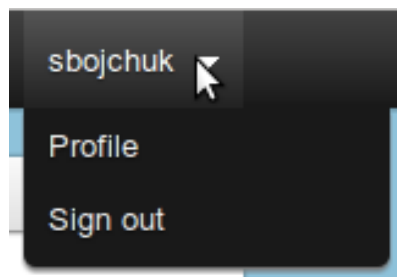


Рисунок 3.4 – Панель користувача

3.2.2 Головне меню

Навігація по веб ресурсу реалізована за допомогою головного меню. В головному меню відображаються всі доступні на сайті навігаційні посилання:

- головна сторінка;
- корпоративна пошта;
- завдання;
- календар;
- нотатки;
- документи;
- список робочих;
- корпоративна вікі;

– блог.

При переході на будь-яке меню, воно зразу підсвічується – це зроблено для зручності користувачеві, щоб було зразу видно де він знаходиться в даний момент часу. Програмно це відбувається за допомогою передачі з контролера в модель атрибута із назвою меню:

```
uiModel.addAttribute("menu", "NOTE");
```

Потім в JSP вигляді головного меню відбувається перевірка на значення поточного меню, і якщо воно сходиться із атрибутом «menu» то додається CSS клас «active» (рисунок 3.5):

```
<c:choose>
<c:when test="${menu eq 'NOTE' }"><li class="active"><a class="
    nav-icon icon-note" href="/ccms/notes">Notes</a></li></c:when>
<c:otherwise><li><a class="nav-icon icon-note" href="/ccms/notes
    ">Notes</a></li></c:otherwise>
</c:choose>
```

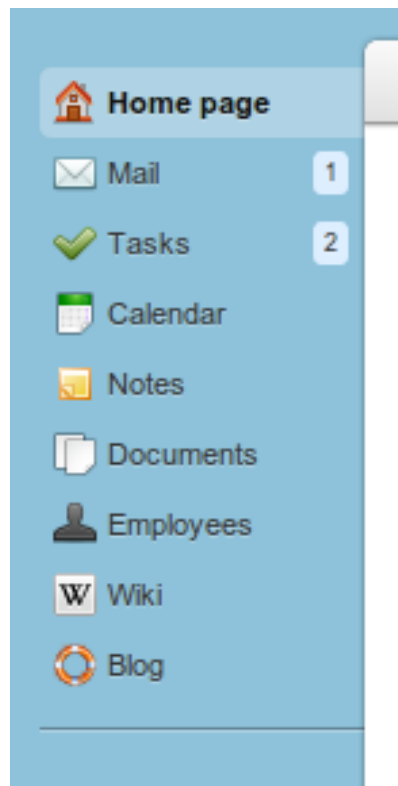


Рисунок 3.5 – Навігаційне меню порталу

Відповідно до переходу на певний пункт, відбувається запит контролеру MVC, і вибірка даних із бази даних через контролер з подальшою передачею на вигляд. Для прикладу запит для запису в БД та перевірка на валідність даних:

```
@RequestMapping(method = RequestMethod.POST, produces = "text/html")
public String create(@Valid Note note, BindingResult
    bindingResult, Model uiModel, HttpServletRequest
    httpRequest) {
    if (bindingResult.hasErrors()) {
        populateEditForm(uiModel, note);
        uiModel.addAttribute("menu", "NOTE");
        return "redirect:/notes";
    }
    uiModel.asMap().clear();
    note.setAuthor(Worker.getPrincipal());
    note.setDatetime(new Date());
    note.persist();
    uiModel.addAttribute("menu", "NOTE");
    return "redirect:/notes";
}
```

3.3 Робота із даними

Для мапінгу даних із форми до бази даних використовується JPA із Hibernate фреймворком поверх нього. Для кожної форми створюється певний домен (по своїй суті persistence bean), котрий за допомогою анотацій із пакету Javaх дає змогу переносити об'єкти Java в базу даних (за допомогою використання мови запитів Hibernate Query Language). Для прикладу bean для запису нотаток в базу даних:

```
@Configurable
@Entity
public class Note {
```

					ДП.ПЗ 04.00.00.000 ПЗ	Аркуш
						56
Зм.	Арк.	№ докум.	Підпис	Дата		

```

@NotNull
private String title;

@NotNull
@Size(max = 1000000)
private String text;

@Temporal(TemporalType.TIMESTAMP)
@DateTimeFormat(style = "M-")
private Date datetime;

@ManyToOne
private Worker author;

public static TypedQuery<Note> findNotesByAuthorEquals(Worker
author) {
    if (author == null)
        throw new IllegalArgumentException("The author argument
        is required");
    EntityManager em = Note.entityManager();
    TypedQuery<Note> q = em.createQuery("SELECT o FROM Note AS o
        WHERE o.author = :author ORDER by o.id DESC", Note.class);
    q.setParameter("author", author);
    return q;
}

public String getTitle() {
    return this.title;
}

public void setTitle(String title) {
    this.title = title;
}

```

В вище наведеному коді кожне поле має свій метод на getter та setter, що дає змогу в вибірці даних, та статичний метод «findNotesByAuthorEquals» для по-

					ДП.ПЗ 04.00.00.000 ПЗ	Архив
						57
Зм.	Арк.	№ докум.	Підпис	Дата		

шуку повідомлень даного автора, що дає змогу в будь-якому місці здійснити операції щодо пошуку цих повідомлень.

3.4 Категорії порталу

3.4.1 Авторизація

Сторінка авторизації пропонує користувачеві ввести свій логін та пароль (рисунки 3.6) та у випадку неправильних даних буде відображена помилка (рисунки 3.7)

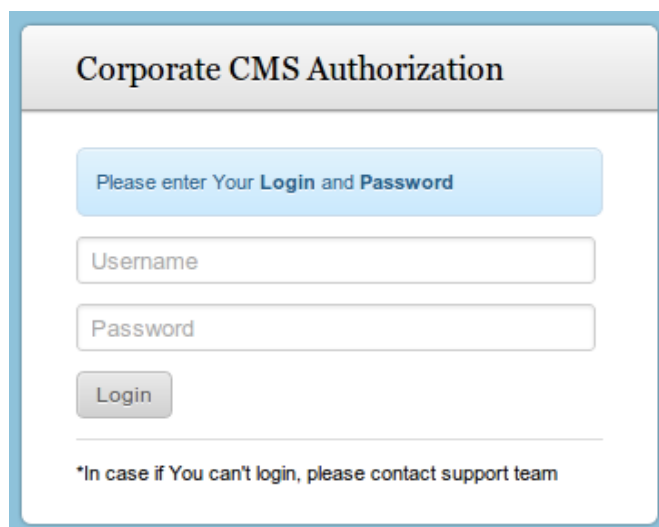


Рисунок 3.6 – Форма авторизації користувача

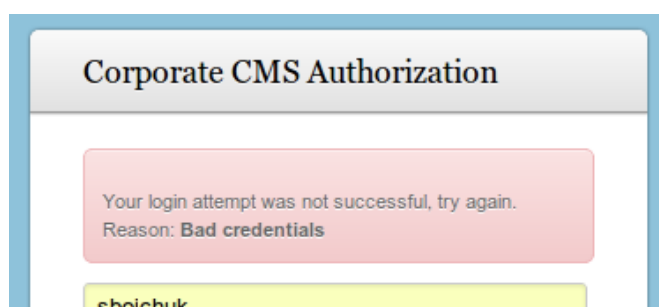


Рисунок 3.7 – Помилка авторизації користувача

Якщо введено вірні дані, то відбувається запис нової сесії в пам'ять та перенаправлення користувача на головну сторінку, або на сторінку із якої прийшов користувач.

3.4.2 Корпоративна пошта

Всі отримані листи користувач може переглянути в пункті пошта. Напроти меню «пошта» відображається кількість не прочитаних повідомлень (рисунок 3.8).

