**МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**„КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”**

**Факультет прикладної математики**

**КОМПЛЕКСНИЙ КУРСОВИЙ ПРОЕКТ**

**Розробка back-end**

з дисципліни “Проектування корпоративних інформаційних систем”

на тему:

**ІНФОРМАЦІЙНА СИСТЕМА ДЛЯ УПРАВЛІННЯ ІТ КОМПАНІЄЮ**

Виконав студент

групи КП-41с

Волинський Я. О.

Залікова книжка КП-4101с

Керівник роботи

доцент, к.т.н. Павловський В.І.

До захисту допущено

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата, підпис)

Захищено з оцінкою

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата, підпис)

Київ – 2014

**РЕФЕРАТ**

Об’єктом розробки є корпоративна інформаційна система для управління ІТ компанією.

Метою розробки було отримання програмного забезпечення, що надає можливість керівникам ІТ компанії здійснювати просте управляння всіма бізнес процесами компанії.

Результатом розробки є оформлений проект системи та реалізоване програмне забезпечення.

Основним методом проектування було структурне проектування. Побудовано функціональну модель бізнес-процесів (IDEF0), діаграму потоків даних (DFD), модель даних підприємства (IDEF1x), а також створено об’єктно-орієнтовану модель корпоративної інформаційної системи, побудовано діаграми варіантів використання та класів.

Програмні інструменти, які використовувалися для проектування: ERWin 7, Visual Paradigm 8.0. Для реалізації використано такі засоби: PostgreSQL, IntelliJ IDEA 14.0.3.

Здійснено аналіз предметної області ІТ компанії. Розроблено програмне забезпечення, що складається з таких компонентів: серверна частина, база даних, клієнтська частина. В базі даних зберігаються відомості про працівників, проекти компанії, обладнання, пошта, конференції, зарплати, відділи, фінанси. Розроблений програмний продукт можна використовувати в ІТ компаніях.

**РЕФЕРАТ**

Объектом разработки является корпоративная информационная система для управления автосервисом.

Целью разработки было получение программного обеспечения, которое дает возможность работникам автосервиса осуществлять простое управление всеми бизнес процессами на предприятии.

Результатом разработки является оформленный проект системы и реализованное программное обеспечение.

Основным методом проектирования было структурное проектирование. Построено функциональную модель бизнес-процессов (IDEF0), диаграмму потоков данных (DFD), модель данных предприятия (IDEF1x), а также создано объектно-ориентированную модель корпоративной информационной системы, построены диаграммы вариантов использования и классов.

Программные инструменты, которые использовались для проектирования: ERWin 7, Visual Paradigm 8.0. Для реализации использованы следующие средства: PostgreSQL, IntelliJ IDEA 14.0.3.

Осуществлен анализ предметной области предприятия автосервиса. Разработано программное обеспечение, состоящее из следующих компонентов: серверная часть, база данных, клиентская часть. В базе данных хранятся сведения о работинках, проектах компании, оборудовании, поште, конференциях, зарплатах, отделах, финансах. Разработанный программный продукт можно использовать в ИТ компаниях.

**ABSTRACT**

The object of development is the information system for automobile repair shop.

The goal of the development was to provide software that enables automobile repair shop employees to perform simple management of all business processes in the enterprise.

Result of the development of the project is designed and implemented software.

The basic method was designing structural design. I constructed a functional model of business processes (IDEF0), data flow diagram (DFD), the enterprise data model (IDEF1x), and created an object-oriented model of corporate information systems based use case diagrams and classes.

Software tools used to design are ERWin 7 and Visual Paradigm 8.0. I have used the following tools for implementation: PostgreSQL, IntelliJ IDEA 14.0.3.

I made an analysis of the domain of automobile repair shop. The software consists of the following components: backend, database client. The database stored information about employees, company projects, equipment, mailing, mettings, salaries, departments, cash flow. The developed software can be used in IT companies.

[ВСТУП 6](#_Toc421062839)

[1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАВДАННЯ НА РОЗРОБКУ І ПРОЕКТУВАННЯ КОРПОРАТИВНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ІТ-КОМПАНІЇ 7](#_Toc421062840)

[1.1 Аналіз діяльності ІТ-компанії 7](#_Toc421062841)

[1.2 Аналіз інформаційних потоків ІТ-компанії 8](#_Toc421062842)

[1.3 Завдання КІС для ІТ-компанії 8](#_Toc421062843)

[1.4. Постановка завдання на проектування і розробку 10](#_Toc421062844)

[2. ПРОЕКТУВАННЯ КОРПОРАТИВНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ІТ-КОМПАНІЇ 11](#_Toc421062845)

[2.1 Вимоги до системи та її фунцкіональності 11](#_Toc421062846)

[2.2 Вибір методологій моделювання 12](#_Toc421062847)

[2.3. Функціональна модель IDEF0 14](#_Toc421062848)

[3. РЕАЛІЗАЦІЯ СИСТЕМИ 24](#_Toc421062849)

[3.1. Побудова моделі. Реалізація класів сутностей. 24](#_Toc421062850)

[3.2. Реалізація DAO-класів. 28](#_Toc421062851)

[3.3. Реалізація додатку. 31](#_Toc421062853)

[ВИСНОВКИ 33](#_Toc421062854)

[СПИСОК ЛІТЕРАТУРИ 34](#_Toc421062855)

# ВСТУП

Ефективна організація прозорих бізнес-процесів на підприємстві є першочерговою задачею для керівництва. Без неї навіть не може стояти питання прибутковості підприємства – виживання підприємства ставиться під сумнів.

З одного боку на підприємство впливає велика кількість зовнішніх чинників у найрізноманітніших формах – тому управлінському складу необхідне всебічне бачення ситуації, з іншого – чітке розуміння процесів, місця і наслідків дій всередині підприємства.

Інформатизація взаємодії із зовнішнім світом (у випадку ІТ-компанії – пошук нових співробітників, організація постачання обладнання та ін.) дозволяє зробити ці процеси більш прозорими. Моделювання уже існуючих бізнес-процесів дозволяє виявити слабкі місця у всьому виробничому циклі, а моделювання нових – перевірити доцільність змін без їхнього втілення.

Створення корпоративної інформаційної системи ІТ-компанії є актуальною задачею. Це пов’язане, перш за все, з ростом кількості ІТ-компаній та розвитком структури таких компаній. Вона дозволить ефективно організувати процес лікування і суміжні з ним процеси, що є дуже важливим у такій специфічній галузі як медицина, оскільки на кону може стояти здоров'я та життя пацієнта.

## 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАВДАННЯ НА РОЗРОБКУ І ПРОЕКТУВАННЯ КОРПОРАТИВНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ІТ-КОМПАНІЇ

## 1.1 Аналіз діяльності ІТ-компанії

Однією з актуальних проблем для українських компаній сьогодні є підвищення ефективності бізнесу за допомогою впровадження сучасних методів управління, заснованих на інформаційних технологіях. Як показує світовий досвід, грамотне застосування ІТ сприяє підвищенню керованості бізнесу, росту продуктивності, зниженню витрат і підвищує вартість компанії.

Сьогодні ІТ-бюджети найбільших українських компаній виміряються багатьма десятками мільйонів доларів. Суми ІТ-бюджетів стають значними навіть у компаній середнього масштабу. У цьому зв'язку керівникам підприємств необхідно розуміти, на що ці гроші витрачаються і як їх правильно витратити. Відповідно, для успішного ведення бізнесу життєво важливими є питання планування використання інформаційних технологій.

Основними напрямками діяльності в області ІТ-консалтингу є:

* формування корпоративної ІТ-стратегії;
* оцінка ефективності інвестицій в ІТ;
* вдосконалення системи управління ІТ-службою та ІТ-інфраструктурою;
* розробка концепцій створення та впровадження корпоративних інформаційних систем (КІС).

