МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет Комп'ютерних наук,
управління та адміністрування

Кафедра Інформаційних технологій

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: Розробка інформаційної системи моніторингу та управління доступом до систем забезпечення функціонування житлового комплексу

Виконав студент 2 курсу групи МІС-20
спеціальності 122 Комп'ютерні науки

Ряшенцев Вадим Сергійович

Керівник к.т.н., доцент
Фразе-Фразенко Олексій Олексійович

Рецензент засновник Компанії «UALinux»
Попов Володимир Леонідович

Одеса 2021

АНОТАЦІЯ

на магістерську кваліфікаційну роботу

«Розробка інформаційної системи моніторингу та управління доступом до систем забезпечення функціонування житлового комплексу»,

студентка Ряшенцева Вадима Сергійовича

Магістерська робота присвячена розгляду системи управління будівельним комплексом з виділенням внутрішніх і зовнішніх факторів прямого і непрямого впливу на систему управління, а також сучасним енергозберігаючим технологіям, що використовуються для комфортного проживання. Описано спосіб реалізації та вказано основні напрямки розвитку сучасних технологій пов’язаних з даною галуззю. Розказано про управління системою «Розумний житловий комплекс». Описано одночасно комфортний та економний шлях охорони ЖК з єдиною базою даних для всіх користувачів, дистанційне керування входом\виходом, система для управління комунальними послугами.

Інформаційно-аналітичні системи (ІАС) та інформаційно-комунікаційні технології (ІКТ) на сьогоднішній день займають головне місце в розвитку та модернізації основних сфер діяльності міста. Особлива увага надається впровадженню ІКТ у житлово-комунальному комплексі (ЖКК). Інформаційно-аналітична система ЖКК має бути інструментом для вирішення завдань підвищення енергоефективності та безпеки мешканців житлового комплексу, автоматизації обліку витрат ресурсів, застосування новітніх технологій моніторингу.

Основна ідея та концепція «розумного ЖК» була використана та реструктуризована для запровадження більших можливостей для програми розрахованої на використання великої кількості користувачів, а також спрощення ведення облікових записів та моніторингу інформації, що надходить.

Для виконання всіх поставлених задач попередньо було проведено дослідження, завдяки якому були встановлені потреби мешканців, а також слабкі місця сучасних систем управління та моніторингу житловими комплексами. Далі вказані можливості, технології, принципи створення комунікаційних систем та обладнання автоматизації. Основні переваги цього методу полягають у теперішньому розвитку інформаційно-комунікаційних технологій.

Магістерська кваліфікаційна робота містить 72 сторінки, 14 рисунків та 17 джерел.

КЛЮЧОВІ СЛОВА: АНАЛІЗ, ЖК, ЕК, КОМУНАЛЬНІ ПОСЛУГИ, КОРИСТУВАЧ, ВХОДИ ТА ВИХОДИ, ПОВІДОМЛЕННЯ.

SUMMARY

for a master's degree

" Development of An Information System for Monitoring and Control of Access to Systems for Ensuring the Functioning of the Residential Complex", students of Riashentsev Vadym

The master's thesis is devoted to the management system of the building complex with the selection of internal and external factors of direct and indirect influence on the management system, as well as modern energy-saving technologies used for comfortable living. The method of implementation is described and the main directions of development of modern technologies which are related to this industry are indicated. The management of the "Smart Housing Complex" system is mentioned. Both comfortable and economical ways to protect the LCD with a single database for all users, remote control of input / output, and a system for managing utilities are present in this work.

Information-analytical systems (IAS) and information-communication technologies (ICT) today occupy a major place in the development and modernization of the main areas of the city. Particular attention is paid to the introduction of ICT in housing and communal services (HCS). The information and analytical system of housing and utilities should be the main tools for solving problems of improving the energy efficiency and safety of residents of the residential complex, automation of accounting for resource costs, and using the latest monitoring technologies.

The basic idea and concept of "smart LCD" was used and restructured to introduce greater capabilities for the program designed to use a large number of users, as well as to simplify account management and monitoring of incoming information.

To fulfill all the set tasks, a preliminary study was conducted, which identified the needs of residents, as well as the weaknesses of modern management systems and monitoring of housing estates. The following are the capabilities, technologies, principles of communication systems and automation equipment. The main advantages of

this method are the current development of information and communication technologies.

The master's thesis contains 72 pages, 14 figures and 17 sources.

KEYWORDS: ANALYSIS, RESIDENTIAL COMPLEX, ELECTRONIC KEY, UTILITIES, USER, INPUTS AND OUTPUTS, MESSAGES.

ЗМІСТ

Перелік скорочень, умовних позначень і термінів……..……...8
Вступ.....9
1 Аналіз предметної області 13
1.1 Аналіз інструментів розробки 13
1.2 Вибір платформи розробки системної частини 14
1.2.1 Огляд .NET.....14
1.2.2 Огляд Java 16
1.2.2 Огляд PHP 17
1.3 Огляд язика програмування для розробки КІ 19
1.5 Вибір версії ПЗ 20
1.6 Вибір СУБД 21
1.6.1 Огляд MySQL 21
1.6.2 Огляд SQLite.....22
1.6.3 Огляд Oracle.....24
1.7 Вибір IDE 25
1.7.1 Розгляд Eclipse.....25
1.7.2 Розгляд IntelliJ Idea 28
1.7.3 Розгляд NetBeans.....31
2 Вибір додаткового інструментарію для розробки пп.....34
2.1 Огляд інструментарію для розробки системної частини.....34
2.2 Огляд інструментарію для розробки КІ.....39
3 Проектна частина.....47

3.1 Постановка завдань 47
3.2 UML діаграма проекту 47
3.3Діаграма прецедентів 49
3.4 Структура БД 52
3.5 Зв’язок між лицьовою та системною частинами 55
3.5.1 Розробка REST API на лицьовій стороні ПП 56
3.5.2 Розробка REST API на системній стороні ПП 57
4 Розробка програмних елементів 59
4.1 Розробка інтерфейсу гостя 59
4.1.1 Особистий кабінет гостя. 59
4.2 Розробка інтерфейсу мешканцю 59
4.2.1 Вітальний екран мешканцю 60
4.2.2 Комунальні послуги мешканцю 61
4.2.3 Повідомлення мешканцю 61
4.2.2 Входи та виходи мешканцю 62
4.3 Розробка інтерфейсу охоронця 63
4.3.1 Входи та виходів охоронця 63
4.4 Розробка інтерфейсу адміністратору 64
4.4.1 Розробка інтерфейсу електронних ключів адміністратора 64
4.4.2 Розробка інтерфейсу адрес 65
4.4.3 Розробка інтерфейсу користувачів 66
Висновки 67
Перелік джерел посилання 69

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

БД – база даних

ПЗ – програмне забезпечення

ПП – програмний продукт

ПК – персональний комп’ютер

СУБД – система управління базами даних

JVM – Java Virtual Machine, віртуальна машина Java

IDE – Integrated Development Environment, інтегроване середовище розробки.

ЖК – Жилий комплекс

КІ – Користувальницький інтерфейс

ВСТУП

Сучасний рівень розвитку інформаційної техніки і технологій дозволяє створювати пристрої і системи різного призначення і масштабу для широкого спектру завдань моніторингу. Моніторинг просторово розподілених параметрів передбачає істотне використання мережевих технологій, де останнім часом також спостерігається вражаючий прогрес.

Автоматизація процесів при експлуатації об'єктів нерухомості, або так званий "розумний будинок" та "розумний квартал" - це багатогранне управління, в якому представлені різні види рішень для того, щоб покращити рівень комфорту та безпеки мешканців. Автоматизація у житловому комплексі (далі ЖК) складається не тільки з систем безпеки, а також з контролю та обліку витрат ресурсів. Для людини, що мешкає у ЖК з такими автоматизованими процесами витрачений час на, наприклад, комунальні послуги зводиться до мінімума.

Також не менш важливим фактором є безпека мешканців. Для цього важливо виконати наступні дії:

1. Оцінювання ризиків безпеки

Оцінка ризиків безпеки – ефективний спосіб підвищити безпеку в будівлі, проте займатися цим мають професіонали, які розуміють усі нюанси та можуть об'єктивно оцінити ситуацію. Співробітник служби безпеки повинен визначати вразливі місця в системі безпеки та вносити пропозиції щодо покращення. Оцінка ризиків безпеки допоможе визначити найкращі методи підвищення безпеки.