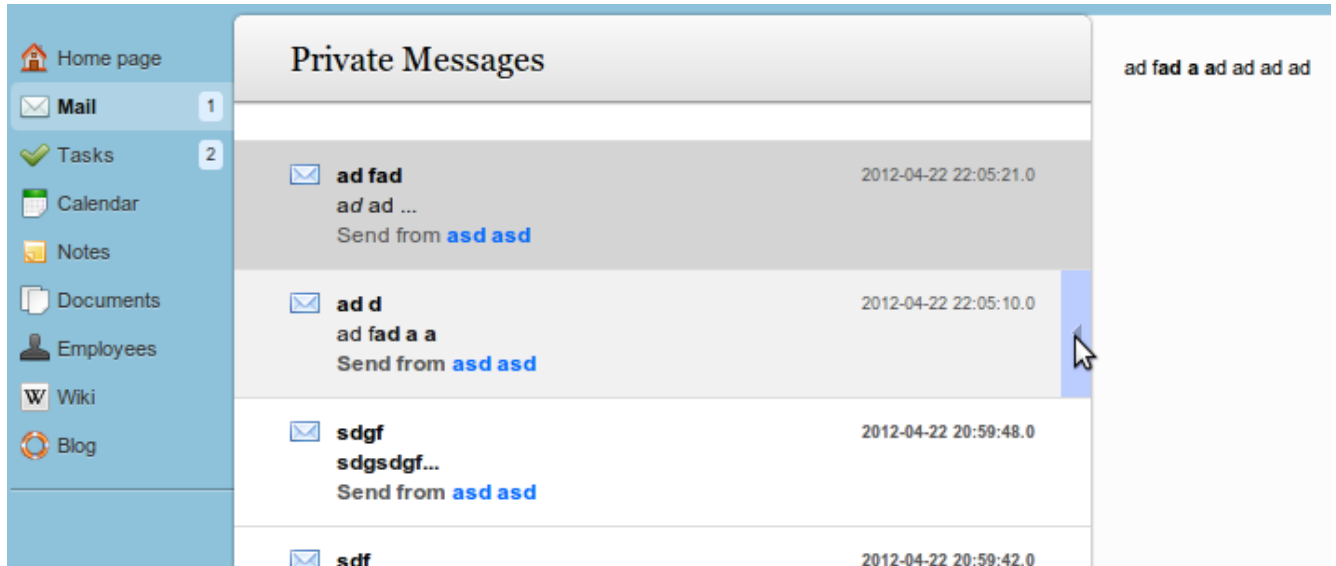


Рисунок 3.8 – Сторінка зі списком повідомлень





Над кожним повідомленням показано тему повідомлення. Головний текст повідомлення обрізаний, проте коли клікнути на повідомлення, збоку висунеться панель із повним описом повідомлення. Також під повідомленням показано час відправлення повідомлення та відправник повідомлення. При кліку на відправника, відбудеться перехід на його персональну сторінку. Кожне не прочитане повідомлення виділяється сірим кольором, для того щоб легше було його знайти, і при детальному перегляді його, колір забереться, і кількість повідомлень, що показуються біля меню – буде зменшено.

Відправлення повідомлення можливе із персональної сторінки кожного користувача. Після переходу на персональну сторінку, слід надрукувати тему повідомлення та саме повідомлення (рисунок 3.9).

Зразу також доступний wysiwyg редактор і live перегляд повідомлення яке друкується. Доставка повідомлення відбувається миттєво, адже використовується локальний сервер бази даних.

Send private message to asd asd

Title * Нове повідомлення

B *I* ~~S~~ |   |  

Текст нового повідомлення

Send message

Рисунок 3.9 – Форма для відправлення повідомлень

3.4.3 Календар

Персональний календар дає змогу показати всі занесені до нього нотатки та записи. Перегляд даних можливий у трьох проміжних режимах: на місяць, на тиждень (рисунок 3.10) та на день.

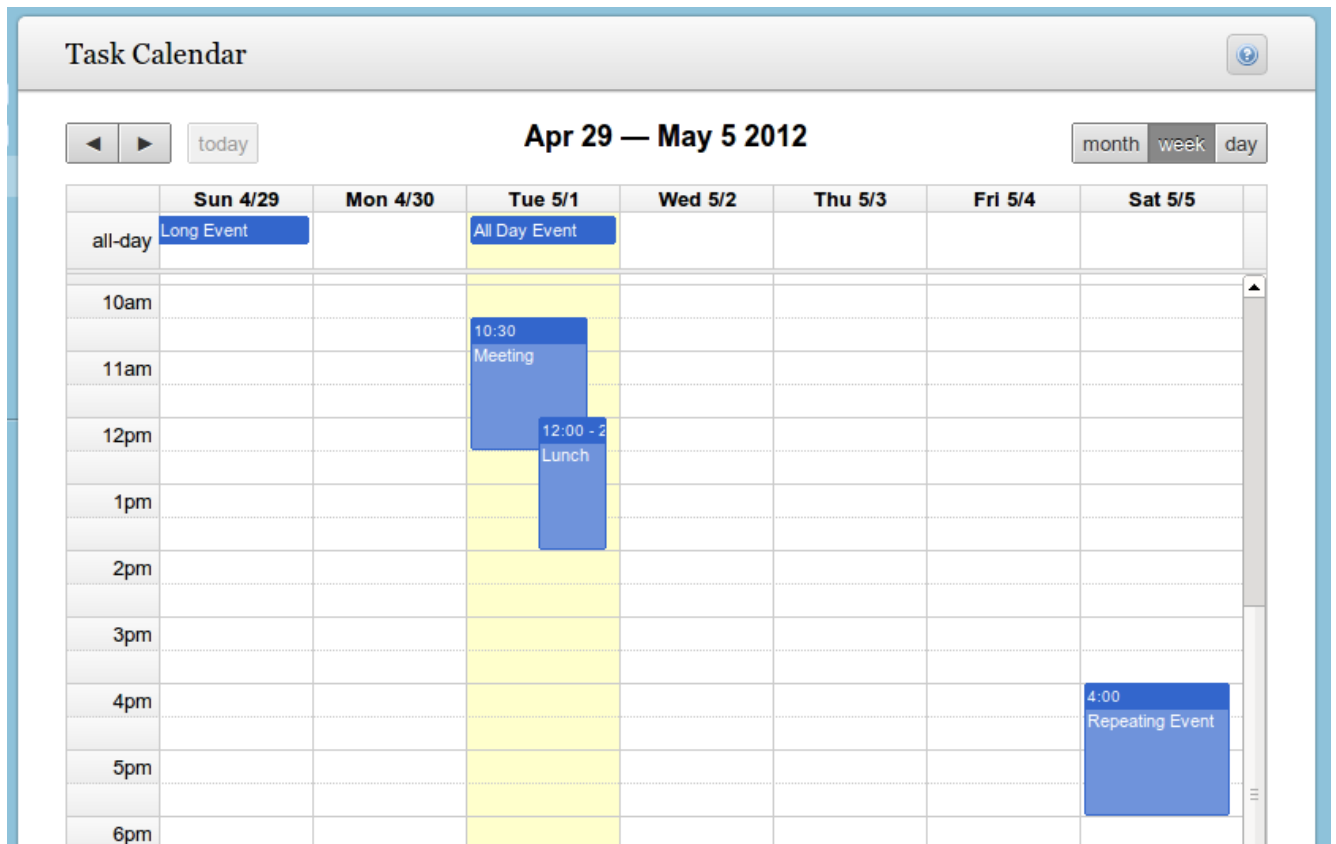


Рисунок 3.10 – Завдання на тиждень

Завдання можуть бути додані як на певний проміжний період, так і на цілий день. Для маніпуляції записів в календарі використана технологія drag & drop від jQuery.

3.4.4 Завдання і задачі

Категорія задач створена для збереження своїх задач (3.3) і можливістю їх перегляду в майбутньому. Це дає змогу всі свої важливі завдання тримати в одному місці (3.11).

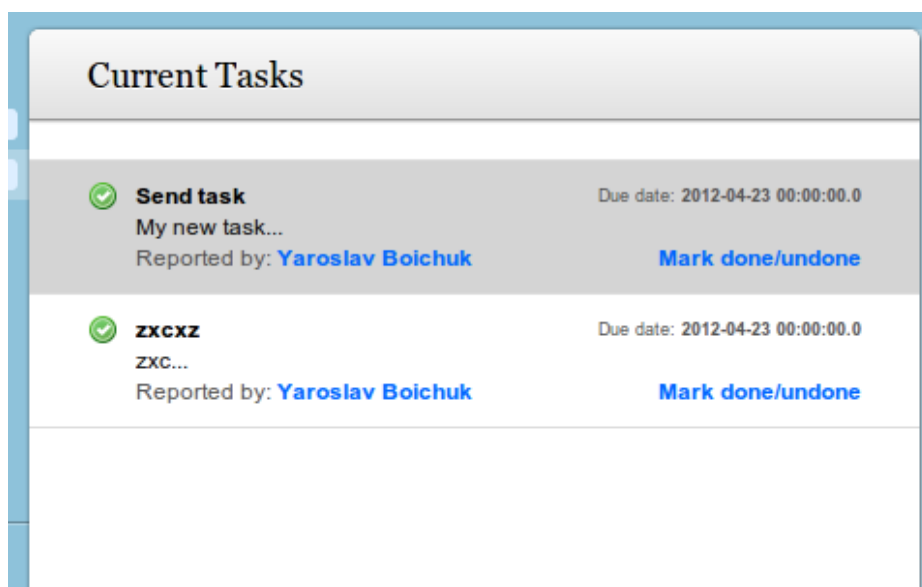


Рисунок 3.11 – Поточні завдання

Кожне завдання яке не виконано ще, позначається аналогічно до непрочитаного повідомлення – сірим кольором, це дає змогу зразу побачити всі поточні завдання. Біля кожного завдання вказано хто створив дане завдання та кінцевий час його виконання. Також в головному меню навпроти пункту «завдання» вказується кількість невиконаних на даний момент завдань. Також кожне завдання може бути позначене як виконане або ж невиконане.