Ефективність роботи корпоративних інформаційних систем управління бізнесом (КІСУ) визначається не тільки їхніми функціональними можливостями, але і якістю впровадження. Заставою успішного створення КІСУ є коректна експертна оцінка поточного рівня автоматизації бізнес-процесів підприємства, а також визначення ключових факторів, що впливають на ефективність управління ІТ-інфраструктурою з боку ІТ-служби.

З даного проекту фахівцями в області ІТ виконуються наступні види робіт:

1. Проведення обстеження структурних підрозділів компанії, у тому числі:

- аудит інформаційних проектів;

- аудит стану парку інформаційно-обчислювальних ресурсів компанії;

- оцінка можливостей ІТ-служб з керування ІТ-інфраструктурою.

2. Розробка варіантів подальшого розвитку КІС.

3. Вироблення детальних вимог у вигляді технічного завдання до створення й впровадження інформаційної керуючої системи компанії.

## 1.2 Аналіз інформаційних потоків ІТ-компанії

Основні потоки даних пов'язані з виконанням таких дій:

* cтворення та супровід записів з відомостями про співробітників;
* пошук співробітників з певною кваліфікацією або досвідом роботи з необхідної спеціальності;
* підготовка звітів про співробітників;
* створення та супровід записів з відомостями про клієнтів компанії;
* cтворення та супровід записів з відомостями про проекти компанії;
* cтворення звітів про рух фінансів;
* cтворення та супровід записів з відомостями про офіси компанії;
* cтворення звітів з відомостями про обладнання компанії;
* cтворення та супровід записів з відомостями про ділові зустрічі працівників;
* корпоративна переписка;
* cтворення та супровід записів з відомостями про тренінги та курси для працівників компанії;

## 1.3 Завдання КІС для ІТ-компанії

КІС ІТ компанії дасть змогу отримувати довідкову інформацію щодо працівників компанії, відділів, в яких вони працюють, їхньої заробітної плати, особистих замовлень, що стосуються покращення робочих комп’ютерів, а також, відносно компанії, дозволить відслідковувати обіг грошей, обладнання, вести історію ділових зустрічей, тренінгів для працівників, проектів компанії.

База даних включатиме такі сутності:

* Працівник, містить дані про працівника, відділ в якому він працює, його заробітну плату, посаду, яку він займає.
* Проект, містить дані про проект, його тривалість, ціну, замовника, працівники відповідальні за проект.
* Місцезнаходження, включає інформацію про місцезнаходження офісів компанії.
* Кімната, містить дані про працюючих в ній людей і обладнання.
* Фінанси, містить інформацію про рух коштів компанії.
* Зарплата, дозволяє відслідковувати історію змін заробітної плати працівників.
* Відділ, містить дані про відділ, працівників, які до нього належать, та обов’язки відділу.
* Корпоративна переписка, дозволяє відслідковувати спілкування всередині компанії та за її межами.
* Зустрічі, містить інформацію про зустрічі з клієнтами.
* Тренінг, містить інформацію про тренінги для працівників компанії.
* Замовлення працівників, містить інформацію про замовлення працівників щодо покращення їхнього робочих умов.

Транзакції включають операції з даними для виконання службових завдань та нормального функціонування інформаційної системи ІТ компанії.

Групи операцій:

* Ведення обліку фінансів.
* Відслідковування кар’єрного росту працівників.
* Ведення обліку обладнання.
* Організація проектів компанії.
* Організація тренінгів для працівників.
* Міграція працівників всередині компанії.

## 1.4. Постановка завдання на проектування і розробку

В рамках даного курсового проекту повинна бути виконана розробка:

* моделі бізнес ˗ процесів корпорації відповідно методології IDEF0;
* моделі потоків даних корпорації відповідно методологій DFD і IDEF3;
* концептуальної, логічної і фізичної моделі даних згідно методології IDEF1x;
* об'єктно˗орієнтованої моделі КІС, що відповідає методології UML, передбачає створення діаграм класів та варіантів використання.

# 2. [ПРОЕКТУВАННЯ КОРПОРАТИВНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ І](#__RefHeading__7128_121574348)Т-КОМПАНІЇ

## 2.1 Вимоги до системи та її фунцкіональності

У функціональній структурі системи ІТ-компанії звичайно виділяють кілька пов’язаних підсистем, що забезпечують роботу компанії. При цьому фізично вся оброблювана інформація знаходиться на центральному сервері компанії і є базою даних, яка об'єднує набір стандартних (базових) об'єктів таких як, проекти, офіси, працівники, обладнання та ін. Оперативний доступ до бази даних ІТ-компанії здійснюється за допомогою персональних комп'ютерів працівників кожного конкретного відділу.

Підсистеми можна розділити на дві частини: основні і допоміжні. До основних відносяться:

* підсистема ведення проекту;
* підсистема роботи з фінансами;
* підсистема для роботи з даними про обладнання;
* підсистема для організації зустрічей;
* підсистема ведення особистого кабінету працівника.

Додаткові підсистеми включають: корпоративну переписку, підсистему формування звітності, тощо.

До загальних вимог до проектованого КІС також відносяться:  
 Інтерфейс користувача – має відповідати вимогам, узгодженим із замовником.

Модульність – система повинна мати можливість доповнюватись іншими розробленими додатковими модулями; має бути реалізоване безперешкодне вивантаження, завантаження та оновлення додаткових модулів, модифікація їх функціональності чи зовнішнього вигляду.

Захищеність – система повинна забезпечувати доступ до інформації в залежності від ролі користувача.

Слід передбачити захист інформації від навмисного руйнування, спотворення і випадкового допуску осіб, які не мають на це права. Доступ користувачів до системи повинен бути регламентованим у відповідності з правами, що надані цим користувачам.

Необхідно організувати узгоджену роботу користувачів і програмно ˗ технічних засобів, не допускати помилок у діяльності персоналу, що обслуговує ЕОМ. Для цього потрібно забезпечити розмежування повноважень і прав доступу користувачів системи за даними. Користувачеві в рамках системи повинна надаватися роль, яка автоматично визначається в процесі аутентифікації. З роллю повинен бути пов'язаний набір прав.

## 2.2 Вибір методологій моделювання

Мета побудови моделі деякого процесу або предметної області полягає у специфікуванні операцій і дій, які виконуються і взаємозв'язків між ними. При адекватній побудові така модель забезпечує повне уявлення про функціонування досліджуваного процесу і про всі потоки інформації та матеріалів, наявних у ньому.

При розробці моделі бізнес – процесів КІС ІТ-компанії використовуються три методології структурного аналізу і моделювання систем ˗ IDEF0, IDEF3 і DFD.

Методологія IDEF0 описує процес (операції) перетворення вхідних матеріалів або інформації в деякий результат на виході з використанням ресурсів у вигляді механізму і при виконанні умов, представлених у вигляді управління.

Методологія DFD описує зовнішні по відношенню до системи джерела і адресати даних, логічні функції, потоки даних та сховища даних, до яких здійснюється доступ. Це робить її більш зручною в порівнянні з IDEF0 для моделювання програмного забезпечення і систем документообігу.

Методологія IDEF3 описує логіку взаємодії інформаційних потоків і дозволяє графічно описати перебіг процесів у часі і відносини між процесами та об'єктами, які є частинами цих процесів. Її специфічним елементом є "перехрестя".

Моделювання з використанням усіх вищеназваних методологій засноване на використанні графічних нотацій, основу яких складають різного виду блоки і з’єднувальні дуги.

Методологія IDEF1x є методом для розробки реляційних БД і використовує умовний синтаксис, спеціально розроблений для побудови концептуальної схеми структури даних підприємства, незалежної від кінцевої реалізації БД і апаратної платформи.

Хоча термінологія IDEF1x практично збігається з термінологією IDEF1, існує ряд фундаментальних відмінностей у теоретичних концепціях цих методологій. Сутність в IDEF1x описує собою сукупність або набір екземплярів схожих за властивостями, але таких, що однозначно відрізняються один від одного за одним або кількома ознаками. Кожен екземпляр є реалізацією сутності. Таким чином, сутність в IDEF1x описує конкретний набір екземплярів реального світу, на відміну від сутності в IDEF1, яка представляє собою абстрактний набір інформаційних відображень реального світу.