2. Встановлення системи відеоспостереження

Система відеоспостереження – це відмінний спосіб стримати злочинність, а у випадку, якщо злочин все ж таки відбувся, то надати докази правопорушення для співробітників поліції та допомогти їм оперативно знайти злочинця. Якщо система відеоспостереження в будівлі вже є, слід подумати про її модернізацію –

можливо, деякі її елементи застаріли та не мають належного функціоналу, наприклад нічна зйомка чи розпізнавання осіб. В іншому випадку камери можуть стояти не в правильних місцях, утворюючи мертві зони. При цьому самих камер має бути достатньо, щоб кожну частину будівлі було видно.

3. Наймання працівників служби безпеки

Співробітники безпеки – надійний спосіб підвищити безпеку на території житлового комплексу та можливість надати мешканцям спокій. Можна розмістити контрольний пункт на спільній прохідній або у кожному окремому під'їзді, вестибюлі або забезпечити патрулювання території ЖК. Співробітники служби безпеки можуть реагувати на будь-які проблеми – контролювати сторонніх, діяти у разі небезпечної ситуації.

4. Встановлення контролю доступу

Системи контролю доступу – чудовий спосіб підвищити безпеку на території житлового комплексу. Мешканці можуть використовувати картки-ключі, щоб потрапити до будівлі, скористатися ліфтом та навіть увійти до своїх квартир. Карти-ключі складніше скопіювати, ніж звичайні ключі, а при втраті замінити нові і змінити коди, що додає додатковий рівень безпеки. Також можна легко вмикати та вимикати доступ до окремих карток-ключів.

Найважливішим чинником з усіх перелічених вище, безумовно, є безпека. У продовженні цієї роботи ми структуровано підійдемо до детального вивчення аспекту системи автоматизованого моніторингу, заснованої на технологіях розподіленого зберігання, обробки даних, а також дистанційного телекомунікаційного доступу до них, націленого на підвищення ефективності управління при поєднанні спеціалізованих і суто інформаційних підходів в охороні здоров'я або інших сферах діяльності.

Система управління ЖК є додатком, де у мешканця є свій особистий кабінет, через який він має доступ до зміни особистої інформації, редагування комунальних послуг та додавання повідомлень. Крім цього, для охоронної системи

певного ЖК у закритому доступі є інформація про зміну статусу входу або виходу, використання журналу з даною інформацією, а також для адміністратора - управління комунальними послугами, обліковими записами, електронними ключами, інформацією про вхід та вихід, повідомлення та видача прав користувачам.

Головним завданням даної роботи є спрощення експлуатаційного навантаження як персоналу так і мешканців будівлі. Нашою основною метою є консолідація управлінських рішень. Управління та обслуговування будівель – це організована та ефективна система операцій з технічного обслуговування, яка створена для вирішення проблем, пов’язаних з утриманням будівлі. Основною метою технічного обслуговування є захист будівлі на її початковій стадії та збереження вартості інвестицій у нерухомість. Підтримка будівлі в такому стані, в якому вона продовжує виконувати своє призначення і впевненість, що вона має привабливий зовнішній вигляд, також є важливими факторами, які стають можливими завдяки належному обслуговуванню будівлі.

Технічне обслуговування будівель є дорогим процесом як з фінансових аспектів (експлуатаційні витрати, управління нерухомістю, адміністрування, робота з боржниками, юридичні послуги тощо), так і з екологічних аспектів (зміна клімату, викиди парникових газів, питання енергоефективності). Підприємства прагнуть знизити витрати на перетворення будівель у більш ефективну та стійку інфраструктуру.

Технічне обслуговування часто визначають як низку заходів, що здійснюються для догляду за конструкцією будівлі та послугами для забезпечення передбачених функцій та оптимальної роботи протягом життєвого циклу будівлі. Відділ управління будівлі, як правило, відповідає за покращення якості внутрішнього середовища шляхом надання послуг, а також за підвищення продуктивності та задоволеності мешканців.

Для проведення детальної та ретельної розробки програми необхідним є вивчення різних методів організації та реалізації процесів управління та обслуговування будівель, які розглядаються в різних наукових публікаціях. Цей огляд літератури базується на аналізі різних статей, щоб дати розуміння того, як терміни «управління будівлею» та «обслуговування будівель» визначаються в науковій літературі. В цій роботі визначено та оцінено управління будівлею та його технічне обслуговування як інструмент системи управління майном будівлі. У науковій літературі з управління та обслуговування будівель використовуються різні підходи до визначення наведених термінів, пояснення сфери їх застосування та умов реалізації.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

У даному розділі будуть розглянуті інструменти, які необхідні використовувати у даному проекті, а також їх потенційне призначення у контексті проблем даного проекту.

У даному проекті буде розглянута можливість обробки даних з використанням системи моніторингу та управління доступом до систем забезпечення функціонування житлового комплексу, а саме:

1. Обробка та аналіз користувачів, тобто заперечення або виділення доступу перегляду усієї інформації щодо ЖК.
2. Обробка та аналіз паркування.
3. Обробка та аналіз комунальних послуг.
4. Обробка доступних адрес.
5. Обробка та аналіз повідомлень користувачів та адміністрації

Використання автоматичних алгоритмів у сфері ЖК зможе:

– повідомляти користувача коли треба оплатити комунальні послуги
– переглядати статистику оплат по комунальним послугам
– бачити повідомлення от усіх жильців
– переглядати доступні паркувальні місця

Система спрямована на полегшення життя користувачам, а саме мешканцям житлового комплексу та поліпшення уровня жизни жильцов

1.1 Аналіз інструментів розробки

Як згадувалося у вступі (стор.6), основними завданнями при розробці ПП були:

– огляд платформи, на якій буде розроблятися ПП;
– огляд інструментарію для розробки КІ;

– огляд інструментарію для розробки системної частини;

– огляд додаткового інструментарію для розробки ПП;
– розгляд версії платформи
– СУБД для створення БД ПП;
– середовище розробки(IDE), у якому буде проводитися розробка ПП.

1.2 Вибір платформи розробки системної частини

При виборі платформи були розглянуті та проведений порівняльний аналіз віртуальних машин .NET, Java та PHP.

1.2.1 Огляд .NET

Платформа .NET Framework – це керована виконавча для ОС Windows, що надає різноманітні служби для запуску в ній додатків. Вона складається з двох основних компонентів: середовища CLR – механізму, керуючого виконуються додатками, і бібліотеки класів .NET Framework – бібліотеки перевіреного коду, призначеного для повторного використання, який розробники можуть викликати зі своїх додатків. Нижче перераховані служби, які надає .NET Framework виконуваних у ній додатків.

По–перше, у багатьох мовах програмісти повинні передбачати виділення і звільнення пам'яті, а також керувати часом життя об'єктів. Управління пам'яттю у додатках .NET Framework виконує середа CLR.

По–друге, у традиційних мовах програмування базові типи визначаються компілятором, що ускладнює взаємодію між мовами. В .NET Framework базові типи визначаються системою типів .NET Framework, при цьому для всіх мов .NET Framework використовуються одні й ті ж базові типи.

По–третє, розробникам не потрібно писати код для виконання стандартних низькорівневих операцій програмування, так як вони використовують зручну бібліотеку типів і членів, що входить в бібліотеку класів .NET Framework.

По–четверте, платформа .NET Framework включає бібліотеки для конкретних областей розробки додатків, наприклад ASP.NET для вебдодатків, ADO.NET для доступу до даних, Windows Communication Foundation для додатків, орієнтованих на служби, а також Windows Presentation Foundation для класичних додатків Windows.

Крім того, мовні компілятори, орієнтовані на .NET Framework, видають проміжний код, званий мовою CIL (Common Intermediate Language), який в свою чергу компілюється під час виконання середовищем CLR. За допомогою цієї функції підпрограми, написані однією мовою, доступні в інших мовах, тому розробники можуть створювати додатки на бажаних мовами.

Можна сказати, що додатки, розроблені на основі конкретної версії платформи .NET Framework, можуть виконуватися без доробок і на більш пізніх версіях платформи.