3.4.5 Користувачі

Список всіх користувачів відображається в таблиці із деяким набором полів. Для переглядаючого доступні певні маніпуляції зі списком, такі як сортування

					ДП.ПЗ 04.00.00.000 ПЗ	Архив
Зм.	Арк.	№ докум.	Підпис	Дата		61

та посторінкова навігація (рисунок 3.12). А у випадку, якщо користувач наділений правами адміністратора – то має право на створення нового користувача (рисунок 3.13).

List of Employees

Login	Surname	Name	Phone	Job type	
asd	asd	asd		Junior Java Developer	
sboichuk	Boichuk	Yaroslav	1470	Junior Java Developer	

First

<

1

>

Last

NOTE

INFO: You can sort and navigate through the list

Рисунок 3.12 – Список користувачів із можливістю сортування

Після переходу на сторінку користувача, у його профайлі буде відображена вся детально інформація (рисунок 3.14)

Якщо при введенні не валідних даних, або ж залишити незаповненим обов’язкове поле – то буде повідомлена відповідна помилка, і дані не потраплять на перевірку на сервер. Якщо ж зломиснику вдасться все ж таки обійти перевірку форми, то сам сервер не пустить додати не валідні дані до бази даних, оскільки всі дані перевіряються другий раз за допомогою binding result об’єкта та анотації @Valid:

```

public String update(@Valid Worker worker, BindingResult
    bindingResult, Model uiModel, HttpServletRequest
    httpRequest) {
    if (bindingResult.hasErrors()) {
        populateEditForm(uiModel, worker);
        uiModel.addAttribute("menu", "WORKER");
        return "workers/update";
    }
}

```

Create new employee

Login *

Enter login

Password *

Enter password

Name *

Enter name

Surname *

Enter surname

Mobile phone number

Private phone number

Email

A valid email address

Region *

Please select

Ivano-Frankivsk ▼

Address

Enter full home address

Birthdate

mm/dd/yyyy

Hire Date

mm/dd/yyyy

Employee role *

Please select role

ROLE_ADMIN ▼

Employee related team *

Please select working team

GPD ▼

Job type *

Like QA, Developer, etc..

Junior Java Developer ▼

Рисунок 3.13 – Створення нового користувача

Якщо після проходження валідації є допущені помилка то дані назад «повертаються» на форму і відображається помилка. В іншому випадку, дані передадуться на модель та відбудеться запис у базу даних

```
uiModel.asMap().clear();
worker.merge();
```

3.4.6 Документи

Дана категорія призначена для зберігання документів для їх спільного використання, для прикладу це можуть бути презентації, облікові документи чи про-

					ДП.ПЗ 04.00.00.000 ПЗ	Архив
Зм.	Арк.	№ докум.	Підпис	Дата		63

Yaroslav Boichuk
Junior Java Developer

Yaroslav's contact information

- sboichuk@gmail.com (email)
- 050-194-56-57 (mobile)
- 1470 (work)

Additional info

- 2007-07-09 (birthday)
- 2011-11-10 (hire date)
- Ivano-Frankivsk (living region)
- 25 Serpnya 9/83 (home address)

Send private message to Yaroslav Boichuk

Title:

B I S |

Send message

Рисунок 3.14 – Профайл користувача

сто інші нотатки. Для додавання доступні три категорії: презентації, текстові документи та таблиці. При завантаженні нового документа на портал, слід вказати категорію в котру повинен потрапити документ, та вказати чи документ призначений для загально використання чи тільки для персонального.

Всі загальні документи доступні для завантаження та можливістю подальшого перегляду.

3.4.7 Корпоративна wiki

Корпоративна wiki в основному призначена для розповсюдження цікавої інформації між користувачами та являє собою єдине центральне сховище з можливістю будь-якої маніпуляції документами. Кожна стаття має свою певну категорію – що спрощує подальшу навігацію та пошук.

3.4.8 Корпоративний блог

Корпоративний блог має спільні риси із wiki, проте додати інформацію в нього тільки має право адміністратор та наділені такими правами групи користувачів. Основна ціль блогу – це швидше інформування робочих про новини порталу та компанії.

					ДП.ПЗ 04.00.00.000 ПЗ	Аркуш
Зм.	Арк.	№ докум.	Підпис	Дата		64

4 ЕКОНОМІЧНА ДОЦІЛЬНІСТЬ ВИКОРИСТАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Економічна доцільність розробки програмного забезпечення та його впровадження

В даному проекті необхідно реалізувати корпоративну систему для спільної і одночасної роботи працівників деякої компанії. В ньому буде реалізовано систему обміну повідомленнями, управління задачами і завданнями, зручне ведення корпоративного календаря, спільна робота над документами різного типу (текстові документи, презентації тощо), система корпоративної вікі та блог.

Як відомо, кожний продукт, який розробляється сьогодні з подальшим впровадженням на ринок потребує обґрунтування з економічної точки зору, а саме доцільності даного продукту. Дане обґрунтування необхідне для того, щоб вчасно припинити (при втраті актуальності або надмірних витратах) розробку або здійснити необхідні інвестування в проект для забезпечення необхідними програмними або апаратними засобами розробників з метою одержання очікуваних результатів. Економічний ефект розробленого продукту визначається на основі економічних показників, які дають можливість прогнозувати результат від впровадження даного програмного продукту.

Існує багато методів визначення економічних показників доцільності впровадження та використання будь якого програмного продукту. Враховуючи інтенсивне впровадження комп'ютерної техніки в корпоративній сфері, на сьогодні такий аналіз є невід'ємною частиною попереднього аналізу аналогічних робіт, оскільки саме результат економічних показників доцільності дозволяє визначити доцільність розробки програмного продукту.

В даній роботі проводиться розрахунок економічних показників та аналіз всієї роботи по розробці корпоративної системи.

4.2 Побудова мережевого графа

Мережевий граф є основним плановим документом в системі мережевого планування і керування, що являє собою інформаційно-динамічну модель, в якій зображуються взаємозв'язки і результати всіх робіт, необхідних для досягнення кінцевої мети розробки, тобто мережевий граф - це наочне відображення плану робіт.

В мережевому графі детально чи укрупнено показано, що, в якій послідовності, коли, за який час, для чого необхідно виконати, щоб забезпечити закінчення всіх робіт не пізніше заданого, директивного терміну.

Порядок побудови мережевих графів визначається прийнятою технологією і організацією робіт. Мережеві графи тільки відображають існуючу або проєктовану черговість і взаємозв'язок виконання робіт.

По кожній роботі необхідно враховувати:

- які роботи повинні бути завершені раніше, ніж почнеться дана робота;
- які роботи можуть початись після завершення даної роботи;
- які інші роботи повинні виконуватись одночасно з виконанням даної роботи.

Аналізуючи мережевий граф можна виділити його головні елементи: події і роботи. Розглянемо детальніше значення термінів:

- подія - це стан, момент досягнення проміжної або кінцевої цілі розробки.
- робота - це розтягнений в часі процес, необхідний для здійснення події.

Кожна робота має попередню подію і закінчується визначеною подією.

На мережевих графах подія відображається колом, а робота — стрілкою. До основних параметрів мережевого графа відносяться: критичний шлях, резерви часу подій. Ці параметри є вихідними для одержання ряду додаткових характеристик, а також для аналізу мережі чи для аналізу складеного плану розробки.

Резерв часу події - це такий проміжок часу, на який може бути відкладене здійснення цієї події без порушення термінів завершення розробки в цілому.

					<i>ДП.ПЗ 04.00.00.000 ПЗ</i>	Аркуш
						66
Зм.	Арк.	№ докум.	Підпис	Дата		

Резерви часу існують в мережевому графі в усіх випадках, коли існує більш ніж один шлях різної тривалості.

Резерв часу події K визначається як різниця між пізнім T_p і раннім T_r термінами завершення події за формулою

$$K = \frac{T_p}{T_r} \quad (4.1)$$

Найбільш пізній з допустимих термінів T_p - це такий термін здійснення події, перевищення якого викличе аналогічну затримку завершальної події. Іншими словами, якщо подія настанула в момент T_p , вона потрапила в критичну зону і наступні за нею роботи повинні знаходитись під таким же контролем як і роботи критичного шляху.

Найбільш ранній з можливих термінів здійснення події T_r — це термін необхідний для виконання всіх робіт, що передують цій події. Цей час знаходиться шляхом вибору максимального значення із тривалості всіх шляхів, що приводять до даної події.