Програмні засоби AllFusion ERWin Data and Process Modeler 7 (ERWin 7) дозволяють проектувати, документувати і супроводжувати БД і сховища даних, проектувати моделі бізнес ˗ процесів. Візуальне моделювання підвищує якість створюваної БД, продуктивність і швидкість її розробки.

ERWin – це не просто засіб проектування, а й інструмент розробки, здатний автоматично створювати таблиці та генерувати текст збережених процедур для всіх популярних СУБД.

Для створення діаграм роботи системи використовується методологія UML. Сьогодні UML є мовою, яка може бути використаною для опису практично будь-яких систем. Головні особливості мови, що забезпечують ефективність її практичного використання, це: акцент на семантиці на противагу нотації (головне в UML – це модель, яка містить визначення модельних елементів. Модельний елемент - це об'єкт, а не просто картинка на діаграмі. Встановлений на діаграму, елемент графічно показує свої властивості, і встановлений на різні діаграми елемент показує свої властивості з різних точок зору); широка налаштовуваність і розширюваність.

## 2.3. Функціональна модель IDEF0

У рамках методології функціонального моделювання IDEF0 бізнес ˗ процес представляється у вигляді набору функцій, які взаємодіють між собою, а також показуються інформаційні, людські і виробничі ресурси, необхідні для кожної функції.

На рис. 2.1 зображено верхній рівень декомпозиції процесу найму нового працівника ІТ-компанію.

Батьківська діаграма з назвою «Найм працівника» відображає процес в цілому (рис. 1). До даного функціональному блоку будуються такі види стрілок:

Вхід (Input) стрілка, яка входить у ліву грань функціонального блоку. Вхід показує, що потрібно для виконання функції: резюме (CV), бажана заробітна плата (desired salary).

Вихід (Output) – стрілка, що виходить із правої межі блоку. Вихід - результат функції: працівник (Employee) або кандидат (Candidat) (провалив один з етапів).

Механізм (Mechanism) стрілка, що входить в нижню грань. Механізм за допомогою чого або кого виконується функція: технічний спеціаліст (Tech specialist), фахівець з відділу кадрів (HR) та кандидат (Candidat).

Управління (Control) стрілка, яка входить у верхню грань блоку. Управління обмежує (регламентує) виконання функції: вимоги вакансії (vacancy requirements), бюджет вакансії (vacancy budget) та алгоритм/правила написання коду (Algorithm/Code Convention). Цим керуються співробітники при перевірці тестового завдання.

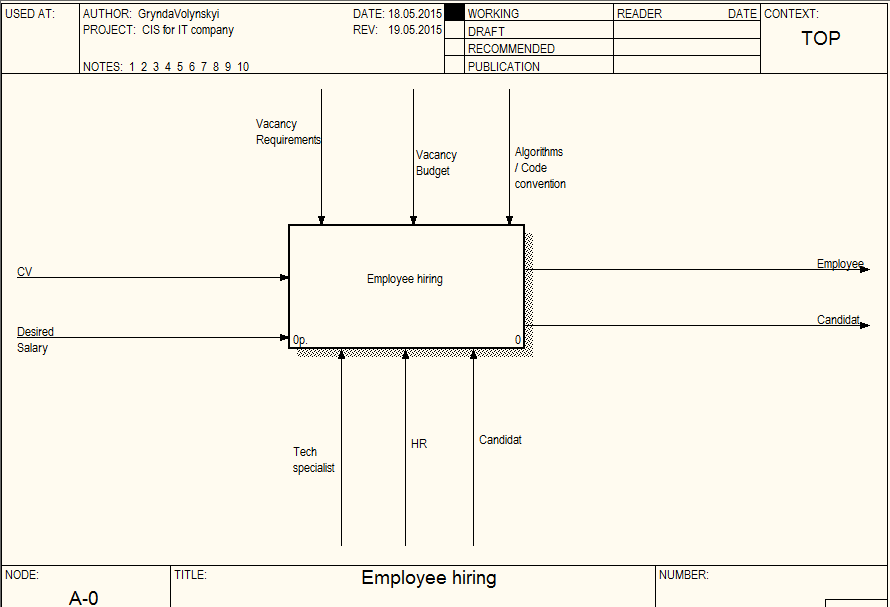


Рис. 2.1. Перший рівень декомпозиції IDEF0.

Після опису контекстної діаграми проводиться функціональна декомпозиція - процес розбивається на підпроцеси і кожен підпроцес описується окремо (діаграми декомпозиції). Потім кожен підпроцес розбивається на більш дрібні і так далі до досягнення потрібного ступеня деталізації.

Процес «Найму працівника» розбивається на:

* «HR-interview» - співбесіда з спеціалістом з відділу кадрів, за результатами якого кандидат проходить на наступний етап;
* «Tech interview» - співбесіда з технічним спеціалістом;
* «Test task solving» - вирішення тестового завдання;
* «Candidats task checking» - перевірка тестового завдання.

Варто відзначити, що на кожному етапі можливе завершення процесу. Другий рівень розбиття зображено на рис. 2.2.

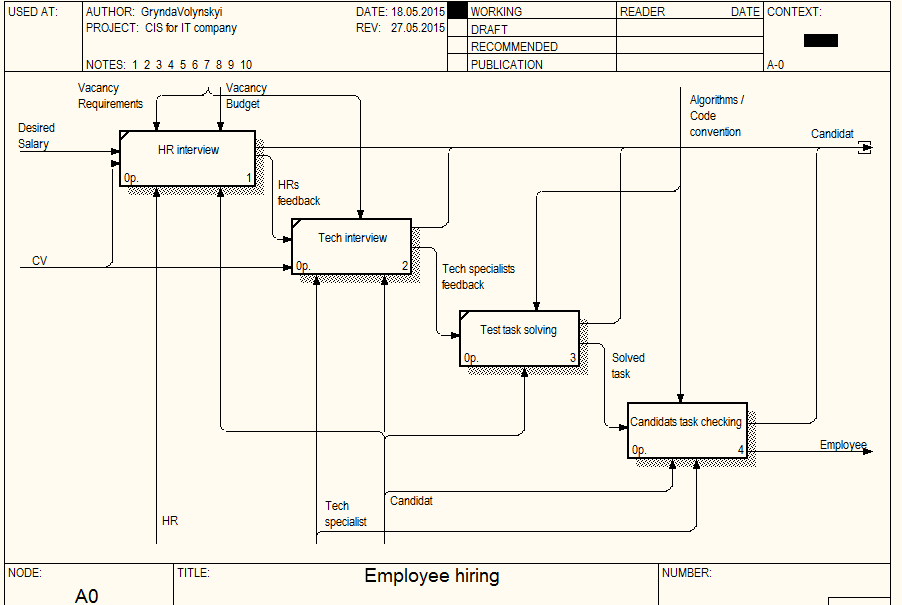


Рис. 2.2. Другий рівень декомпозиції IDEF0.

Декомпозиція IDEF0 проводилася до 3 рівня. Розглянемо для прикладу декомпозицію роботи перевірки тестового завдання (рис 2.3). З цієї діаграми ми бачимо, що процес перевірки тестового завдання складається з 3 етапів:

* «Compare actual task's output with expected one» - перевірка результатів роботи тестового завдання з очікуваними результатами;
* «Check algorithm's correctness» - перевірка коректності обраного алгоритму і його реалізації;
* «Match source code with code conventions» - перевірка відповідності написного коду правилам написання коду в компанії.