Оскільки, платформа .NET Framework допомагає вирішувати конфлікти версій, оскільки на комп'ютері можуть бути встановлені кілька версій середовища CLR. Це означає, що кілька версій додатків можуть співіснувати і що додаток може виконуватися на версії платформи .NET Framework, для якої воно було створено. Паралельне виконання застосовується до груп версій .NET Framework 1.0 / 1.1, 2.0 / 3.0 / 3.5 і 4 / 4.5.x / 4.6.x / 4.7.x / 4.8.

На завершення, під час налаштування відповідно до стандарту .NET розробники створюють бібліотеки класів, які працюють на різних платформах .NET Framework, підтримуваних відповідної версією стандарту. Наприклад, бібліотеки, розроблені відповідно до стандарту .NET 2.0, можуть використовуватися додатками, орієнтованими на платформи .NET Framework 4.6.1, .NET Core 2.0 і UWP 10.0.16299.

1.2.2 Огляд Java

Програмна платформа Java – ряд програмних продуктів і специфікацій компанії Sun Microsystems, раніше незалежної компанії, а нині дочірньої компанії корпорації Oracle, які спільно надають систему для розробки прикладного програмного забезпечення та вбудовування її в будь–який кросплатформенне програмне забезпечення. Java використовується в самих різних комп'ютерних платформах від вбудованих пристроїв і мобільних телефонів в нижньому ціновому сегменті, до корпоративних серверів і суперкомп'ютерів у вищому ціновому сегменті.

Технологія Java–аплетів стала рідко використовуваної в настільних комп'ютерах, проте вона іноді використовується для поліпшення функціональності і підвищення безпеки при перегляді всесвітньої павутини.

Програмний код, написаний на Java, віртуальна машина Java виконує байт–код Java. Однак є компілятори байт–коду для інших мов програмування, таких як Ada, JavaScript, Python, і Ruby. Також є кілька нових мов програмування, розроблених для роботи з віртуальною машиною Java. Це такі мови як Scala, Clojure та Groovy. Синтаксис Java (англ.) В основному запозичений з Сі і C++, але об'єктно–орієнтовані можливості засновані на моделі, використовуваної в мовах програмування Smalltalk і Objective–C. В Java відсутні певні низькорівневі конструкції, такі як покажчики, також Java має дуже просту модель пам'яті, де кожен об'єкт розташований в купі і всі змінні об'єктного типу є посиланнями. Управління пам'яттю здійснюється за допомогою інтегрованої автоматичної збірки сміття, яку виконує JVM.

Компанія Sun Microsystems зробила велику частину своєї реалізації Java доступною відповідно до GNU General Public License (GPL), хоча деякі частини поставляються в скомпільованому вигляді через питання авторського права з кодом, на який має ліцензію, але не право власності, компанія Sun Microsystems.

Програмна платформа Java – це ім'я для пакета програм компанії Sun, які дозволяють розробляти і запускати програми, написані на мові програмування Java. Ця програмна платформа не є специфічною для якого–небудь одного процесора або операційної системи, але механізм виконання (званий віртуальною машиною) і компілятор з набором бібліотек, які реалізовані для різного апаратного забезпечення і різних операційних систем, щоб Java–програми могли працювати скрізь однаково.

По–перше, існує технологія Java Card, яка дозволяє невеликим Java–застосуванням (аплетам) надійно працювати на смарт–картах та інших подібних пристроїв с малим об'ємом пам'яті.

По–друге, є підмножина платформи Java – Java ME, яка включає в себе кілька різних наборів бібліотек (відомих як профілі) для пристроїв з обмеженим обсягом місця для зберігання, невеликим розміром дисплея і батареї. Часто використовується для розробки додатків для мобільних пристроїв, КПК, ресиверів цифрового телебачення і принтерів.

По–третє, існує базова платформа програмування Java – Java SE, яка використовується на настільних ПК, серверах і іншому подібному обладнанні.

В решті–решт, Java EE – це Java SE плюс API, корисне для багаторівневих клієнт–серверних бізнес–додатків.

1.2.2 Огляд PHP

PHP (рекурсивний акронім словосполучення PHP: Hypertext Preprocessor) – це поширена мова програмування загального призначення з відкритим вихідним кодом. PHP спеціально сконструйований для веб-розробок, і його код може впроваджуватися безпосередньо в HTML.

Галузь застосування

1) автоматичне вилучення POST- та GET-параметрів, а також змінних оточення веб-сервера в певні масиви;
2) взаємодія з великою кількістю різних систем управління базами даних через додаткові модулі (MySQL, MySQLi, SQLite, PostgreSQL, Oracle Database (OCI8), Microsoft SQL Server, Sybase, ODBC, mSQL, IBM DB2, Cloudscape та Apache Derby, Informix, Ovrimos SQL, Lotus Notes, DB++, DBM, dBase, DBX, FrontBase, FilePro, Ingres II, SESAM, Firebird та InterBase, Paradox File Access, MaxDB, інтерфейс PDO, Redis);
3) автоматизоване відправлення HTTP-заголовків;
4) робота з HTTP-авторизацією;
5) робота з cookies та сесіями;
6) робота з локальними та віддаленими файлами, сокетами;
7) обробка файлів, що завантажуються на сервер;
8) робота з XForms.

Синтаксис PHP дуже подібний до синтаксису C або Perl. Люди, які знайомі з програмуванням, дуже швидко зможуть почати писати програми на PHP. У цій мові немає строгої типізації даних і немає необхідності в діях виділення/звільнення пам'яті.

Програми, написані на PHP, досить легкочитаємо. Написаний PHP – код легко візуально прочитати та зрозуміти, на відміну від Perl-програм.

Недоліки PHP:

1. PHP є мовою, що інтерпретується, і, внаслідок цього, не може зрівнятися за швидкістю з компілюваним C. Однак при написанні невеликих програм, що, загалом, властиве проектам на PHP, коли весь проект складається з багатьох невеликих сторінок з кодом, вступають у силу накладні витрати на завантаження в пам'ять та виклик CGI-програми, написаної на C.

2. Не така велика база готових модулів, як, наприклад, CPAN у Perl. З цим нічого не можна вдіяти – це справа часу. У PHP 4 розробники передбачили спеціальний репозиторій PEAR, аналогічний CPAN, і я думаю, дуже скоро буде написано достатньо модулів для його наповнення.

За результатами проведеного аналізу було прийнято рішення, розробляти ПП використовуючи платформу Java. Вибір платформи обґрунтовано тим, що у Java є допоміжні модулі для створення ігор. Java гірша за .NET тим, що треба качати додаткове ПЗ, прикладом такого ПЗ може бути JVM, але у сучасний час JVM стоїть майже на кожному комп’ютері.

1.3 Огляд язика програмування для розробки КІ

Гіпертекстова інформаційна система складається з безлічі інформаційних вузлів, безлічі гіпертекстових зв'язків, визначених на цих вузлах та інструментах маніпулювання вузлами та зв'язками. Технологія World Wide Web - це технологія ведення гіпертекстових розподілених систем в Інтернеті, і, отже, вона повинна відповідати загальному визначенню таких систем. Це означає, що це перелічені вище компоненти гіпертекстової системи мають бути й у Web.

Web, як гіпертекстової системи, можна розглядати з двох точок зору. По-перше, як сукупність сторінок, що відображаються, пов'язаних гіпертекстовими переходами (посиланнями - контейнер ANCHOR). По-друге, як безліч елементарних інформаційних об'єктів, що складають сторінки, що відображаються (текст, графіка, мобільний код і т.п.). В останньому випадку безліч гіпертекстових переходів сторінки - це такий самий інформаційний фрагмент, як і вбудована в текст картинка.

При другому підході гіпертекстова мережа визначається на безлічі елементарних інформаційних об'єктів самими HTML-сторінками, які грають роль гіпертекстових зв'язків. Цей підхід більш продуктивний з точки зору побудови сторінок, що відображаються "на льоту" з готових компонентів.

При генерації сторінок у Web виникає дилема, пов'язана з архітектурою "клієнт-сервер". Сторінки можна генерувати як за клієнта, і за сервера. У 1995 році фахівці компанії Netscape створили механізм управління сторінками на стороні клієнта, розробивши мову програмування JavaScript.

Таким чином, JavaScript – це мова керування сценаріями перегляду гіпертекстових сторінок Web на стороні клієнта. Якщо бути точнішим, то JavaScript – це не лише мова програмування на стороні клієнта. Liveware, прабатько JavaScript, є засобом підстановок на стороні сервера Netscape. Проте найбільшу популярність JavaScript забезпечило програмування за клієнта.