Вихідні дані мережевого графа представлені в таблицях 4.1 та 4.2.

Табл. 4.1 – Події мережевого графа

№ події	Подія
0	Отримання завдання на дипломне проектування
1	Аналіз проблеми дипломного проектування
2	Ознайомлення з літературою на задану тему
3	Пошук інформації в мережі INTERNET
4	Підбір необхідних джерел інформації
5	Аналіз підбраного матеріалу
6	Визначення задач, які виникають при розробці
7	Розгляд існуючих способів розробки корпоративних систем
8	Аналіз існуючих способів розробки
9	Пошук існуючих корпоративних систем
10	Аналіз знайдених аналогів та їх функціональності
11	Розробка структури алгоритму

№ події	Подія
12	Розробка алгоритму програми
13	Вибір серверної платформи для реалізації завдання
14	Визначення основних та допоміжних програмних модулів
15	Реалізація програмних модулів в середовищі програмування
16	Попереднє налагодження програмних модулів
17	Остаточне налагодження програми
18	Тестування програмного продукту
19	Визначення економічної доцільності використання програми
20	Завершення роботи АБВГ

Табл. 4.2 – Роботи мережевого графа

Номери робіт	Роботи	Тривалість, дні
0-1	Аналіз завдання дипломного проекту	2
1-2	Огляд літератури	3
1-3	Огляд інформації в INTERNET	3
3-4	Робота з підібраним матеріалом з INTERNET	4
2-4	Робота з підібраним технічним матеріалом	3
4-5	Аналіз вимог до системи та її функціональності	4
5-6	Виділення та групування задач розробки	3
6-7	Пошук та розгляд існуючих методів реалізації	7
7-8	Аналіз та компонування існуючих способів розробки	4
8-9	Пошук аналогів розробленої системи	5
8-10	Аналіз аналогів розробленої системи	4
9-11	Завершення аналізу аналогів та вибір способу реалізації	2
11-13	Розробка структури алгоритму	5
10-12	Складання алгоритму програми та його аналіз	2
12-13	Розробка структури програми	7
13-14	Уточнення виду вхідних даних для програми	5
14-15	Аналіз інструментальних засобів створення програми	2
15-16	Підбір середовища програмування	3
16-17	Написання коду модулів програми	14
17-18	Налагодження всіх модулів програми	7

Номери робіт	Роботи	Тривалість, дні
18-19	Завершення етапу налагодження програми	3
19-20	Тест програми та аналіз результатів тестування	2
20-21	Аналіз економічних показників	5
21-22	Завершення роботи	14

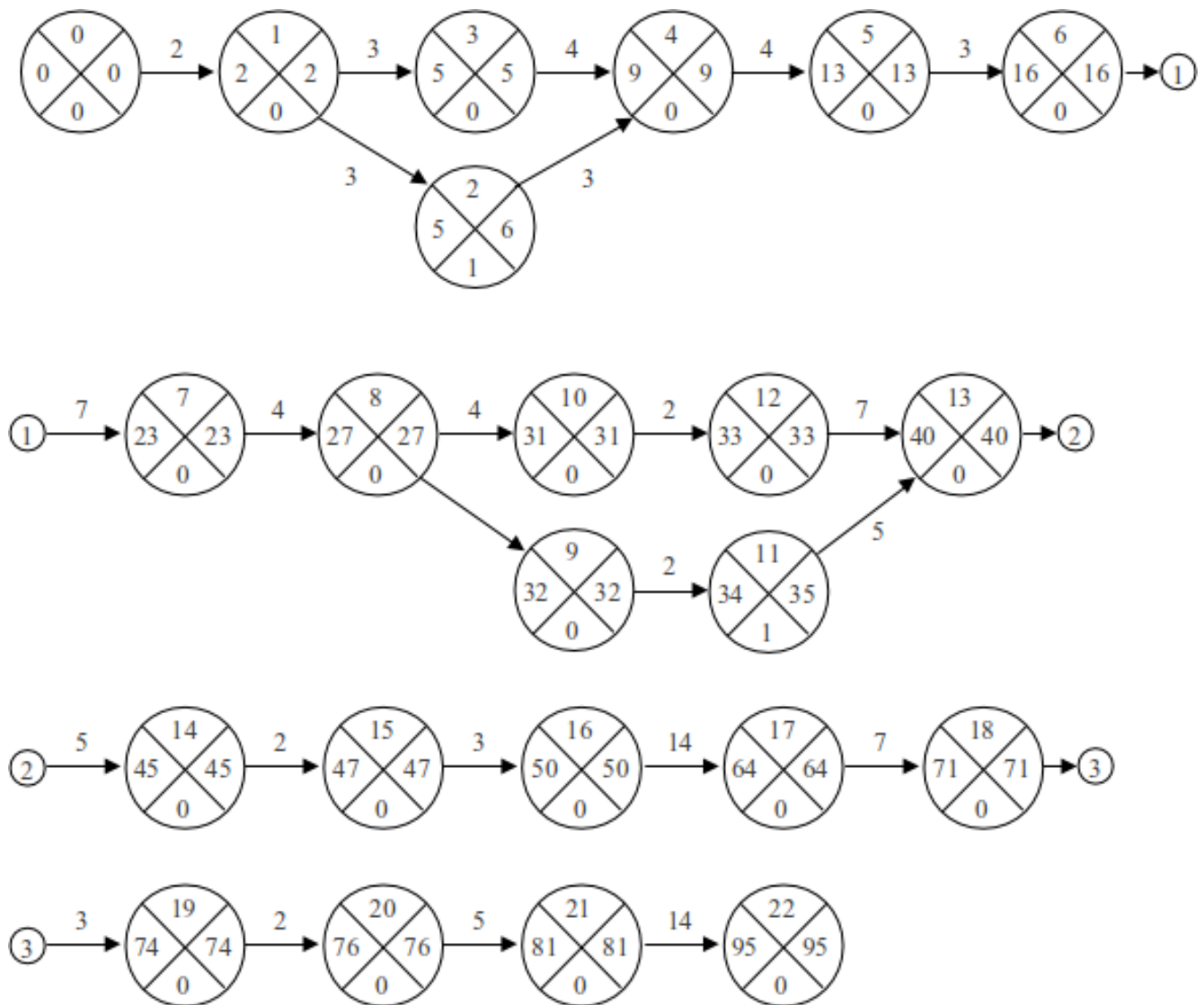


Рисунок 4.1 – Мережевий граф виконаних робіт

На рисунку 4.1 зображений мережевий граф, який отримано із вихідних даних таблиць. Знаходимо критичний шлях і розраховуємо ранній, пізній час і резерв часу.

Критичний шлях — це найбільш тривала по часу послідовність робіт, які

ведуть від вихідної до завершальної події. Величина критичного шляху визначає термін виконання всього комплексу по плануванню робіт.

Зміна тривалості будь-якої роботи, що лежить на критичному шляху, відповідним чином змінює термін настання завершальної події, тобто дату досягнення кінцевої мети, яка ставиться при плануванні розробки.

При плануванні комплексу операцій критичний шлях дозволяє знайти термін настання завершальної події. В процесі керування ходом розробки увага керівництва зосереджується на роботах критичного шляху. Це дозволяє найбільш доцільно і оперативно контролювати обмежене число робіт, що впливають на термін розробки, а також краще використати існуючі ресурси.

Оскільки в даному випадку мережевий граф досить простий, очевидно що критичний шлях рівний 95.

Дані розрахунків часу подій приведені в таблиці 4.3.

Табл. 4.3 – Параметри подій мережевого графіка

№ події	Ранній час	Пізній час	Резерв часу
0	0	0	0
1	2	2	0
2	5	6	1
3	5	5	0
4	9	9	0
5	13	13	0
6	16	16	0
7	23	23	0
8	27	27	0
9	32	32	0
10	31	31	0
11	34	35	1
12	33	33	0
13	40	40	0
14	45	45	0
15	47	47	0
16	50	50	0

№ події	Ранній час	Пізній час	Резерв часу
17	64	64	0
18	71	71	0
19	74	74	0
20	76	76	0
21	81	81	0
22	95	95	0

4.3 Економічне обґрунтування розробки та впровадження програми

Економічне обґрунтування розробки та впровадження програми будемо здійснювати на аналізі таких економічних показників:

S_{po} – сумарні витрати на розробку програмного забезпечення;

$\Delta E_{e2/1}$ – експлуатаційні витрати.

Розрахунок відповідних коефіцієнтів проводиться з врахуванням того, що варіаційні задачі діагностування раніше виконувались вручну.