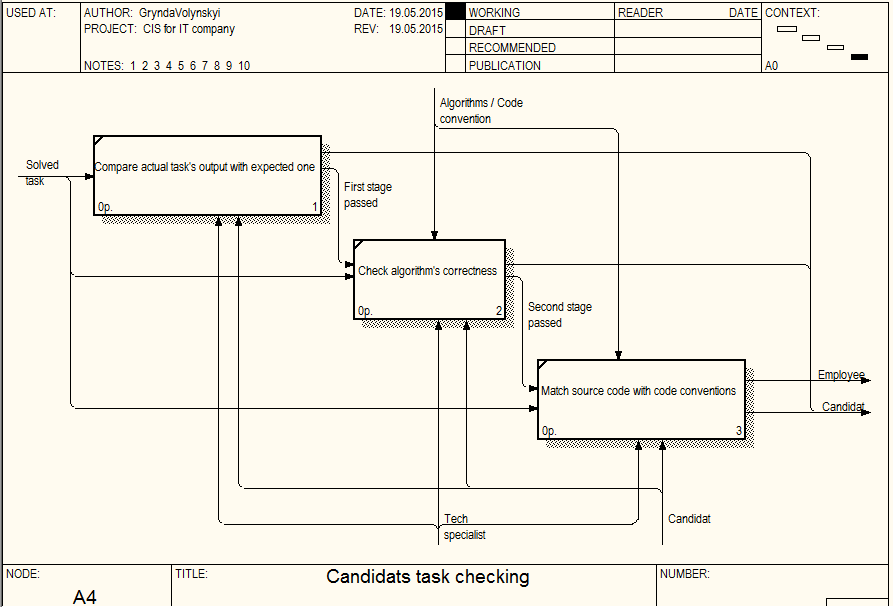


Рис. 2.3. Третій рівень декомпозиції IDEF0.

**2.4. Діаграма потоків даних DFD**

Діаграми потоків даних (DFD) використовуються для опису документообігу та обробки інформації. Нотація DFD включає такі поняття, як «зовнішня посилання» і «сховище даних», що робить її більш зручною (у порівнянні з IDEF0) для моделювання процесу замовлення обладнання для нових працівників. Перший рівень декомпозиції зображено на рис. 2.4. Цей рівень описує загальний вигляд роботи.

Другий рівень декомпозиції зображено на рис. 2.5. Він розбиває основну роботу на підроботи. Тут ми бачимо 3 підроботи, які працюють з даними, що зберігаються в базах даних:

* «Build requirements» - побудова вимог щодо нового обладнання. Працює з базою даних працівників, для отримання списку нових працівників. На виході отримуємо список необхідного обладнання;
* «Order equipment» - замовлення обладнання. На вхід приходить список необхідного обладнання. Робота включає взаємодію з базою даних коштів для перевірки поточного бюджету та внесення даних по нових замовленнях;
* «Build complex equipment» - побудова складного обладнання (комп’ютерів, столів тощо) та присвоєння його певному працівнику (використовується база даних обладнання).

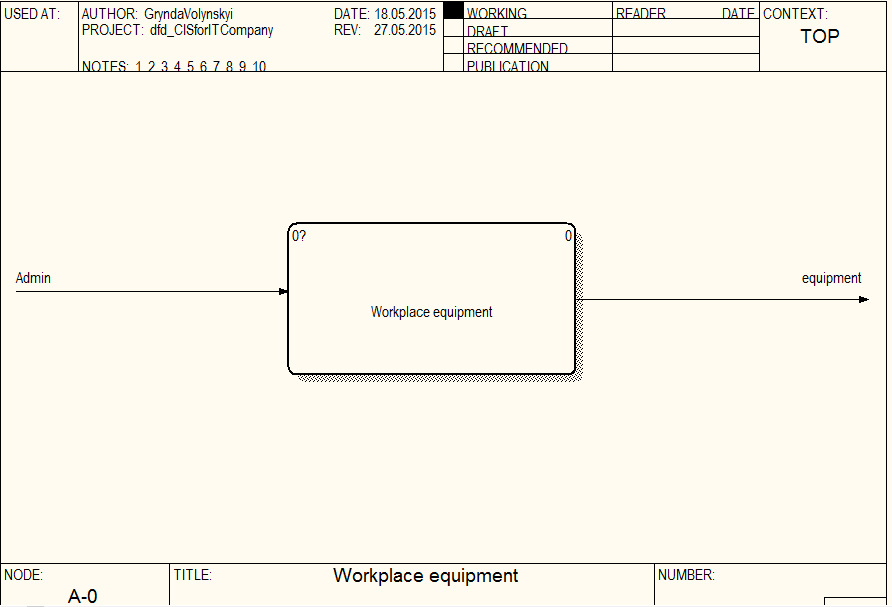


Рис. 2.4. Перший рівень декомпозиції DFD.

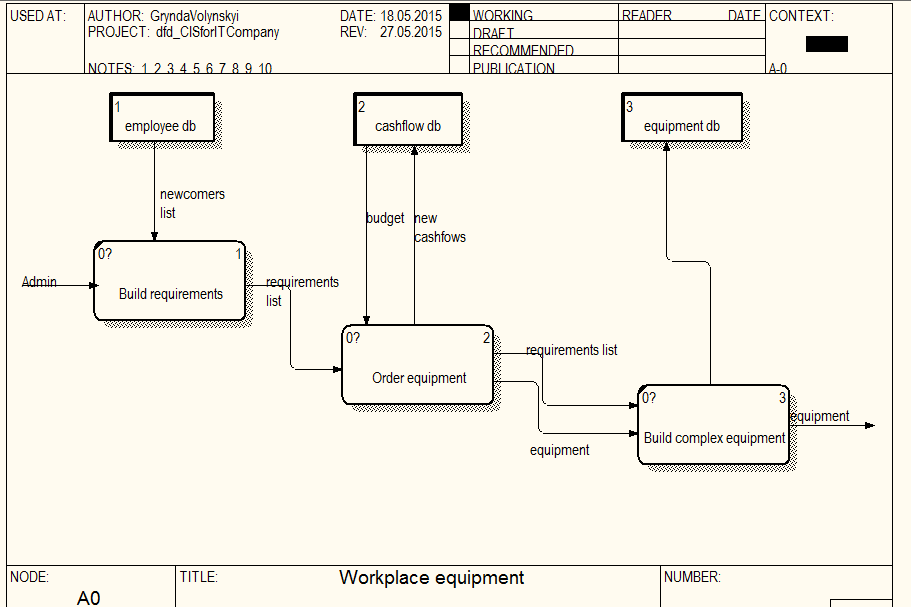


Рис. 2.5. Другий рівень декомпозиції DFD.

**2.5. Діаграма процесів IDEF3.**

Для опису логіки взаємодії інформаційних потоків більше підходить workflow diagramming. Діаграми Workflow можуть бути використані в моделюванні бізнес ˗ процесів для аналізу завершеності процедур обробки інформації.

На рис.2.6. представлена діаграма декомпозиції «Перевірка тестового завдання» в нотації IDEF3.

1. Після порівняння результатів роботи тестового завдання з очікуваними результатами застосовується оператор АБО:

* Якщо результати не зійшлись – завдання провалено;
* Якщо результати зійшлись – переходимо до п. 2.

1. Запускаються два паралельних процеси:

* Перевірка правильності алгоритму;
* Перевірка відповідності написаного коду рекомендаціям написання коду;

Кожен з процесів може використовує оператор АБО. При невиконанні одного з процесів завдання вважається проваленим. Якщо ж обидва процеси закінчились успішно, то завдання вважається виконаним і кандидат приймається на роботу.