Основна ідея JavaScript полягає у можливості зміни значень атрибутів HTML-контейнерів та властивостей середовища відображення у процесі перегляду HTML-сторінки користувачем. При цьому перезавантаження сторінки не відбувається.

Насправді це виявляється у тому, що можна, наприклад, змінити колір фону сторінки чи інтегровану у документ картинку, відкрити нове вікно чи попередження.

Назва "JavaScript" є власністю Netscape. Реалізація мови, здійснена розробниками Microsoft офіційно називається Jscript. Версії JScript сумісні (якщо бути точним, то до кінця) з відповідними версіями JavaScript, тобто. JavaScript є підмножиною мови JScript.

1.5 Вибір версії ПЗ

Java Development Kit (скорочено JDK) – безкоштовно поширюваний компанією Oracle Corporation (раніше Sun Microsystems) комплект розробника додатків на мові Java, що включає в себе компілятор Java (javac), стандартні бібліотеки класів Java, приклади, документацію, різні утиліти і виконавчу систему Java (JRE). До складу JDK не входить інтегроване середовище розробки на Java, тому розробник, що використовує тільки JDK, змушений використовувати зовнішній текстовий редактор і компілювати свої програми, використовуючи утиліти командного рядка.

У своєму ПП використовувалось JDK 11, на момент розробки ПЗ це була остання версія з новітнім генератором документації Javadoc та компілятором.

Вибір JDK являє собою важливу частину розробки ПЗ, обираючи стару версію є ризик, що будуть додаткові помилки при компіляції коду та можливий не запуск.

1.6 Вибір СУБД

1.6.1 Огляд MySQL

MySQL – це реляційна система управління базами даних з відкритим вихідним кодом. В даний час ця СУБД одна з найбільш популярних в вебдодатках – переважна більшість CMS використовує саме MySQL (часто тільки її, без альтернатив), а майже всі веб–фреймворки підтримують MySQL вже на рівні базової конфігурації (без додаткових модулів).

З переваг СУБД MySQL слід відзначити простоту використання, гнучкість, низьку вартість володіння (щодо платних СУБД), а також масштабованість і продуктивність.

MySQL дозволяє зберігати цілочисельні значення зі знаком і беззнакові, довжиною в 1, 2, 3, 4 і 8 байтів, працює із строковими і текстовими даними фіксованої і змінної довжини, дозволяє здійснювати SQL–команди SELECT,

DELETE, INSERT, REPLACE і UPDATE, забезпечує повну підтримку операторів і функцій в SELECT– і WHERE– частинах запитів, працює з GROUP BY і ORDER BY, підтримує групові функції COUNT, AVG, STD, SUM, MAX і MIN, дозволяє використовувати JOIN в запитах, в т.ч. LEFT OUTER JOIN і RIGHT OUTER JOIN, підтримує реплікацію, транзакції, роботу з зовнішніми ключами і каскадні зміни на їх основі, а також забезпечує багато інших функціональні можливості.

Гнучкість СУБД MySQL забезпечується підтримкою великої кількості типів таблиць: користувачі можуть вибрати як таблиці типу MyISAM, що підтримують повнотекстовий пошук, так і таблиці InnoDB, що підтримують транзакції на рівні окремих записів. Є й інші типи таблиць, розроблені спільнотою.

СУБД MySQL з'явилася в 1995. Написана на C і C ++, протестована на безлічі різних компіляторів і працює на різних платформах. С 2010 року розроблення та підтримку MySQL здійснює корпорація Oracle. Продукт поширюється як під GNU GPL, так і під власною комерційною ліцензією. Однак за умовами GPL, якщо яка–небудь програма включає вихідні коди MySQL, то і ця програма теж повинна розповсюджуватися за ліцензією GPL. Для небажаючих відкривати вихідні тексти своїх програм якраз передбачена комерційна ліцензія, яка, на додаток до можливості розробки під «закритою» ліцензією, забезпечує якісну сервісну підтримку. Спільнотою розробників MySQL створені різні відгалуження – Drizzle, OurDelta, Percona Server і MariaDB, всі ці відгалуження вже існували на момент отримання прав на MySQL корпорацією Oracle. Зараз MySQL разом з Форком MariaDB займають почесне перше місце, а слідом за ними йде PostgreSQL. Решта СУБД в веб–проектах використовуються значно рідше.

1.6.2 Огляд SQLite

SQLite – компактна вбудована реляційна база даних. Вихідний код бібліотеки переданий в суспільне надбання. Є чисто реляційної базою даних.

Слово «вбудовується» означає, що SQLite не використовує парадигму клієнт–сервер. Тобто движок SQLite не є окремо працюючим процесом, з яким взаємодіє програма, а надає бібліотеку, з якої програма компонується і движок стає складовою частиною програми. Таким чином, в якості протоколу обміну використовуються виклики функцій (API) бібліотеки SQLite. Такий підхід зменшує накладні витрати, час відгуку і спрощує програму. SQLite зберігає всю базу даних (включаючи визначення, таблиці, індекси і дані) в єдиному стандартному файлі на тому комп'ютері, на якому виповнюється програма. Простота реалізації досягається за рахунок того, що перед початком виконання транзакції записи весь файл, який зберігає базу даних, блокується; ACID – функції досягаються в тому числі за рахунок створення файлу журналу.

Кілька процесів або потоків можуть одночасно без будь–яких проблем читати дані з однієї бази. Запис в базу можна здійснити тільки в тому випадку, якщо ніяких інших запитів в даний момент не обслуговується; в іншому випадку спроба запису закінчується невдачею, і в програму повертається код помилки. Іншим варіантом розвитку подій є автоматичне повторення спроб запису протягом заданого інтервалу часу. Можна, також, ввести таймаут операцій. Тоді підключення, зіткнувшись із зайнятістю БД, чекатиме N секунду до того, як відвалитися з помилкою SQLITE_BUSY.

Також з версії 3.7.0 присутній режим WAL (Write–Ahead Logging), за допомогою якого можна використовувати одну і ту ж базу кількома додатками, як на читання, так і на запис.

Завдяки архітектурі можливо використовувати SQLite як на вбудовуваних системах, так і на виділених машинах з гігабайтними масивами даних.

Формат файлу бази даних є крос–платформних, що дозволяє без проблем використовувати одну і ту ж базу на декількох операційних системах. Також присутня можливість зберігання бази в пам'яті, без її записи на диск. Цей варіант

використовується за умовчанням для консольної утиліти sqlite3, якщо не вказано ім'я файлу.

1.6.3 Огляд Oracle

В наші дні всі складні СУБД є реляційними, але коли в 1970 році співробітник дослідницької лабораторії IBM Едгар Кодд запропонував новий метод організації баз даних, у багатьох фахівців були сумніви в тому, що він взагалі виправдає себе.

Головною конкурентною перевагою СУБД Oracle була висока швидкість обробки величезних масивів інформації, яку відзначили тоді всі експерти. На відміну від System R, для роботи якої був необхідний потужний суперкомп'ютер - мейнфрейм, Oracle 2 справлялася з обробкою інформації на більш скромних машинах. Це й сприяло неймовірно широкому поширенню дітища Еллісона на початку 80-х. А коли з появою третьої версії Oracle стала багатоплатформною, тобто з'явилася можливість встановлювати цю СУБД на різноманітні комп'ютерні системи (їх тоді було не менше 20), популярність її почала зростати ще швидше.

Найпоширенішою СУБД є система від компанії Oracle. Ця СУБД є клієнт-серверною, отже вона призначена для компаній, що мають інформаційну мережу з потужним сервером. Ця СУБД використовує також реляційну модель даних, але містить елементи об'єктно-орієнтованої моделі даних. На 2009 найновішою є версія Oracle 11g Release 2. Розглянемо особливості цієї програми. Масштабованість додатків. Модуль Oracle Real Application Clusters, наступне покоління продукту Oracle Parallel Server, забезпечує прозору масштабованість додатків за рахунок швидкого та ефективного спільного використання кластерного кешу для узгодженого доступу до даних. Oracle Real Application Clusters надає такі можливості:

1) Коробкові програми, які можна масштабувати практично лінійно та абсолютно прозоро.
2) Сумісність з усіма програмами без необхідності їх перебудови.
3) Швидке збільшення кластерів, можливість швидкого додавання вузлів та дисків.