4.3.1 Розрахунок витрат на розробку програмного забезпечення

Сумарні витрати на розробку програмного забезпечення S_{po} визначаються за формулою:

$$S_{po} = \sum_i t_{poi} \cdot B_{poi} \cdot [(1 + \omega_d) \cdot (1 + \omega_c) + \omega_n] + t_{mo} \cdot e_g, \quad (4.2)$$

де t_{poi} – час, що витрачається на розробку даної програми працівником i – ої кваліфікації, люд.-міс;

B_{poi} – основна заробітна плата розробника i – ої кваліфікації, грн/міс;

ω_d – коефіцієнт, що враховує додаткову заробітну плату розробникам програми, у відсотках від основної заробітної плати;

ω_c – коефіцієнт, що враховує нарахування органам соціального захисту на заробітну плату, у відсотках від основної та додаткової заробітної плати;

					ДП.ПЗ 04.00.00.000 ПЗ	Аркуш
						71
Зм.	Арк.	№ докум.	Підпис	Дата		

ω_n – коефіцієнт, що враховує накладні витрати установи, в якій розробляється ця програма, у відсотках до основної заробітної плати розробника;

t_{mo} – машинний час ЕОМ, необхідний для налагоджування даної програми, машино-год;

e_g – експлуатаційні витрати, що припадають на 1 год машинного часу.

Значення коефіцієнтів $\omega_d = 0$; $\omega_c = 0.375$; $\omega_n = 0.42$. Нехай $t_{mo} = 1$ люд.-міс, а $B_{pro_i} = 3000$ грн. Експлуатаційні витрати, що припадають на 1 год машинного часу, можуть бути визначені за витратою електроенергії:

$$S_g = P_{cp} \cdot C_{bod}, \quad (4.3)$$

де $P_{cp} = 90$ Вт – споживана потужність ЕОМ (ноутбук);

$C_{bod} = 0.8762$ – вартість 1 кВт/год електроенергії для підприємств.

Отже, за (4.3):

$$e_g = 0.09 \cdot 0.8762 = 0,079 \text{ грн/год.}$$

Необхідний час налагодження програми становить 24 машино-год.

Сумарні витрати на розробку програмного забезпечення складуть:

$$S_{po} = 1 \cdot 3000 \cdot ((1 + 0) \cdot (1 + 0.375) + 0.42) + 24 \cdot 0.079 = 5386.90 \text{ грн.}$$

Використання запропонованої програми не потребує додаткових капітальних вкладень у користувача.

4.3.2 Розрахунок можливого прибутку

Даний продукт буде розповсюджуватися на ліцензії GNU General Public License, що означає безкоштовне її розповсюдження. Тому для того щоб повернутися витрачені кошти на її розробку і підтримку, варто використовувати загальні методи поширення open source програм, це: заробіток підтримки користувачів продукту (супорт).

					ДП.ПЗ 04.00.00.000 ПЗ	Аркуш
Зм.	Арк.	№ докум.	Підпис	Дата		72

Буде введено два тарифи: річна підписка (2000 грн.), та помісячна (200 грн.).

Прогнози, зроблені на основі дослідження ринку, дозволяють нам очікувати наступний прибуток за 1 рік підтримки користувачів продукту на ринку.

Середня кількість компаній за рік буде становити порядку 10-ти. В середньому на ринку, кожна друга компанія буде користуватися послугою супорту і налаштування продукту. Решта половина буде тільки використовувати разову місячну передплату. Отже очікуваний прибуток за 1 рік на ринку буде становити:

$$5 \text{ місячний передплат} - 5 \cdot 200 = 1000 \text{ грн.}$$

$$5 \text{ річних передплат} - 5 \cdot 2000 = 10000 \text{ грн.}$$

Очікуваний прибуток за рік становитиме:

$$P = (10000 + 1000) - 5386.90 = 5613.1 \text{ грн.}$$

$$\text{Чистий прибуток: } P_{ch.} = (1 - 0.21) \cdot 5613.1 = 4434.35 \text{ грн.}$$

$$\text{Чистий місячний прибуток буде становити: } P_{m.ch.} = 369.53 \text{ грн.}$$

4.3.3 Розрахунок зведених економічних показників

Термін окупності додаткових капітальних вкладень визначається за формулою:

$$T_{OK} = \frac{S_{po}}{P_{ch.}} \quad (4.4)$$

Отже, за (4.4)

$$T_{OK} = 5386.90 / 369.53 = 14.5 \text{ місяця.}$$

Ефект, який отримує корпорація при користуванні даним продуктом полягає у легкості і гнучкості взаємодії між користувачами, спільною роботою над документами і завданнями.

В таблиці 4.4 наведені зведені економічні показники системи. З вище наведених розрахунків видно, що розробка та впровадження даної програми є економічно доцільною.

					ДП.ПЗ 04.00.00.000 ПЗ	Аркуш
						73
Зм.	Арк.	№ докум.	Підпис	Дата		

Табл. 4.4 – Зведені економічні показники розробки системи

Показник	Розмірність	Значення
Витрати на розробку програмного забезпечення	грн	5386.90
Очікуваний економічний ефект (за рік)	грн	4434.35
Термін окупності розробки графічного редактора	місяць	14.5

Таким чином, з цих економічних розрахунків випливає, що розробка корпоративної системи, розповсюдження якої базується на ліцензії GNU є економічно доцільним і дозволяє отримувати прибутки від підтримки користувачів і налаштування ПЗ.

5 ОХОРОНА ПРАЦІ

5.1 Значення охорони праці для забезпечення безпечних і здорових умов праці людей

					<i>ДП.ПЗ 04.00.00.000 ПЗ</i>	Аркуш
						75
<i>Зм.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>		

5.2 Аналіз потенційних небезпек та шкідливих факторів виробничого середовища

					ДП.ПЗ 04.00.00.000 ПЗ	Аркуш
						76
Зм.	Арк.	№ докум.	Підпис	Дата		

					ДП.ПЗ 04.00.00.000 ПЗ	Аркуш
						77
Зм.	Арк.	№ докум.	Підпис	Дата		

5.3 Забезпечення нормальних умов праці при роботі з ЕОМ

					<i>ДП.ПЗ 04.00.00.000 ПЗ</i>	Аркуш
						78
<i>Зм.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>		

					ДП.ПЗ 04.00.00.000 ПЗ	Аркуш
						79
Зм.	Арк.	№ докум.	Підпис	Дата		

					<i>ДП.ПЗ 04.00.00.000 ПЗ</i>	Аркуш
						80
<i>Зм.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>		

					ДП.ПЗ 04.00.00.000 ПЗ	Аркуш
						81
Зм.	Арк.	№ докум.	Підпис	Дата		

					ДП.ПЗ 04.00.00.000 ПЗ	Аркуш
Зм.	Арк.	№ докум.	Підпис	Дата		82

					ДП.ПЗ 04.00.00.000 ПЗ	Аркуш
Зм.	Арк.	№ докум.	Підпис	Дата		83

**5.4 Забезпечення безпеки монтажу, пусконаладжувальних, ремонтних
робіт та експлуатації ЕОМ і комп’ютерних мереж**

					<i>ДП.ПЗ 04.00.00.000 ПЗ</i>	Аркуш
<i>Зм.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>		84

					<i>ДП.ПЗ 04.00.00.000 ПЗ</i>	Аркуш
<i>Зм.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>		85

5.5 Пожежна безпека та безпека в НС

					ДП.ПЗ 04.00.00.000 ПЗ	Аркуш
						86
Зм.	Арк.	№ докум.	Підпис	Дата		

					<i>ДП.ПЗ 04.00.00.000 ПЗ</i>	Аркуш
<i>Зм.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>		87

					<i>ДП.ПЗ 04.00.00.000 ПЗ</i>	Аркуш
<i>Зм.</i>	<i>Арк.</i>	<i>№ докум.</i>	<i>Підпис</i>	<i>Дата</i>		88

					ДП.ПЗ 04.00.00.000 ПЗ	Аркуш
						89
Зм.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

Завдяки сучасним технологіям і корпоративним стандартам, розвиток розробки комерційних продуктів виріс дуже стрімко. Зокрема сюди і відноситься відносно молодий напрямок — це розробка корпоративних порталів. Було встановлено стандарти щодо розробки додатків і аплікацій – це допомогло добитися легкої інтеграції і взаємодії. Також проведено аналіз сучасного стану і потреб ринку в даній сфері, наведено всі вимоги до програмного продукту. Проведено аналіз щодо економічної вигоди та прораховано всі важливі аспекти щодо охорони праці.