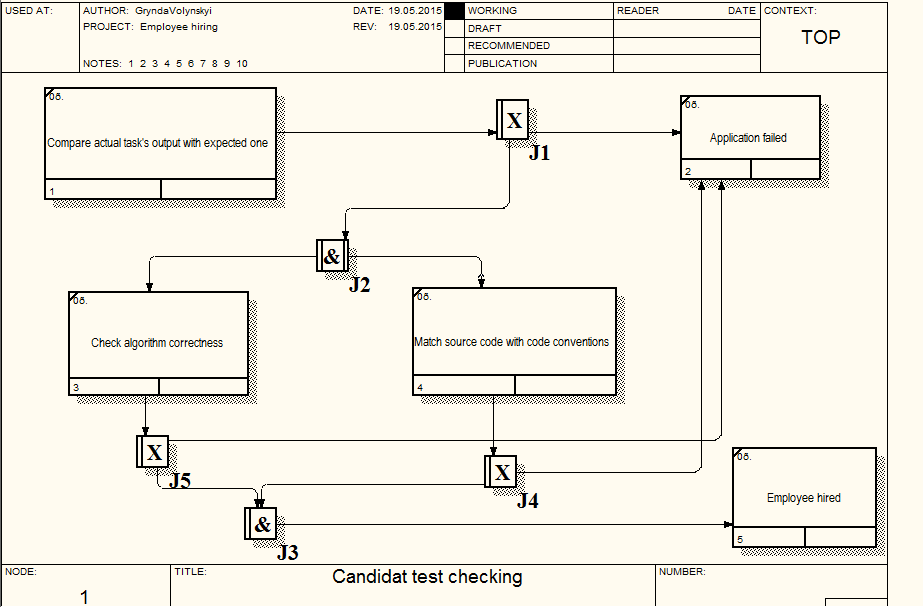


Рис. 2.6. Діаграма процесів IDEF3

* 1. **Моделювання даних IDEFx1**
     1. Логічна модель даних

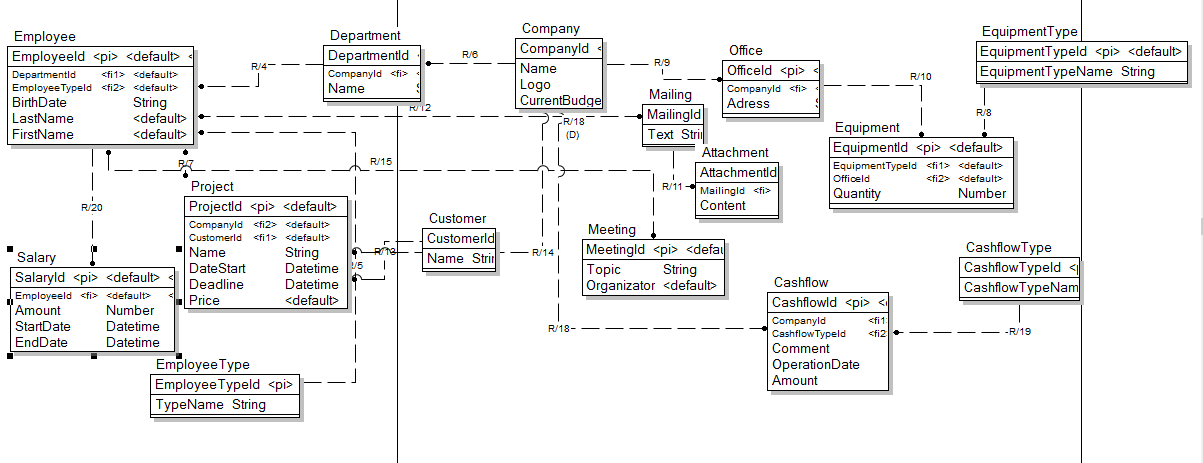


Рис. 2.7. Логічна модель даних

* + 1. Фізична модель даних

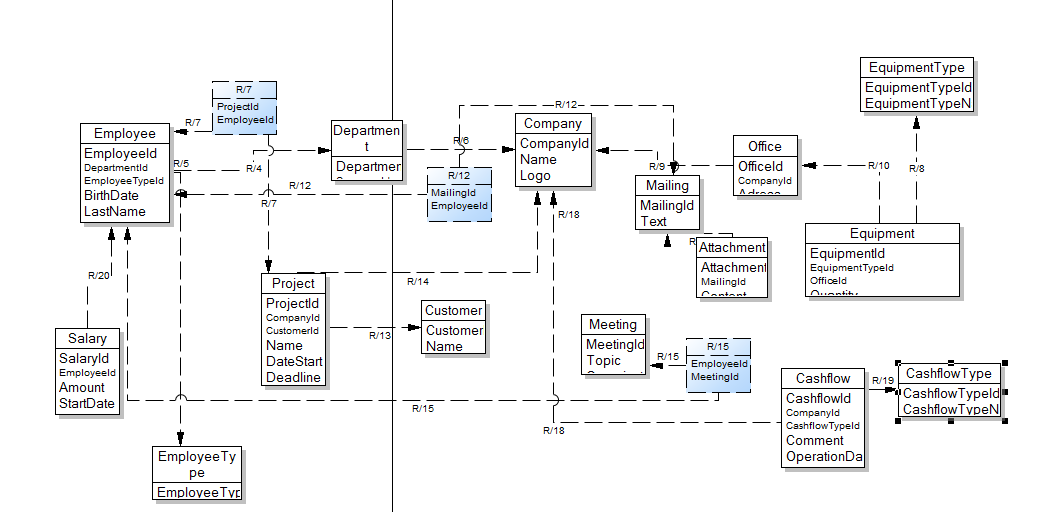


Рис. 2.8. Фізична модель даних

* 1. **Об’єктно-орієнтована модель даних**
     1. Діаграма варіантів використання.

*Діаграма варіантів використання* є графом, що складається з множини акторів, прецедентів (варіантів використання) обмежених границею системи (прямокутник), асоціацій між акторами та прецедентами, відношень серед прецедентів, та відношень узагальнення між акторами. Діаграми прецедентів відображають варіанти використання системи. Суть даної діаграми полягає в наступному: проектована система представляється у вигляді безлічі сутностей чи акторів, взаємодіючих із системою за допомогою так званих варіантів використання. Варіант використання (UseCase) слугує для опису сервісів, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, чинений системою при діалозі з актором. При цьому нічого не говориться про те, яким чином буде реалізована взаємодія акторів із системою. Діаграма варіантів використання КІС ІТ-компанії представлена ​​на рис. 2.9.

Актори: користувач, директор, співробітник, адміністратор.

В залежності від ролі акторам доступні наступні дії:

*Користувач:*

* авторизація;
* особистий кабінет;
* перегляд історії;
* перегляд інформації про заробітню платню.

*Співробітник:*

* створення звітів;
* отримання списку проектів;
* замовлення обладнання;
* призначення зустрічі;

*Директор:*

* створення нового відділу;
* отримання списку працівників;
* отримання історії працівників;
* генерація фінансового звіту;
* звільнення співробітника;
* отримання списку проектів;
* призначення бюджету;
* редагування заробітньої платні;
* додавання проекту;
* переміщення співробітників між відділами;
* видалення відділу;
* найм працівників.

*Адміністратор:*

* замовлення обладнання;
* розподіл обладнання;
* отримання списку необхідного обладнання.