Oracle є однією з найзатребуванішої та найперспективнішої СУБД. Було з'ясовано, що СУБД Oracle мають ряд переваг: висока надійність та безпека, можливість роботи на платформі будь-якої операційної системи.

Проаналізувавши всі СУБД вище, для розробки ПП було обрано Oracle

1.7 Вибір IDE

У своєму проекту я обирав між тьома IDE – Eclipse, IntelliJ Idea, NetBeans.

1.7.1 Розгляд Eclipse

Eclipse – вільне модульне інтегроване середовище розробки програмного забезпечення. Розробляється і підтримується Eclipse Foundation і включає проекти, такі як платформа Eclipse, набір інструментів для програмістів на мові Java, системи контролю версій, конструктори GUI тощо. Написаний в основному на Java, може бути використаний для розробки застосунків на Java і, за допомогою різних плагінів, на інших мовах програмування, включаючи Ada, C, C++, COBOL, Fortran, Perl, PHP, Python, R, Ruby, Scala, Clojure та Scheme. Середовища розробки зокрема включають Eclipse ADT (Ada Development Toolkit) для Ada, Eclipse CDT для C/C++, Eclipse JDT для Java, Eclipse PDT для PHP. Eclipse насамперед повноцінна Java IDE, націлена на групову розробку, має засоби роботи з системами контролю версій (підтримка CVS входить у поставку Eclipse, активно розвиваються кілька варіантів SVN модулів, існує підтримка VSS та інших). З

огляду на безкоштовність, у багатьох організаціях Eclipse – корпоративний стандарт для розробки ПЗ на Java.

Друге призначення Eclipse – служити платформою для нових розширень. Такими стали C/C++ Development Tools (CDT), розроблювані інженерами QNX разом із IBM, засоби для підтримки інших мов різних розробників. Безліч розширень доповнює Eclipse менеджерами для роботи з базами даних, серверами застосувань та інших.

З версії 3.0 Eclipse став не монолітною IDE, яка підтримує розширення, а набором розширень. У основі лежать фреймворки OSGi, і SWT/JFace, на основі яких розроблений наступний шар – платформа і засоби розробки повноцінних клієнтських застосунків RCP (Rich Client Platform). Платформа RCP є базою для розробки різних RCP програм як торент–клієнт Azareus чи File Arranger.

Eclipse написана на Java, тому є платформонезалежним продуктом, крім бібліотеки графічного інтерфейсу SWT, яка розробляється окремо для більшості поширених платформ.

Структура і склад eclipse

Eclipse складається з проектів, кожен з яких займається Комітетом управління проектом (Project Management Committee, PMC) відповідно до Статуту проекту. Проекти поділяються на підпроекти (subprojects). Для оформлення нового проекту (підпроекту) ініціативною групою вноситься пропозиція на проект (project proposal), основним документом якого є декларація проекту (project declaration) та проект отримує статус запропонованого (proposed project). Для запропонованих проектів проводиться обговорення їх цілей, тематичного охоплення, організаційної структури проекту, результатів проекту та формування проектної команди.

Докладніше організаційна структура, проектні ролі та життєвий цикл проектів у рамках Eclipse описані у документі The Eclipse Development Process.

На жовтень 2004 року склад Eclipse за проектами, підпроектами та запропонованими проектами відображено на наступній схемі:

Архітектура платформи eclipse

Графічно архітектура платформи Eclipse може бути виражена схемою:

Кожен інструментальний засіб розробки оформляється у вигляді плагіна (plug-in), що реалізує певний набір функцій, що приєднується до платформи Eclipse за допомогою API і написаного на Java. Як правило, плагін являє собою Java-код, оформлений у вигляді архіву JAR, кілька файлів для читання та набір інших ресурсів, необхідних для роботи плагіна, наприклад бібліотеки, графічні зображення, шаблони і т.д.

Open-source технології, що увагли в eclipse

IDE Eclipse є результатом колективної праці як компаній-вендорів, так і некомерційних об'єднань компаній та приватних осіб. Будучи універсальним середовищем розробки додатків, Eclipse використовує і включає велику кількість open-source технологій та програмних продуктів, які добре зарекомендували себе на ринку у відповідних областях. Нижче ми наводимо перелік цих розробок:

· Ant - Apache Ant Apache Ant; http://ant.apache.org/index.html

· Blowfish Encryption Algorithm – плагін, що реалізує 64-бітний алгоритм шифрування;

· GTK+ - багатоплатформний інструментарій створення GUI, частина проекту GNU Project; http://www.gnu.org

· JSch - Java-реалізація SSH2; http://www.jcraft.com/jsch

· JUnit - інструментарій для тестування та контролю роботи додатків; http://www.junit.org/index.htm

· Lucene – пошуковий "движок", частина проекту Apache Jakarta; http://jakarta.apache.org

· Open Motif for Linux – частина проекту opengroup.org;

· The Java Ssh Applet - плагін, доступний за адресою: http://www.cl.cam.ac.uk/~fapp2/software/java-ssh;

· Tomcat – веб-контейнер, частина проекту Apache Jakarta;

· XML Parser for Java - XML-парсер для Java (http://www.alphaworks.ibm.com/tech/xml4j) заснований на Apache Xerces.

В цілому, платформа Eclipse надає базис, що складається із загальних будівельних блоків та API для робочих областей та робочого середовища, а також різні точки розширення, через які може інтегруватися нова функціональність. Через ці точки розширення утиліти, реалізовані як окремих плагінів, можуть розширювати платформу Eclipse. Користувач бачить інтегроване середовище розробки (IDE), спеціалізоване набором доступних плагінів. Проте замість того, щоб бути закінченням історії, це насправді лише її початок. Утиліти можуть також задавати нові точки розширення та API, служачи цим будівельними блоками і точками інтеграції з іншими утилітами.

1.7.2 Розгляд IntelliJ Idea

IntelliJ IDEA – комерційне інтегроване середовище розробки для різних мов програмування (Java, Python, Scala, PHP та ін.) від компанії JetBrains. Система поставляється у вигляді урізаної по функціональності безкоштовної версії «Community Edition» і повнофункціональної комерційної версії «Ultimate Edition», для якої активні розробники відкритих проектів мають можливість отримати безкоштовну ліцензію. Сирцеві тексти Community–версії поширюються рамках ліцензії Apache 2.0. Бінарні збірки підготовлені для Linux, Mac OS X і Windows.

Community версія середовища IntelliJ IDEA підтримує інструменти (у вигляді плагінів) для проведення тестування TestNG і JUnit, системи контролю версій CVS, Subversion, Mercurial і Git, засоби складання Maven, Ant, Gradle, мови

програмування Java, Scala, Clojure, Groovy і Dart. Підтримується розробка застосунків для мобільної платформи Android. До складу входить модуль візуального проектування GUI–інтерфейсу Swing UI Designer, XML–редактор, редактор регулярних виразів, система перевірки коректності коду, система контролю за виконанням завдань і доповнення для імпорту та експорту проектів з Eclipse. Доступні засоби інтеграції з системами відстеження помилок JIRA, Trac, Redmine, Pivotal Tracker, GitHub, YouTrack, Lighthouse.

Комерційна версія «Ultimate Edition» відрізняється наявністю підтримки додаткових мов програмування (наприклад, PHP, Ruby, Python, JavaScript, CoffeeScript, HTML, CSS, SQL), підтримкою технологій Java EE, UML–діаграм, підрахунок покриття коду, можливістю роботи з фреймворками (Rails, Grails, Google Web Toolkit, Spring, Play Framework і Hibernate), засобами інтеграції з Perforce, Microsoft Team Foundation Server і Rational ClearCase.

IntelliJ IDEA - це насамперед середовище розробки для Java. З цією мовою вона дружить найбільше, добре його розуміє і допомагає в написанні розробнику. Але це не означає, що все закінчується на Джаві та власній мові Kotlin. Не менш важлива підтримка їхньої коробки таких передових технологій, як Groovy, Scala та інших. Вони поєднують у собі можливості динозаврів, як Java разом із функціоналом Ruby, Smalltalk та інших. Не забувайте в компанії та засобах веб-пошуку для окремих середовищ, заснованих на IntelliJ IDEA.

Автодоповнення - не винятковий функціонал. Такі самі можливості ми можемо спостерігати практично в кожної безкоштовної програми. Причому реалізовані вони на досить високому рівні. Але інтелект, як завжди, відрізняє IntelliJ IDEA. Там, де звичайна програма запропонує кілька, IntelliJ IDEA видає лише один, але вірний. Це заощаджує лише секунду, зате в сукупності робить розробку набагато більш оптимізованою.