СПИСОК ПОСИЛАНЬ НА ДЖЕРЕЛА

1. <http://www.jcp.org/en/jsr/detail?id=286> - стандарт портлетів Java Portlet 2.0
2. <http://www.jcp.org/en/jsr/detail?id=168> - стандарт портлетів Java Portlet 1.0
3. <http://google.com> - пошук доступної в інтернеті інформації
4. <http://tomcat.apache.org/tomcat-5.5-doc/> – специфікація серлетів
5. <http://pz.nung.edu.ua/> - сайт кафедри ПЗАС
6. <http://www.intranetno.ru/> - бізнес рішення на базі SaaS, PaaS
7. http://en.wikipedia.org/wiki/Entity-attribute-value_model - EAV модель

					<i>ДП.ПЗ 04.00.00.000 ПЗ</i>	Аркуш
						91
Зм.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

ДОДАТОК А

Домен для мапінгу даних

```
package com.diploma.ccms.domain;
import java.util.Collection;
import java.util.Date;
import java.util.HashSet;
import java.util.List;
import java.util.Set;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.EntityManager;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Lob;
import javax.persistence.ManyToOne;
import javax.persistence.PersistenceContext;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;
import javax.persistence.TypedQuery;
import javax.persistence.Version;
import javax.validation.constraints.NotNull;
import org.apache.commons.lang3.builder.ReflectionToStringBuilder
    ;
import org.apache.commons.lang3.builder.ToStringStyle;
import org.springframework.beans.factory.annotation.Configurable;
import org.springframework.format.annotation.DateTimeFormat;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.context.
    SecurityContextHolder;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.transaction.annotation.Transactional;
import com.diploma.ccms.domain.service.DomainUserInterface;
@Entity
@Configurable
public class Worker implements UserDetails, DomainUserInterface {
```


Продовження додатку А

Продовження додатку А

```
private static final long serialVersionUID = 1148968754347580432L
    ;

@NotNull
@Column(unique = true)
private String login;
@NotNull
private String pass;
@NotNull
private String name;
@NotNull
private String surname;
private String phone;
private String mobile;
private String privateMail;
private String street;
@Temporal(TemporalType.DATE)
@DateTimeFormat(pattern = dd/MM/yyyy)
private Date birthday;
@Temporal(TemporalType.DATE)
@DateTimeFormat(pattern = dd/MM/yyyy)
private Date dateHire;
@ManyToOne
private WorkerRole roleName;
@NotNull
@ManyToOne
private Region regionName;
@NotNull
@ManyToOne
private Team teamName;
@NotNull
@ManyToOne
private WorkerJobType jobTypeName;
@Lob
private byte[] photo;
public static Worker getPrincipal(){
```

```
return (Worker) SecurityContextHolder.getContext().  
    getAuthentication().getPrincipal();  
  
}  
public String getLogin() {  
return this.login;  
}  
public void setLogin(String login) {  
this.login = login;  
}  
public String getPass() {  
return this.pass;  
}  
public void setPass(String pass) {  
this.pass = pass;  
}  
public String getName() {  
return this.name;  
}  
public void setName(String name) {  
this.name = name;  
}  
public String getSurname() {  
return this.surname;  
}  
public void setSurname(String surname) {  
this.surname = surname;  
}  
public String getPhone() {  
return this.phone;  
}  
public void setPhone(String phone) {  
this.phone = phone;  
}  
public String getPrivateMail() {  
return this.privateMail;  
}
```

```

public void setPrivateMail(String privateMail) {

this.privateMail = privateMail;

}

public String getStreet() {

return this.street;

}

public void setStreet(String street) {

this.street = street;

}

public Date getBirthday() {

return this.birthday;

}

public void setBirthday(Date birthday) {

this.birthday = birthday;

}

public Date getDateHire() {

return this.dateHire;

}

public void setDateHire(Date dateHire) {

this.dateHire = dateHire;

}

public WorkerRole getRoleName() {

return this.roleName;

}

public void setRoleName(WorkerRole roleName) {

this.roleName = roleName;

}

public Region getRegionName() {

return this.regionName;

}

public void setRegionName(Region regionName) {

this.regionName = regionName;

}

public Team getTeamName() {

return this.teamName;

}

```

```

}

public void setTeamName(Team teamName) {
this.teamName = teamName;
}

public WorkerJobType getJobTypeName() {
return this.jobTypeName;
}

public void setJobTypeName(WorkerJobType jobTypeName) {
this.jobTypeName = jobTypeName;
}

public byte[] getPhoto() {
return this.photo;
}

public void setPhoto(byte[] photo) {
this.photo = photo;
}

@PersistenceContext
transient EntityManager entityManager;

public static final EntityManager entityManager() {
EntityManager em = new Worker().entityManager;
if (em == null)
throw new IllegalStateException(Entity manager has not been
    injected (is the Spring Aspects JAR configured as an AJC/AJDT
    aspects library?));
return em;
}

public static long countWorkers() {
return entityManager().createQuery(SELECT COUNT(o) FROM Worker o,
    Long.class).getSingleResult();
}

public static List<Worker> findAllWorkers() {
return entityManager().createQuery(SELECT o FROM Worker o, Worker
    .class).getResultList();
}

public static Worker findWorker(Long id) {
if (id == null)

```

```

return null;

return entityManager().find(Worker.class, id);
}

public static List<Worker> findWorkerEntries(int firstResult, int
    maxResults) {
return entityManager().createQuery(SELECT o FROM Worker o, Worker
    .class).setFirstResult(firstResult).setMaxResults(maxResults).
    getResultList();
}

@Transactional
public void persist() {
if (this.entityManager == null)
this.entityManager = entityManager();
this.entityManager.persist(this);
}

@Transactional
public void remove() {
if (this.entityManager == null)
this.entityManager = entityManager();
if (this.entityManager.contains(this)) {
this.entityManager.remove(this);
} else {
Worker attached = Worker.findWorker(this.id);
this.entityManager.remove(attached);
}
}

@Transactional
public void flush() {
if (this.entityManager == null)
this.entityManager = entityManager();
this.entityManager.flush();
}

@Transactional
public void clear() {
if (this.entityManager == null)
this.entityManager = entityManager();

```

```

this.entityManager.clear();

}

@Transactional
public Worker merge() {
if (this.entityManager == null)
this.entityManager = entityManager();
Worker merged = this.entityManager.merge(this);
this.entityManager.flush();
return merged;
}

public String toString() {
return ReflectionToStringBuilder.toString(this, ToStringStyle.
    SHORT_PREFIX_STYLE);
}

public static TypedQuery<Worker> findWorkersByLoginEquals(String
    login) {
if (login == null || login.length() == 0)
throw new IllegalArgumentException(The login argument is required
    );
EntityManager em = Worker.entityManager();
TypedQuery<Worker> q = em.createQuery(SELECT o FROM Worker AS o
    WHERE o.login = :login, Worker.class);
q.setParameter(login, login);
return q;
}

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

@Version
@Column(name = version)
private Integer version;
public Long getId() {
return this.id;
}

public void setId(Long id) {

```

```
this.id = id;

}

public Integer getVersion() {
return this.version;
}

public void setVersion(Integer version) {
this.version = version;
}

@Override
public Collection<? extends GrantedAuthority> getAuthorities() {
Set<GrantedAuthority> roles = new HashSet<GrantedAuthority>();
roles.add(roleName);
return roles;
}

@Override
public String getPassword() {
return pass;
}

@Override
public String getUsername() {
return login;
}

@Override
public boolean isAccountNonExpired() {
return true;
}

@Override
public boolean isAccountNonLocked() {
return true;
}

@Override
public boolean isCredentialsNonExpired() {
return true;
}

@Override
public boolean isEnabled() {
```



```

return true;

}

public String getUiValue() {
return name + " " + surname;
}

public String getMobile() {
return mobile;
}

public void setMobile(String mobile) {
this.mobile = mobile;
}
}

```

Контроллер даних

```

package com.diploma.ccms.web.controller;

import java.io.UnsupportedEncodingException;
import java.util.ArrayList;
import java.util.List;

import javax.servlet.http.HttpServletRequest;
import javax.validation.Valid;

import org.joda.time.format.DateTimeFormat;
import org.springframework.context.i18n.LocaleContextHolder;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.util.UriUtils;
import org.springframework.web.util.WebUtils;