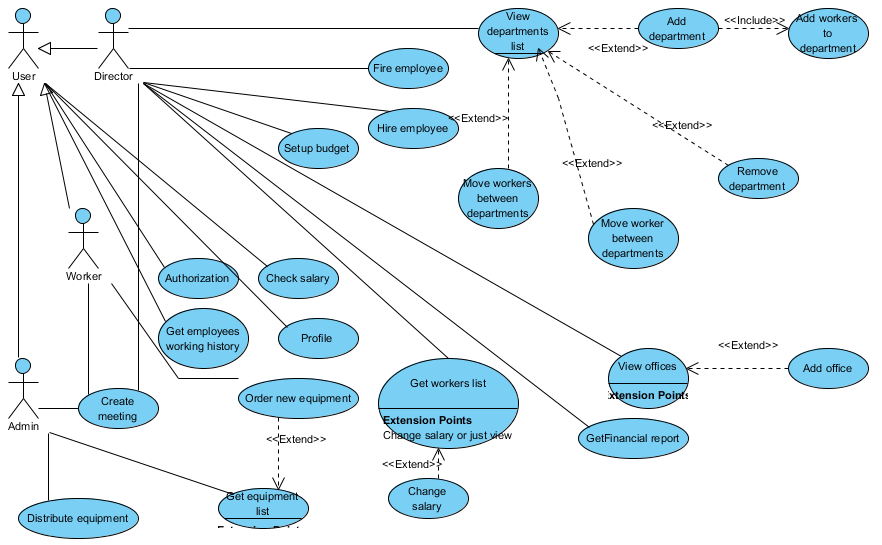


Рис. 2.9. Діаграма варіантів використання

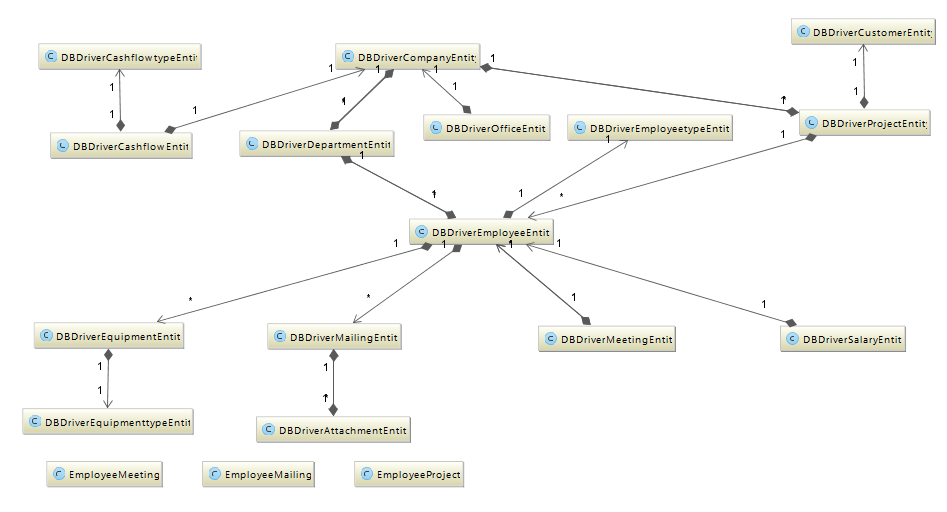


Рис. 2.10. Діаграма класів

## 3. РЕАЛІЗАЦІЯ СИСТЕМИ

## 3.1. Побудова моделі. Реалізація класів сутностей.

Для генерації класів сутностей було використано стандартні засоби Intellij IDEA для імпорту бази даних, згенеровано JPA анотації. Використано СУБД PosgreSQL 9.3.

На рис. 3.1. зображено створення нового підключення до бази даних.

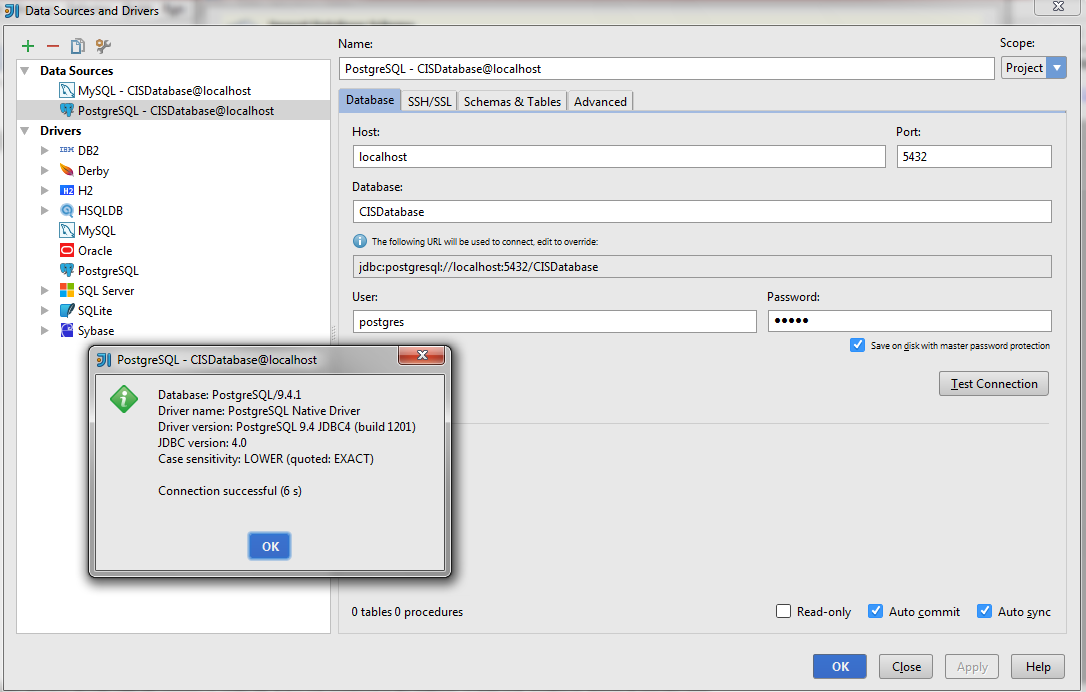


Рис. 3.1. Створення нового підключення до бази даних

На рис. 3.2. зображено вибір класів, які мають бути згенеровані.

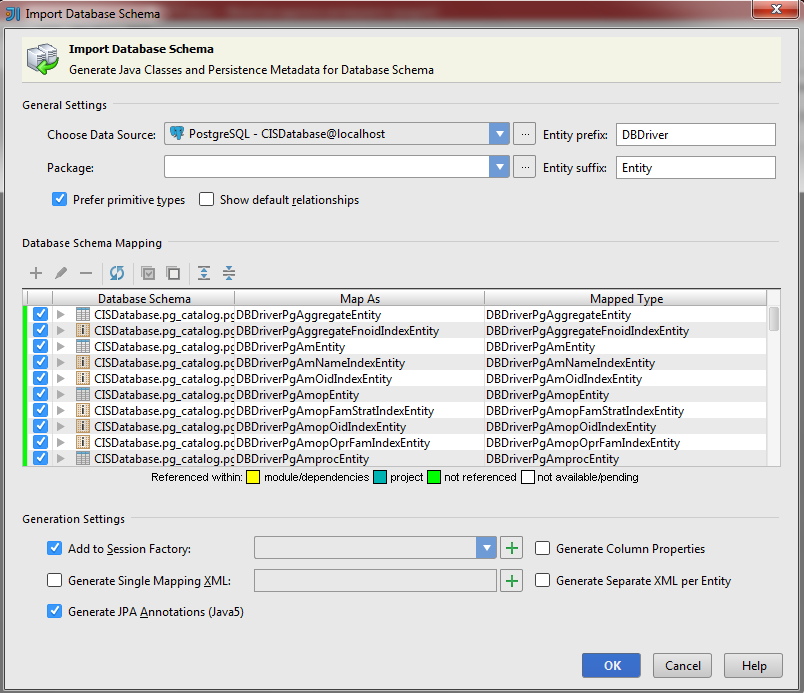


Рис 3.2. Вибір класів

Згенеровані класи містять посилання на поля з таблиць бази даних.