Однією із сильних сторін у JetBrains вважають підтримку широкого кола технологій. Наприклад, у всьому світі технології Flash вважаються застарілими.

Але незважаючи на це, в деяких продуктах Flash реалізується досить широко і скидати його з рахунків рано. Так вирішили і розробники IntelliJ IDEA. Вони не виключають технологію, поки всі довкола не перестануть нею користуватися. У результаті більшість Flash-розробників мігрує на IntelliJ IDEA. Компанії не такі важливі світові тенденції, у той час як лояльність користувачів зростає.

Не менш важливим інструментом є графічний редактор для створення інтерфейсу. Це значно зручніше, ніж писати шаблонний код. Розробник може лише перетягувати мишкою елементи на робоче поле, тоді як код генерується самостійно. Серед досвідчених програмістів такий підхід можна вважати дилетантським. Але правда в тому, що таким чином економиться багато часу, а якість коду не стає гіршою. Через це візуальні редактори використовують не лише новачки, а й ті, хто просто хоче оптимізувати свою роботу.

Можливості IntelliJ IDEA:

1) розумне автодоповнення, інструменти для аналізу якості коду, зручна навігація, розширені рефакторинги та форматування для Java, Groovy, Scala, Clojure та Erlang;
2) професійний набір інструментів для розробки Android-додатків;
3) підтримка JavaFX 2.0 та інтеграція з SceneBuilder;
4) дизайнер інтерфейсу для Swing;

5) інтеграція з автоматизованими інструментами збирання та управління проектом, включаючи Maven, Gradle, Ant та інші.
6) інструменти для тестування програм з підтримкою JUnit, TestNG, Spock, ScalaTest та spec2;
7) інтеграція із системами управління версіями, включаючи Git, Subversion, Mercurial та CSV.

1.7.3 Розгляд NetBeans

NetBeans IDE — вільне інтегроване середовище розробки додатків (IDE) мовами програмування Java, JavaFX, Python, PHP, JavaScript, C++, Ада та інших.

Для розробки програм у середовищі NetBeans та для успішної інсталяції та роботи самого середовища NetBeans має бути попередньо встановлений Sun JDK або J2EE SDK відповідної версії. Середовище розробки NetBeans за умовчанням підтримувало розробку для платформ J2SE та J2EE. Починаючи з версії 6.0, Netbeans підтримує розробку для мобільних платформ J2ME, C++ (тільки g++) і PHP без встановлення додаткових компонентів.

Проект NetBeans IDE підтримується та спонсорується компанією Oracle, проте розробка NetBeans ведеться незалежно спільнотою розробників-ентузіастів (NetBeans Community) та компанією NetBeans Org.

За якістю та можливостями останні версії NetBeans IDE не поступаються кращим комерційним (платним) інтегрованим середовищам розробки для мови Java, таким, як IntelliJ IDEA, підтримуючи рефакторинг, профільування, виділення синтаксичних конструкцій кольором, автодоповнення набираються конструкцій на льоту, безліч визначених шаблонів .

У версії NetBeans IDE 6.1 декларується підтримка UML, SOA, мови програмування Ruby (включно з підтримкою Ruby on Rails), а також засоби для створення програм на J2ME для мобільних телефонів. У версії 6.5 додано підтримку мови PHP. Також для тестування викладено модуль підтримки Python.

NetBeans IDE підтримує плагіни, дозволяючи розробникам розширювати можливості середовища. Одним із найпопулярніших плагінів є потужний дизайнер звітів iReport (заснований на бібліотеці JasperReports).

На ідеях, технологіях та в значній частині на вихідному коді NetBeans IDE базуються пропоновані фірмою Sun комерційні інтегровані середовища розробки

для Java – Sun Java Studio Creator, Sun Java Studio Enterprise та Sun Studio (для розробки на C, C++ або Фортран). Порівняно недавно Sun стала пропонувати ці середовища розробки безкоштовно для розробників, що зареєструвалися в Sun Developer Network (SDN), сама ж реєстрація на сайті безкоштовна і не вимагає жодних попередніх умов, крім згоди з ліцензією CDDL.

NetBeans IDE доступна у вигляді готових дистрибутивів (прекомпільованих бінарних файлів) для платформ Microsoft Windows, Linux, FreeBSD, Mac OS X, OpenSolaris та Solaris (як для SPARC, так і для x86 – Intel та AMD). Для решти платформ доступна можливість скомпілювати NetBeans самостійно з вихідних текстів.

У релізі NetBeans IDE 6.7 була додана інтеграція з Project Kenai, підтримка мови Groovy та веб-фреймворку Grails. У версії 6.8 - підтримка PHP-фреймворку Symfony, а в 6.9 - Zend Framework.

Поточні версії:

NetBeans IDE 6.0, створена на основі попередньої версії 5.5.1, надає гнучку підтримку створення модулів для IDE та інтернет-додатків, заснованих на платформі NetBeans, новий дизайнер інтерфейсів користувача (відомий під назвою «Проект Matisse»), нову та перероблену підтримку системи управління версіями CVS, підтримку Weblogic 9 та JBoss 4, та безліч покращень у редакторі. NetBeans 6.0 постачається у складі дистрибутивів Ubuntu 8.04 та Debian.

NetBeans IDE 6.5, випущена у листопаді 2008 року, розширює можливості Java EE (включаючи підтримку Java Persistence, EJB 3 та JAX-WS). Додатково NetBeans Enterprise Pack підтримує розробку програм Java EE 5 Enterprise, включаючи візуальні засоби SOA, засоби для роботи з XML schema, роботу з веб-сервісами (для BPEL), та моделювання мовою UML. Складання NetBeans IDE Bundle for C/C++ підтримує проекти мовами C/C++.

Поточна версія середовища - NetBeans IDE 7.0

Після аналізу для себе я обрав IntelliJ Idea, яку компанія jetBrains надає безкоштовно для студентів на рік, та в Ultimate Edition є спеціальний додаток для UML – діаграм, також ця IDE являє собою дуже зручну систему підказок.

2 ВИБІР ДОДАТКОВОГО ІНСТРУМЕНТАРІЮ ДЛЯ РОЗРОБКИ ПП

2.1 Огляд інструментарію для розробки системної частини

1. Spring framework

Spring – рівневий фреймворк для розробки Java/J2EE додатків, оснований на коді, опублікованому в “Expert One-on-One J2EE Design and Development” Родом Джонсоном.

Spring включає:

- Найповніший легкий контейнер, який забезпечує централізовану, автоматизовану конфігурацію та з’єднання об’єктів. Контейнер може з наборів вільно з’єднаних компонентів складної системи (POJOs) скласти стійку та прозору форму. Контейнер забезпечує швидкість, забезпечує тестованість та масштабованість додатку шляхом можливості розробки та тестування компонентів ізольовано з подальшою масштабованістю для використання у будь-якому середовищі (J2SE чи J2EE).
- Спільний абстрактний рівень для управління трансакціями, який дозволяє змінювати менеджерів трансакцій та розподіляти трансакції, не маючи справи з низькорівневими проблемами.
- Включено стратегії JTA та єдиного джерела даних JDBC DataSource. На противагу простим JTA чи EJB CMT, підтримка трансакцій в Spring не прив’язана до середовища J2EE.
- Абстрактний рівень JDBC, який пропонує чітку ієрархію виключень, спрощує обробку помилок, і значно зменшує об’єм необхідного коду. Тепер вже не потрібно писати ще один блок для того, щоб знову використати JDBC. JDBC-орієнтовані виключення реалізовують Spring-ову ієрархію виключень DAO.

- Інтеграція з Toplink, Hibernate, JDO, та iBATIS SQL Maps: в якості сховища ресурсів, підтримки реалізації DAO та стратегій трансакцій. Першокласна підтримка Hibernate з безліччю зручних функцій IoC, що стосуються багатьох типових інтеграційних функцій Hibernate.
- Функціональність AOP, повністю інтегрована в менеджмент конфігурацій Spring-а. Можна AOP-enable будь-який об'єкт керований Spring-ом, додавши такі аспекти як декларативне управління трансакціями. З Spring можна мати декларативне управління трансакціями навіть без EJB.
- Гнучкий MVC фреймворк веб додатків, побудований на основній функціональності Spring. Цей фреймворк високо конфігурований через інтерфейси і забезпечує численні технології як JSP, Velocity, Tiles, iText, та POI. Зауважте, що проміжний рівень Spring може бути легко з'єднаний з веб рівнем на будь-якому іншому MVC веб фреймворку, наприклад, Struts, WebWork, чи Tapestry.