```



```
import com.diploma.ccms.domain.Region;

import com.diploma.ccms.domain.Team;
import com.diploma.ccms.domain.Worker;
import com.diploma.ccms.domain.WorkerJobType;
import com.diploma.ccms.domain.WorkerRole;

@RequestMapping("/workers")
@Controller
public class WorkerController {

    @RequestMapping(method = RequestMethod.POST, produces = "text/
        html")
    public String create(@Valid Worker worker, BindingResult
        bindingResult, Model uiModel, HttpServletRequest
        httpRequest) {
        if (bindingResult.hasErrors()) {
            populateEditForm(uiModel, worker);
            return "workers/create";
        }
        uiModel.asMap().clear();
        worker.persist();
        uiModel.addAttribute("menu", "WORKER");
        return "redirect:/workers/" + encodeUrlPathSegment(worker.getId()
            .toString(), httpRequest);
    }

    @RequestMapping(params = "form", produces = "text/html")
    public String createForm(Model uiModel) {
        populateEditForm(uiModel, new Worker());
        List<String[]> dependencies = new ArrayList<String[]>();
        if (Region.countRegions() == 0) {
            dependencies.add(new String[] { "region", "regions" });
        }
        if (Team.countTeams() == 0) {
            dependencies.add(new String[] { "team", "teams" });
        }
    }
}
```

```

}

if (WorkerJobType.countWorkerJobTypes() == 0) {
dependencies.add(new String[] { "workerjobtype", "workerjobtypes"
    });
}
uiModel.addAttribute("dependencies", dependencies);
uiModel.addAttribute("menu", "WORKER");
return "workers/create";
}

@RequestMapping(value =("/{id}", produces = "text/html")
    public String show(@PathVariable("id") Long id, Model uiModel) {
addDateTimeFormatPatterns(uiModel);
uiModel.addAttribute("worker", Worker.findWorker(id));
uiModel.addAttribute("itemId", id);
uiModel.addAttribute("menu", "WORKER");
return "workers/show";
}

@RequestMapping(produces = "text/html")
    public String list(@RequestParam(value = "page", required = false
        ) Integer page, @RequestParam(value = "size", required = false
        ) Integer size, Model uiModel) {
uiModel.addAttribute("workers", Worker.findAllWorkers());
//
uiModel.addAttribute("menu", "WORKER");
addDateTimeFormatPatterns(uiModel);
return "workers/list";
}

@RequestMapping(method = RequestMethod.PUT, produces = "text/html
    ")
    public String update(@Valid Worker worker, BindingResult
        bindingResult, Model uiModel, HttpServletRequest
        httpRequest) {
if (bindingResult.hasErrors()) {

```

```

populateEditForm(uiModel, worker);

uiModel.addAttribute("menu", "WORKER");
return "workers/update";
}

uiModel.asMap().clear();
worker.merge();
uiModel.addAttribute("menu", "WORKER");
return "redirect:/workers/" + encodeUrlPathSegment(worker.getId()
    .toString(), httpRequest);
}

@RequestMapping(value =("/{id}", params = "form", produces = "
    text/html")
public String updateForm(@PathVariable("id") Long id, Model
    uiModel) {
populateEditForm(uiModel, Worker.findWorker(id));
uiModel.addAttribute("menu", "WORKER");
return "workers/update";
}

@RequestMapping(value =("/{id}", method = RequestMethod.DELETE,
    produces = "text/html")
public String delete(@PathVariable("id") Long id, @RequestParam(
    value = "page", required = false) Integer page,
    @RequestParam(value = "size", required = false) Integer size,
    Model uiModel) {
Worker worker = Worker.findWorker(id);
worker.remove();
uiModel.asMap().clear();
uiModel.addAttribute("page", (page == null) ? "1" : page.toString
    ());
uiModel.addAttribute("size", (size == null) ? "10" : size.
    toString());
uiModel.addAttribute("menu", "WORKER");
return "redirect:/workers";
}

```

```

void addDateTimeFormatPatterns(Model uiModel) {
uiModel.addAttribute("worker_birthdate_date_format",
    DateTimeFormat.patternForStyle("M-", LocaleContextHolder.
        getLocale()));
uiModel.addAttribute("worker_datehire_date_format",
    DateTimeFormat.patternForStyle("M-", LocaleContextHolder.
        getLocale()));
}

void populateEditForm(Model uiModel, Worker worker) {
uiModel.addAttribute("worker", worker);
addDateTimeFormatPatterns(uiModel);
uiModel.addAttribute("regions", Region.findAllRegions());
uiModel.addAttribute("teams", Team.findAllTeams());
uiModel.addAttribute("workerjobtypes", WorkerJobType.
    findAllWorkerJobTypes());
uiModel.addAttribute("workerroles", WorkerRole.findAllWorkerRoles
    ());
uiModel.addAttribute("menu", "WORKER");
}

String encodeUrlPathSegment(String pathSegment,
    HttpServletRequest httpRequest) {
String enc = httpRequest.getCharacterEncoding();
if (enc == null) {
enc = WebUtils.DEFAULT_CHARACTER_ENCODING;
}
try {
pathSegment = UriUtils.encodePathSegment(pathSegment, enc);
} catch (UnsupportedEncodingException uee) {
}
return pathSegment;
}

```

```
@RequestMapping(params = { "find=ByLoginEquals", "form" }, method
    = RequestMethod.GET)
```

```
public String findWorkersByLoginEqualsForm(Model uiModel) {
uiModel.addAttribute("menu", "WORKER");
return "workers/findWorkersByLoginEquals";
}
```

```
@RequestMapping(params = "find=ByLoginEquals", method =
    RequestMethod.GET)
public String findWorkersByLoginEquals(@RequestParam("login")
    String login, Model uiModel) {
uiModel.addAttribute("workers", Worker.findWorkersByLoginEquals(
    login).getResultList());
uiModel.addAttribute("menu", "WORKER");
return "workers/list";
}
```

```
/**
 * Generate preview for e-mail
 *
 * @param petId
 * @param model
 * @return
 */
@RequestMapping(value = "/preview_email", method = RequestMethod.
    POST)
@ResponseBody
public String getPreview(@RequestParam("data") String data) {
return data;
}

}
```

Сервіс для збереження даних

```
package com.diploma.ccms.service;
```



```
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.
    UserDetailsServiceImpl;
import org.springframework.security.core.userdetails.
    UsernameNotFoundException;

import com.diploma.ccms.domain.Worker;

public class UserService implements UserDetailsServiceImpl {

    @Override
    public UserDetails loadUserByUsername(String username) throws
        UsernameNotFoundException {
        return Worker.findWorkersByLoginEquals(username).getSingleResult
            ();
    }

}
```

Інтерфейс для збереження даних користувача

```
package com.diploma.ccms.domain.service;

public interface DomainUserInterface {
    /**
     * used for getting value for showing on UI form
     */
    public String getUiValue();
}
```

Конфігурація виглядів для контролера

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<tiles-definitions>
<definition extends="default" name="workers/list">
```



```

<put-attribute name="body" value="/WEB-INF/views/workers/list.
    jsp" />
</definition>
<definition extends="default" name="workers/show">
<put-attribute name="body" value="/WEB-INF/views/workers/show.
    jsp" />
</definition>
<definition extends="default" name="workers/create">
<put-attribute name="body" value="/WEB-INF/views/workers/create.
    jsp" />
</definition>
<definition extends="default" name="workers/update">
<put-attribute name="body" value="/WEB-INF/views/workers/update.
    jsp" />
</definition>
<definition extends="default" name="workers/
    findWorkersByLoginEquals">
<put-attribute name="body"
value="/WEB-INF/views/workers/findWorkersByLoginEquals.jsp" />
</definition>
</tiles-definitions>

```

Вигляд створення для користувача

```

<?xml version=1.0 encoding=UTF-8 standalone=no?>
<div xmlns:c=http://java.sun.com/jsp/jstl/core xmlns:field=urn:
    jsptagdir:/WEB-INF/tags/form/fields xmlns:form=urn:jsptagdir:/
    WEB-INF/tags/form
xmlns:jsp=http://java.sun.com/JSP/Page xmlns:spring=http://www.
    springframework.org/tags version=2.0>
<jsp:directive.page contentType=text/html;charset=UTF-8 />
<jsp:output omit-xml-declaration=yes />

<section class=main-section grid_7>

<div class=main-content>

```



```
<header>

<h2>Employees</h2>
</header>
<section class=container_6 clearfix>
<field:input field=login id=
    c_com_diploma_ccms_domain_Worker_login required=true label=
    Login smallLabel=Enter login type=text/>
<field:input field=pass id=c_com_diploma_ccms_domain_Worker_pass
    required=true label=Password smallLabel=Enter password type=
    password/>
<field:input field=name id=c_com_diploma_ccms_domain_Worker_name
    required=true label=Name smallLabel=Enter name type=text/>
<field:input field=surname id=
    c_com_diploma_ccms_domain_Worker_surname required=true label=
    Surname smallLabel=Enter surname type=text/>
<field:input field=mobile id=
    c_com_diploma_ccms_domain_Worker_mobile required=false label=
    Mobile phone number smallLabel=Private phone number type=text
    />
<field:input field=privateMail id=
    c_com_diploma_ccms_domain_Worker_privateMail required=false
    label=Email smallLabel=A valid email address type=email/>
<field:select field=regionName id=
    c_com_diploma_ccms_domain_Worker_regionName items=${regions}
    required=true label=Region smallLabel=Please select/>
<field:input field=street id=
    c_com_diploma_ccms_domain_Worker_street required=false label=
    Address smallLabel=Enter full home address type=text/>
<field:input field=birthday id=
    c_com_diploma_ccms_domain_Worker_birthday required=false label
    =Birthdate smallLabel=mm/dd/yyyy type=date/>
<field:input field=dateHire id=
    c_com_diploma_ccms_domain_Worker_dateHire required=false label
    =Hire Date smallLabel=mm/dd/yyyy type=date/>
```