|  |
| --- |
| DBDriverEmployeeEntity.java |
| **package** cis.itcompany;  **import** javax.persistence.\*;  @Entity @Table(name = **"employee"**, schema = **"public"**, catalog = **"Database4"**) **public class** DBDriverEmployeeEntity {  **private int employeeid**;  **private** String **birthdate**;  **private** String **lastname**;  **private** String **firstname**;   **public** DBDriverDepartmentEntity **departmentEntity**;  **public** DBDriverEmployeetypeEntity **employeetypeEntity**;   **public** DBDriverEquipmentEntity[] **equipment**;  **public** DBDriverMailingEntity[] **mail**;   @Id  @Column(name = **"employeeid"**)  **public int** getEmployeeid() {  **return employeeid**;  }   **public void** setEmployeeid(**int** employeeid) {  **this**.**employeeid** = employeeid;  }   @Basic  @Column(name = **"birthdate"**)  **public** String getBirthdate() {  **return birthdate**;  }  **public void** setBirthdate(String birthdate) {  **this**.**birthdate** = birthdate;  }   @Basic  @Column(name = **"lastname"**)  **public** String getLastname() {  **return lastname**;  }   **public void** setLastname(String lastname) {  **this**.**lastname** = lastname;  }   @Basic  @Column(name = **"firstname"**)  **public** String getFirstname() {  **return firstname**;  }   **public void** setFirstname(String firstname) {  **this**.**firstname** = firstname;  }   @Override  **public boolean** equals(Object o) {  **if** (**this** == o) **return true**;  **if** (o == **null** || getClass() != o.getClass()) **return false**;   DBDriverEmployeeEntity that = (DBDriverEmployeeEntity) o;   **if** (**employeeid** != that.**employeeid**) **return false**;  **if** (**birthdate** != **null** ? !**birthdate**.equals(that.**birthdate**) : that.**birthdate** != **null**) **return false**;  **if** (**firstname** != **null** ? !**firstname**.equals(that.**firstname**) : that.**firstname** != **null**) **return false**;  **if** (**lastname** != **null** ? !**lastname**.equals(that.**lastname**) : that.**lastname** != **null**) **return false**;   **return true**;  }   @Override  **public int** hashCode() {  **int** result = **employeeid**;  result = 31 \* result + (**birthdate** != **null** ? **birthdate**.hashCode() : 0);  result = 31 \* result + (**lastname** != **null** ? **lastname**.hashCode() : 0);  result = 31 \* result + (**firstname** != **null** ? **firstname**.hashCode() : 0);  **return** result;  } } |

Лістинг 3.1. Приклад класу-сутності

|  |
| --- |
| DBDriverCompanyEntity.java |
| **package** cis.itcompany;  **import** javax.persistence.\*;  @Entity @Table(name = **"company"**, schema = **"public"**, catalog = **"Database4"**) **public class** DBDriverCompanyEntity {  **private int companyid**;  **private** String **name**;  **private** String **logo**;  **private** Integer **currentbudget**;   **public** DBDriverProjectEntity[] **projects**;  **public** DBDriverDepartmentEntity[] **departments**;    @Id  @Column(name = **"companyid"**)  **public int** getCompanyid() {  **return companyid**;  }   **public void** setCompanyid(**int** companyid) {  **this**.**companyid** = companyid;  }   @Basic  @Column(name = **"name"**)  **public** String getName() {  **return name**;  }   **public void** setName(String name) {  **this**.**name** = name;  }   @Basic  @Column(name = **"logo"**)  **public** String getLogo() {  **return logo**;  }   **public void** setLogo(String logo) {  **this**.**logo** = logo;  }   @Basic  @Column(name = **"currentbudget"**)  **public** Integer getCurrentbudget() {  **return currentbudget**;  }   **public void** setCurrentbudget(Integer currentbudget) {  **this**.**currentbudget** = currentbudget;  }   @Override  **public boolean** equals(Object o) {  **if** (**this** == o) **return true**;  **if** (o == **null** || getClass() != o.getClass()) **return false**;   DBDriverCompanyEntity that = (DBDriverCompanyEntity) o;   **if** (**companyid** != that.**companyid**) **return false**;  **if** (**currentbudget** != **null** ? !**currentbudget**.equals(that.**currentbudget**) : that.**currentbudget** != **null**)  **return false**;  **if** (**logo** != **null** ? !**logo**.equals(that.**logo**) : that.**logo** != **null**) **return false**;  **if** (**name** != **null** ? !**name**.equals(that.**name**) : that.**name** != **null**) **return false**;   **return true**;  }   @Override  **public int** hashCode() {  **int** result = **companyid**;  result = 31 \* result + (**name** != **null** ? **name**.hashCode() : 0);  result = 31 \* result + (**logo** != **null** ? **logo**.hashCode() : 0);  result = 31 \* result + (**currentbudget** != **null** ? **currentbudget**.hashCode() : 0);  **return** result;  } } |

Лістинг 3.2. Приклад класу-сутності

## 3.2. Реалізація DAO-класів.

DAO (data access object) – об’єкт, що надає абстрактний інтерфейс до деяких видів баз даних чи механізмів персистентності, реалізуючи певні операції без розкриття деталей бази даних. Він надає відображення від програмних викликів до рівня персистентності. Така ізоляція розділює запити до даних в термінах предметної області та їх реалізацію засобами СУБД.

Цей шаблон проектування можна застосовувати до більшості мов програмування, видів програмного забезпечення з потребою персистентності та більшості типів баз даних, але він традиційно асоціюється з застосунками Java EE та реляційними БД, доступ до яких здійснюють через JDBC API.

Перевагою використання об’єкту доступу до даних є досить просте розділення двох частин програми: бізнес логіки та логіки персистентності. При такій реалізації зміна бізнес логіки зовсім не буде впливати на роботу механізмів персистентності, а зміна схеми даних чи способу їх зберігання не впливатиме на роботу бізнес логіки, якщо інтерфейс реалізований правильно.

Інтерфейс GenericDao<T> містить в собі визначення 5 методів спільних для всіх реалізацій. Також він забезпечує перевірку типів на стадії компіляції за рахунок використання Generic типу T.

|  |
| --- |
| GenericDao.java |
| public interface GenericDao<T> extends Serializable {  public void create(T e);  public T read(Object id);  public void update(T e);  public void delete(T e);  } |

Лістинг 3.3. Узагальнений інтерфейс роботи з даними

## Абстрактний клас AbstractDao<T> реалізує ці 5 методів з допомогою класу EntityManager. Також додається абстрактний метод getEntityClass, що повертає об’єкт метаданих класу-сутності, з яким працює та чи інша реалізація DAO.

|  |
| --- |
| AbstractDao.java |
| public abstract class AbstractDao<T> implements GenericDao<T> {  private String persistentName;  private EntityManagerFactory factory;  public void createEntityManagerFactory(String name) {  this.persistentName = name;  factory = Persistence.createEntityManagerFactory(persistentName);  }  public EntityManager getEntityManager() {  return factory.createEntityManager();  }  public void create(T e) {  EntityManager manager = factory.createEntityManager();  manager.getTransaction().begin();  manager.persist(e);  manager.getTransaction().commit();  manager.close();  }  @SuppressWarnings("unchecked")  public T read(Object id) {  T readObject = null;  EntityManager manager = factory.createEntityManager();  manager.getTransaction().begin();  readObject = (T) manager.find(getEntityClass(), id);  manager.getTransaction().commit();  manager.close();  return readObject;  }  public void update(T e) {  EntityManager manager = factory.createEntityManager();  manager.getTransaction().begin();  manager.refresh(e);  manager.getTransaction().commit();  manager.close();  }  public void delete(T e) {  EntityManager manager = factory.createEntityManager();  manager.getTransaction().begin();  manager.remove(e);  manager.getTransaction().commit();  manager.close();  }  public abstract Class<?> getEntityClass();  } |

Лістинг 3.4. Базовий клас роботи з даними

Кожна з конкретних реалізацій DAO розширює вище вказаний клас, реалізуючи метод для отримання метаданих, уточнюючи тип «T», а також може додавати власні методи.

|  |
| --- |
| ProjectDao.java |
| public class ProjectDao extends AbstractDao<DBDriverProjectEntity> {  public ProjectDao() {  createEntityManagerFactory(DBDriverProjectEntity.nameInDB);  }  @Override  public void update(DBDriverProjectEntity e) {  e.setDatestart(new Date(Calendar.getInstance().getTime().getTime()));  super.update(e);  }  public Collection<DBDriverProjectEntity> getAllProjects() {  Collection<DBDriverProjectEntity> projects = null;  EntityManager entityManager = getEntityManager();  try {  entityManager.getTransaction().begin();  StringBuffer queryString = new StringBuffer("select p from").append(DBDriverProjectEntity.nameInDB).append("p");  Query query = entityManager.createQuery(queryString.toString());  projects = (Collection<DBDriverProjectEntity>) query.getResultList();  entityManager.getTransaction().commit();  } catch (Exception e) {  e.printStackTrace();  } finally {  if (entityManager != null && entityManager.isOpen()) {  entityManager.close();  }  }  return projects;  }  @Override  public void create(DBDriverProjectEntity e) {  e.setDatestart(new Date(Calendar.getInstance().getTime().getTime()));  super.create(e);  }  @Override  public Class<?> getEntityClass() {  return DBDriverProjectEntity.class;  }  } |

Лістинг 3.5. Приклад конкретної реалізації DAO

## 3.3. Реалізація додатку.

Для реалізації інформаційної системи обрано мову Java. Java - високорівнева мова програмування загального призначення, орієнтована на підвищення продуктивності розробника. Основною її перевагою є її кросплатформенність.