Використовувати всю функціональність Spring можна на будь-якому J2EE сервері, і більшість – в некерованих середовищах. Spring робить фокус на бізнес об'єктах та об'єктах доступу до даних повторно використання, які не прив'язані до специфічних сервісів J2EE. Такі об'єкти можуть бути багаторазово використані в середовищах J2EE (веб або EJB), автономних додатках, тестових середовищах, і т.п. без будь-яких перешкод.

Головна мета Spring – полегшити використання J2EE та пропагувати хорошу практику програмування. Він реалізує це, забезпечуючи модель програмування на основі POJO, яка застосовна у значній кількості середовищ.

Spring не перевинаходить колесо. Тому в Spring не знайти логуючих пакетів, пулів з'єднань, координатора розподілених трансакцій. Всі ці речі надаються проектами open source (такими як Commons Logging або Commons DBCP) або завдяки серверу застосувань. З тієї самої причини, O/R mapping рівень теж не забезпечується. Для цього хорошим вирішенням буде TopLink, Hibernate та JDO.

Spring не має на меті полегшити використання наявних технологій. Наприклад, хоча координація низькорівневих трансакцій не охоплюється спектром, абстрактний рівень над JTA чи будь-якою іншою транзакційною стратегією забезпечується.

Spring відкрито не змагається з іншими open source до того часу, поки немає можливості запропонувати щось нове. В деяких сферах, таких як IoC контейнер та AOP фреймворк, Spring має пряму конкуренцію, але Spring був піонером в цих сферах.

Spring має переваги від внутрішньої consistency. Всі розробники співають гімн з одного листка, тобто фундаментальні ідеї черпаються з *Expert One-on-One J2EE Design and Developmen.* Центральні концепції, як наприклад IoC, використовуються в багатьох місцях.

Spring – мобільна між серверами застосувань. Звичайно, мобільність – це завжди певні труднощі і небезпека, але творці Spring намагаються уникати будь-що платформенно-залежне чи нестандартне з погляду розробника, та надають підтримку на WebLogic, Tomcat, Resin, JBoss, Jetty, Geronimo, WebSphere та інших серверах застосування. Підхід з POJO надає можливість скористатись перевагами рис, специфічних для певного середовища, не жертвуючи при цьому мобільністю.

Spring Boot - це корисний проект, метою якого є спрощення створення програм на основі Spring. Він дозволяє найпростішим способом створити web-додаток, вимагаючи від розробників мінімум зусиль з його настроювання та написання коду

2. Особливості Spring Boot

Spring Boot має великий функціонал, але його найбільш значущими особливостями є: управління залежностями, автоматична конфігурація та вбудовані контейнери сервлетів

Простота управління залежностями

Щоб прискорити процес управління залежностями, Spring Boot неявно пакує необхідні сторонні залежності для кожного типу програми на основі Spring і надає їх розробнику за допомогою так званих starter-пакетів (spring-boot-starter-web, spring-boot-starter-data-jpa і т.д. .Д.)

Starter-пакети є набором зручних дескрипторів залежностей, які можна включити у свою програму. Це дозволить отримати універсальне рішення для всіх, пов'язаних зі Spring технологій, позбавляючи програміста від зайвого пошуку прикладів коду та завантаження з них необхідних дескрипторів залежностей (приклад таких дескрипторів та стартових пакетів буде показано нижче)

Для створитення Spring web-додатка, треба додати залежність spring-boot-starter-web, яка підтягне в проект усі бібліотеки, необхідні для розробки Spring MVC-додатків, таких як spring-webmvc, jackson-json, validation-api та Tomcat

Іншими словами, Spring Boot збирає всі загальні залежності та визначає їх в одному місці, що дозволяє розробникам просто використовувати їх, замість того, щоб винаходити колесо щоразу, коли вони створюють нову програму

Отже, при використанні Spring Boot файл pom.xml містить набагато менше рядків, ніж при використанні його в Spring-додатках

Автоматична конфігурація

Другою чудовою можливістю Spring Boot є автоматична конфігурація програми

Після вибору відповідного starter-пакета, Spring Boot спробує автоматично налаштувати Spring-додаток на основі доданих вами jar-залежностей

2. Apache Maven

Apache Maven - фреймворк для автоматизації складання проектів, специфікованих на XML-мові POM (англ. Project Object Model).

Maven, на відміну від іншого збирача проектів Apache Ant, забезпечує декларативне, а не імперативне складання проекту. Тобто у файлов проекту

pom.xml міститься його декларативний опис, а не окремі команди. Всі завдання обробки файлів Maven виконує через плагіни.

Одна з головних рис фреймворку — декларативний опис проекту. Це означає, що розробнику не потрібно приділяти увагу кожному аспекту збирання - всі необхідні параметри налаштовані за замовчуванням. Зміни потрібно вносити лише тому обсязі, у якому програміст хоче відхилитися від стандартних налаштувань.

Ще одна перевага проекту - гнучке управління залежностями. Maven вміє підвантажувати до свого локального репозиторію сторонні бібліотеки, вибирати необхідну версію пакета, обробляти транзитивні залежності.

Розробники також наголошують на незалежності фреймворку від ОС. При роботі з командного рядка параметри залежать від платформи, але Maven дозволяє не зважати на цей аспект.

При необхідності систему збирання можна налаштувати під власні потреби, використовуючи готові плагіни та архетипи. А якщо нічого придатного не знайшлося — можна написати свої.

3. Log4j

Log4j - бібліотека журналування (логування) Java-програм, частина загального проекту "Apache Logging Project".

Log4j спочатку розвивався в рамках парасолькового Apache Jakarta Project, відповідального за всі Java-проекти Apache, але згодом виділився в окремий, дуже популярний проект журналування.

Використовується часто при написанні програм Java, для ведення логів.

На рис. 1.1 зображена таблиця на якій можна побачити вбудовані рівні журналу та повідомлення Log4j в порядку зростання серйозності. У лівому стовпці перераховані позначення рівня журналу Log4j, а правому стовпці наведено короткий опис кожного рівня журналу.

Уровень	Описание
OFF	Максимально возможный уровень, предназначен для выключения логирования.
FATAL	Серьезные ошибки, которые вызывают преждевременное прекращение. Ожидается, что они будут немедленно видны на консоли состояния.
ERROR	Ошибки во время выполнения или неожиданные условия. Ожидайте, что они будут немедленно видны на консоли состояния.
WARN	Использование устаревших API, неправильное использование API, «почти» ошибки, ситуации времени выполнения которые нежелательны или неожиданны, но не обязательно «неправильные». Ожидается, что они будут немедленно видны на консоли состояния.
INFO	Уведомления во время выполнения (запуск / выключение). Ожидается, что они будут немедленно видны на консоли.
DEBUG	Подробная информация о потоке через систему. Ожидается, что они будут записаны только в журналы. В общем, большинство строк, написанных вашим приложением, должны быть записаны как DEBUG.
TRACE	Наиболее подробная информация. Ожидается, что они будут записаны только в журналы. Начиная с версии 1.2.12.

Рисунок 1.1 – таблиця рівнів журналу

2.2Огляд інструментарію для розробки КІ

1. Angular Js

Angular представляє фреймворк від компанії Google для створення клієнтських програм. Насамперед він націлений розробку SPA-решений (Single Page Application), тобто односторінкових додатків. У цьому плані Angular є спадкоємцем іншого фреймворку AngularJS. У той же час Angular – це не нова версія AngularJS, а принципово новий фреймворк.

Angular надає таку функціональність як двостороннє зв'язування, що дозволяє динамічно змінювати дані в одному місці інтерфейсу при зміні даних моделі в іншому, шаблони, маршрутизація і так далі.

Однією з ключових особливостей Angular є те, що він використовує мову програмування TypeScript.

HTML чудово підходить для оголошення статичних документів, але коли ми намагаємося використовувати його для оголошення динамічних представлень у веб-додатках, він не вдається. AngularJS дозволяє розширити словниковий запас HTML для вашої програми. Отримане середовище надзвичайно виразне, читабельне та швидко розвивається.