```

<field:select field=roleName id=
    c_com_diploma_ccms_domain_Worker_roleName items=${workerroles}
    required=true label=Employee role smallLabel=Please select
    role/>
<field:select field=teamName id=
    c_com_diploma_ccms_domain_Worker_teamName items=${teams}
    required=true label=Employee related team smallLabel=Please
    select working team/>
<field:select field=jobTypeName id=
    c_com_diploma_ccms_domain_Worker_jobTypeName items=${
    workerjobtypes} required=true label=Job type smallLabel=Like
    QA, Developer, etc../>
<field:input field=phone id=
    c_com_diploma_ccms_domain_Worker_phone required=false label=
    Phone number smallLabel=Working office phone number type=text
    />
<field:textarea field=photo id=
    c_com_diploma_ccms_domain_Worker_photo />
</form:create>
</section>
</div>
</section>
</div>

```

Вигляд для показу даних про

```

<?xml version=1.0 encoding=UTF-8 standalone=no?>
<div xmlns:field=urn:jsptagdir:/WEB-INF/tags/form/fields
xmlns:c=http://java.sun.com/jsp/jstl/core
xmlns:sec=http://www.springframework.org/security/tags
xmlns:jsp=http://java.sun.com/JSP/Page xmlns:page=urn:jsptagdir:/
    WEB-INF/tags/form version=2.0>
<jsp:directive.page contentType=text/html;charset=UTF-8 />
<jsp:output omit-xml-declaration=yes />
<!-- Main Section -->
<section class=main-section grid_7>
<div class=main-content grid_4 alpha>

```



```
<header class=clearfix>

<span class=avatar> <!-- -->
</span>
<hgroup>
<h2>${worker.name} ${worker.surname}</h2>
<h4>
<a href=#>${worker.jobTypeName.jobTypeName}</a>
</h4>
</hgroup>
<p class=tags>
<!-- <a href=#>Add To Contacts</a> -->
</p>
</header>
<section>
<script type=text/javascript>
    $(document).ready(function() {
        // Add markItUp! to your textarea in one line
        $('.markItUpTextarea').markItUp(mySettings, {
            root : 'markitup/skins/simple/'
        });
    });
</script>
<style>
.markItUp {
width: 390px;
}
</style>
<sec:authentication var=principal_login property=principal.login
/>
<c:choose>
<c:when test=${worker.login eq principal_login}>
<h3> Have a nice day ;) </h3>
</c:when>
<c:otherwise>
<h3>Send private message to ${worker.name} ${worker.surname}</h3>
```



```

<form method=post action=/ccms/messages?id=${worker.id} style=
  width: 400px;>

<field:input id=c_com_diploma_ccms_domain_Message_title field=
  title required=true label=Title smallLabel= type=text/>
<textarea id=c_com_diploma_ccms_domain_Message_text name=text
  class=markItUpTextarea style=height: 80px; width: 100%;
  required=required><!-- --></textarea>
<button class=fr button button-gray type=submit>Send message</
  button>
</form>
</c:otherwise>
</c:choose>
<div class=clear>
<!-- -->
</div>
<!-- <h3>History</h3>
<ul class=listing list-view>
<li class=note><a href=editnote.html class=more></a> <span class=
  timestamp>Dec 28, 2010</span>
<p>Vestibulum ultrices vehicula leo ac tristique. Mauris id nisl
  nibh.</p>
<div class=entry-meta>Posted by Administrator</div></li>
<li class=note><a href=editnote.html class=more></a> <span class=
  timestamp>Dec 28, 2010</span>
<p>Vestibulum ultrices vehicula leo ac tristique. Mauris id nisl
  nibh.</p>
<div class=entry-meta>Posted by Administrator</div></li>
<li class=note><a href=editnote.html class=more></a> <span class=
  timestamp>Dec 28, 2010</span>
<p>Vestibulum ultrices vehicula leo ac tristique. Mauris id nisl
  nibh.</p>
<div class=entry-meta>Posted by Administrator</div></li>
</ul> -->
</section>
</div>
<div class=preview-pane grid_3 omega>

```

```

<div class=content>

<h3>${worker.name}s contact information</h3>
<ul class=profile-info>
<li class=email><c:if test=${empty worker.privateMail}>:: none
    ::</c:if> ${worker.privateMail}<span>email</span></li>
<li class=mobile><c:if test=${empty worker.mobile}>:: none ::</c:
    if>${worker.mobile}<span>mobile</span></li>
<li class=phone><c:if test=${empty worker.phone}>:: none ::</c:if
    >${worker.phone}<span>work</span></li>
</ul>
<!-- <h3>Tasks About John</h3>
None so far. <a href=#>Add a task now</a> -->
<h3>Additional info</h3>
<ul class=profile-info>
<li class=calendar-day><c:if test=${empty worker.birthday}>::
    none ::</c:if>${worker.birthday}<span>birthday</span></li>
<li class=calendar-day><c:if test=${empty worker.dateHire}>::
    none ::</c:if>${worker.dateHire}<span>hire date</span></li>
<li class=building><c:if test=${empty worker.regionName.
    regionName}>:: none ::</c:if>${worker.regionName.regionName}<
    span>living region</span></li>
<li class=house><c:if test=${empty worker.street}>:: none ::</c:
    if>${worker.street}<span>home address</span></li>
</ul>
</div>
<div class=preview>
<!-- -->
</div>
</div>
</section>
<!-- Main Section End -->
<!-- <page:show id=ps_com_diploma_ccms_domain_Worker object=${
    worker}
path=/workers z=iQamgfMz7tfHSdGgWB8YCdDd83Q=>
<field:display field=login
id=s_com_diploma_ccms_domain_Worker_login object=${worker}

```

```

z=4HjNYZAHZPDLTqADXwCdyd4GRAk= />

<field:display field=pass id=
    s_com_diploma_ccms_domain_Worker_pass
object=${worker} z=HYhzipM8Xx937WviE05t1DOTtIbU= />
<field:display field=name id=
    s_com_diploma_ccms_domain_Worker_name
object=${worker} z=egBKl4f8HsCgN5Ynj/jGjRUelAo= />
<field:display field=surname
id=s_com_diploma_ccms_domain_Worker_surname object=${worker}
z=ES6rDyTeRfbtJ4oZ9zGg6RpkdI0= />
<field:display field=phone
id=s_com_diploma_ccms_domain_Worker_phone object=${worker}
z=1VohSBGktVWKME/kgylZA99MluU= />
<field:display field=privateMail
id=s_com_diploma_ccms_domain_Worker_privateMail object=${worker}
z=JZug5Vfq4gKYkQsNndLyunI+pLo= />
<field:display field=street
id=s_com_diploma_ccms_domain_Worker_street object=${worker}
z=e/Rkpgq+iZVtm51YsnThOISuQXRA= />
<field:display date=true
dateTimePattern=${worker_birthday_date_format} field=birthday
id=s_com_diploma_ccms_domain_Worker_birthday object=${worker}
z=EmRENzOD8cefRn6vszpdLPB3+04= />
<field:display date=true
dateTimePattern=${worker_datehire_date_format} field=dateHire
id=s_com_diploma_ccms_domain_Worker_dateHire object=${worker}
z=oQ0LtnMFCgHhLh6m2nvl4LtMrJU= />
<field:display field=roleName
id=s_com_diploma_ccms_domain_Worker_roleName object=${worker}
z=wJnkS5H0d6iZ3AnuqTU07TI2wX8= />
<field:display field=regionName
id=s_com_diploma_ccms_domain_Worker_regionName object=${worker}
z=obgn4zC59pMlQ8GaXlDJYjUcoGs= />
<field:display field=teamName
id=s_com_diploma_ccms_domain_Worker_teamName object=${worker}
z=iRDO/HwFc7JQ+f+sLvRXQwEzBOW= />

```

```
<field:display field=jobTypeName  
id=s_com_diploma_ccms_domain_Worker_jobTypeName object=${worker}  
z=a43X3qenfQB/KwX1hCUeDmDGIYM= />  
<field:display field=photo  
id=s_com_diploma_ccms_domain_Worker_photo object=${worker}  
z=lcI4umkwCdlmQFaIzB4gefboNPI= />  
</page:show>  
-->  
</div>
```

БІБЛІОГРАФІЧНА ДОВІДКА

ТЕМА ДИПЛОМНОГО ПРОЕКТУ: «Розробка програмного та алгоритмічного забезпечення багатофункціональної корпоративної системи для спільної роботи, управління документами і проектами засобами Java EE»

Обсяг пояснювальної записки: 122 аркуша

Дата закінчення проекту 4 травня 2012 р.

Підпис студента-дипломника _____