В якості СУБД обрано PostgreSQL 9.3.

PostgreSQL є дуже гнучким рішенням і має значну кількість налаштувань. Основні можливості:

- максимальний розмір таблиці 32 Тбайт;

- підтримка БД практично необмеженого розміру;

- потужні та надійні механізми транзакцій і реплікації;

- наслідування таблиць;

- має велику кількість вбудованих типів даних;

- підтримка індексів таких типів: B-дерево, хеш, R-дерево, GiST;

- підтримує одночасну модифікацію БД кількома користувачами за допомогою механізму MultiversionConcurrencyControl (MVCC);

- підтримка ACID;

- підтримка регулярних виразів;

- контроль цілісності;

- реплікація.

Для реалізації клієнтської частини розробленої системи було використано javascript фреймворк AngularJS. Взаємодія з серверною частиною додатку відбувається через Restful сервіси. На рис. 3.3 та рис. 3.4 подано графічний інтерфейс розробленої корпоративної інформаційної системи ІТ компанії.

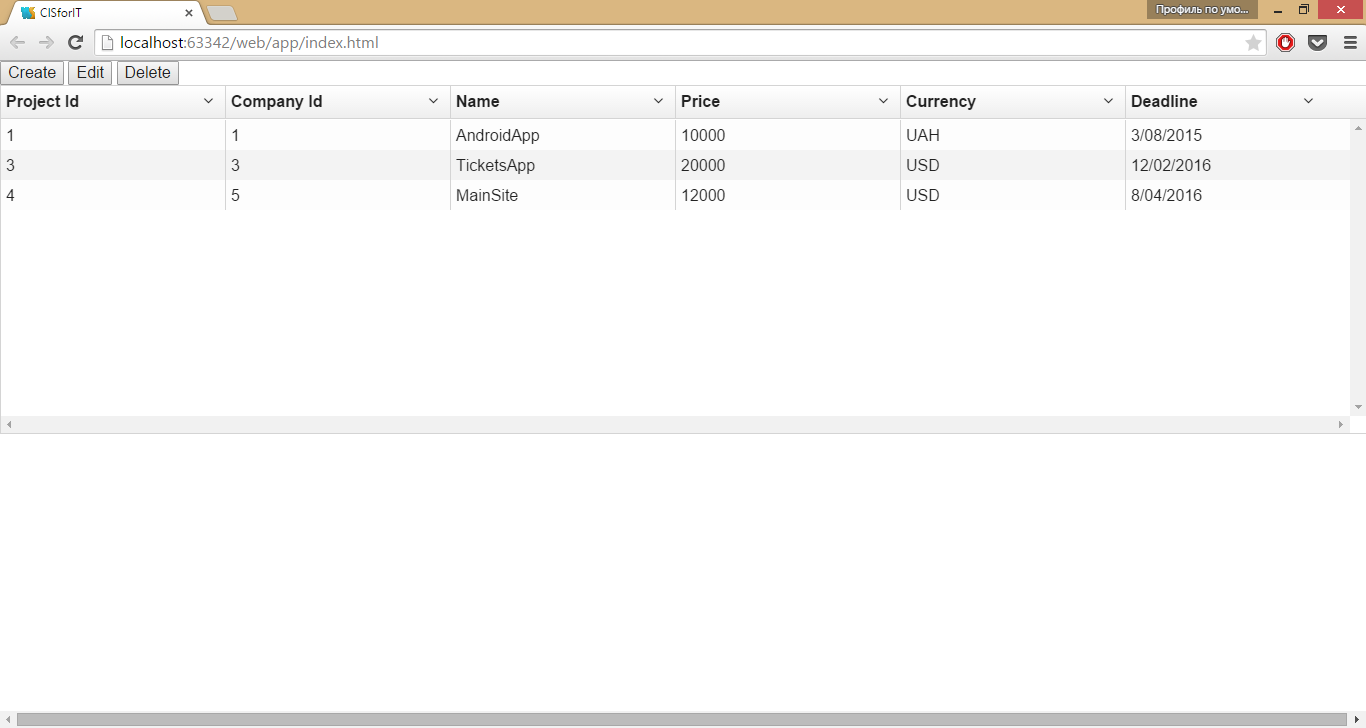


Рис. 3.3. Форма перегляду проектів

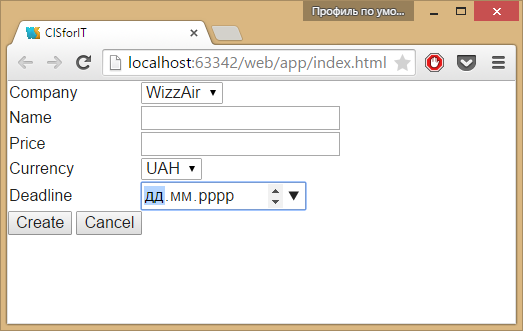


Рис. 3.4. Форма додавання нового проекту

## ВИСНОВКИ

У ході виконання роботи було проаналізовано предметну область, побудовано модель предметної області та даних цієї області. Для побудови моделі процесів використовувалися нотації IDEF0, DFD, IDEF3. Для побудови моделей даних у системі було використано нотацію IDEF1x, а всі роботи з побудови виконувалися в програмних пакетах AllFusionProcessModeler і ErwinDataModeler. Також було побудовано UML-діаграми, зокрема діаграму варіантів використання, діаграму класів та діаграму послідовностей. Для цих цілей було використано середовище Visual Paradigm for UML.

Розроблено інформаційну систему для управління ІТ компанією. Для цього було використано такі інструменти:

- Java в якості мови для серверної частини додатку;

- PostgreSQL 9.3 в якості СУБД;

# СПИСОК ЛІТЕРАТУРИ

1. Проектування корпоративних інформаційних. Методичні вказівки до курсового проектування з дисципліни «Проектування корпоративних інформаційних систем» для студентів спеціальності 7.080403 ‒ Програмне забезпечення комп'ютерних систем, 8.080403 ‒ Програмне забезпечення комп'ютерних систем. / Укл. В.І. Павловський, Д.В. Побєда. ‒ Київ: ЧДТУ, 2012. ‒ 90 с.
2. Структурний аналіз і моделювання в середовищі ComputerAssociateAllFusionProcessModeler. Методичні вказівки до вивчення та лабораторних робіт з дисципліни «Проектування корпоративних інформаційних» для студентів спеціальності 7.080403 – Програмне забезпечення комп’ютерних систем, 8.080403 – Програмне забезпечення комп’ютерних систем. / Укл. В.І. Павловський, Д.В. Побєда. – Київ: ЧДТУ, 2011. ‒ 216 с.
3. Моделювання інформаційних систем засобами UML. Методичні вказівки до лабораторних робіт з дисципліни «Проектування корпоративних інформаційних систем» для студентів спеціальностей “Програмна інженерія” / Укл. В.І. Павловський, Д.В. Побєда, М.В. Харченко. – Київ: НТУУ "КПІ", 2010. ‒ 78 с. рос. мовою.
4. Unified Modeling Language [Електронний ресурс] – Режим доступу : http://en.wikipedia.org/wiki/Unified\_Modeling\_Language. – 15.04.2015.