Інші фреймворки справляються з недоліками HTML, абстрагуючи HTML, CSS та/або JavaScript, або забезпечуючи необхідний спосіб маніпулювання DOM. Жоден із них не вирішує основну проблему, що HTML не був розроблений для динамічних переглядів.

AngularJS — це набір інструментів для створення фреймворка, який найбільше підходить для розробки ваших додатків. Він повністю розширюється і добре працює з іншими бібліотеками. Кожну функцію можна змінити або замінити відповідно до вашого унікального робочого процесу розробки та потреб функцій.

Функції AngularJs

Пайпи (pipes)

1. fuel-ui - OrderBy і Range, портовані з AngularJS 1.x в Angular
2. ngx-filter-pipe Пайп (pipe) для фільтрації масивів
3. ngx-pipes набір корисних пайпів для Angular
4. ngx-order-pipe OrderBy - сортування колекцій
5. angular2-camelcase Пайп для перетворення рядків у camelCase
6. angular-pipes — Використовуємо круті пайпи
7. ngx-pipes — Пайпи без жодної залежності
8. ng-pipes — Набір корисних пайпів
9. angular-linky — Linky пайп

Стукрутури даних

1. angular-localstorage — Декоратор для автоматичного збереження та відновлення полів класів з LocalStorage
2. ng-webstorage - LocalStorage та SessionStorage менеджер
3. ng-storage localStorage та sessionStorage обгортки
4. angular-safeguard — Обертання над cookies/sessionStorage/localStorage
5. @ngx-cache/core — Розумне кешування в Angular

6. angular-cookie Бібліотека імплемінтуюча з AngularJS 1.x $cookies-сервіс в Angular
7. ng-http-cache - Кешування http-запитів

Роутінг

1. ng-breadcrumb - генератор ієрархії маршрутизації на основі вкладеного роутингу
2. ng-page-transition — Простий компонент з анімованими переходами під час маршрутизації
3. @ngx-i18n-router/core — Інструмент інтернаціоналізації роутингу

Валідація

1. ng-validators — Простий валідатор реактивних форм на основі validator.js
2. ng-validation -Проста валідація на Angular
3. ngx-dynamic-form-builder — Крутий валідатор реактивних форм на основі class-validator та class-transformer

Логування

1. angular-logger — Реалізація log4j для angular
2. @nsalaun/ng-logger — Простий сервіс логування

i18n

2. @ngx-translate/core — Зручна бібліотека для роботи з файлами локалізацій (i18n)
3. @angular-ua/ngx-i18n-combine — Об'єднання файлів i18n з компонентів та загальних файлів для ваших локалізацій (json, ts, js, jsx, tsx)
4. angular-l10n — Бібліотечка для перекладу повідомлень, дат та цифр
5. @ngx-universal/translate-loader — Лоадер, який забезпечує переклади на стороні браузер або сервера

Мінуси:

1. Вище поріг входження через Observable (RxJS) та Dependency Injeciton;
2. Щоб все працювало добре і швидко потрібно витрачати час на додаткові оптимізації (він не супер швидкий, за замовчуванням, але швидше за AngularJS у багато разів і з кожною новою версією стає все швидше);
3. Angular-Univesal має багато підводних каменів;
4. Динамічне створення компонентів виявляється нетривіальним завданням.

Form Builder - для розробки справді складних форм, вам слід знати реактивні форми, точніше важливо забути про декларативні форми. Ось один із добрих прикладів (реактивна форма + валідація);

Change Detection - оскільки Angular за замовчуванням використовує двостороннє зв'язування моделі даних, то при роботі з великим обсягом таких даних ваші програми працюватимуть повільніше, тому в деяких випадках варто подбати про правильну стратегію виявлення змін. Можна переглянути різні OpenSource проекти: PrimeNG, Angular Material, Clarity UI, Angular Bootstrap та інші, всі вони використовують ChangeDetection.OnPush.

Templating - синтаксис шаблонів з точки зору абстракції змінився не надто сильно в порівнянні з AngularJS, тобто ми також можемо написати умови, цикли, зв'язати модель даних та інше. Все, що вам слід добре зрозуміти і розібратися в Angular шаблонах - що таке структурні та декларативні директиви, а також що таке Input-параметри та Output-події.

Routing — це, напевно, одне з основних явищ у створенні веб-додатків. Тут просто важливо розуміти, що маршрутизація так само, як і компоненти, має свій життєвий цикл. Ще варто зазначити: якщо на будь-якому з маршрутів вішаєте модуль, а не компонент, який відповідає за відображення сторінки цього маршруту, то модуль буде підвантажуватися на сторінці на вимогу.

Annotations - до речі, багато новачків не знають, але варто відзначити. Декоратори, які використовуються вдосталь при написанні додатків на Angular, не є якоюсь жорсткою магією TypeScript. Декоратори є специфікацією EcmaScript і коли браузери почнуть підтримувати їх, вони будуть виконуватися в browser runtime. Насправді декоратори дуже корисні і забезпечують досить високою читальністю ваш код. Один із прикладів – це валідація моделей даних із використанням декораторів або десеріалізація/серіалізація даних.

Observables - насправді, тут варто відзначити тільки те, що незабаром Observables будуть специфікацією EcmaScript і все це буде підтримуватися в браузерах. З погляду теорії, якщо розкрити поняття Observer (спостерігач) — це шаблон проектування. Також відомий як "підлеглі" (Dependents). Створює механізм класу, який дозволяє отримувати екземпляру об'єкта цього класу оповіщення від інших об'єктів про зміну їх стану, тим самим спостерігаючи за ними.

Shadow DOM - це засіб для створення окремого DOM-дерева всередині елемента, який не видно зовні без застосування спеціальних методів, є специфікацією W3C. Грубо кажучи, це зручний спосіб створення ізольованих та перевикористовуваних веб-компонентів. Чисто технічно, якби браузери вже сьогодні підтримували багато концепцій, які використовує Angular, нам не потрібні були б транспайлери та інші системи складання, все, що ми писали на Angular, працювало б вже нативно.

2. TypeScript

TypeScript — мова програмування, представлена Microsoft у 2012 році і позиціонується як засіб розробки веб-додатків, що розширює можливості JavaScript.

TypeScript є сумісним з JavaScript і компілюється в останній. Фактично, після компіляції програму на TypeScript можна виконувати у будь-якому сучасному браузері або використовувати разом із серверною платформою Node.js. Код

експериментального компілятора, що транслює TypeScript JavaScript, поширюється під ліцензією Apache. Його технологія ведеться у громадському репозиторії через обслуговування GitHub.

TypeScript відрізняється від JavaScript можливістю явного статичного призначення типів, підтримкою використання повноцінних класів (як у традиційних об'єктно-орієнтованих мовах), а також підтримкою підключення модулів, що покликане підвищити швидкість розробки, полегшити читання, рефакторинг та повторне використання коду, допомогти здійснювати пошук помилок на етапі розробки та компіляції, і, можливо, прискорити виконання програм.

Планується, що з повної зворотної сумісності адаптація існуючих додатків новий мову програмування може відбуватися поетапно, шляхом поступового визначення типів.

На момент релізу представлені файли для сприйняття розширеного синтаксису TypeScript для Vim та Emacs, а також плагін для Microsoft Visual Studio.

Одночасно з виходом специфікації розробники підготували файли з деклараціями статичних типів деяких популярних JavaScript-бібліотек, серед яких jQuery.

TypeScript виник через ймовірні недоліки JavaScript у великомасштабних додатках як у компанії Microsoft, так і в інших користувачів JavaScript. Проблеми з розробкою складних програм на JavaScript призвели до полегшення розробки компонентів мови.

Розробники TypeScript шукали рішення, яке не порушуватиме сумісність зі стандартом та його кросплатформовою підтримкою. Знаючи, що тільки стандарт ECMAScript пропонує підтримку в майбутньому для програмування на базі класів (Class-based programming), TypeScript був заснований на цьому припущенні. Це призвело до створення компілятора JavaScript з набором синтаксичних мовних розширень, збільшеним на основі речення, що трансформує розширення JavaScript. У цьому сенсі TypeScript є уявленням того, що очікувати від

ECMAScript 6. Унікальний аспект не в реченні, а в додаванні в TypeScript статичної типізації, що дозволяє статично аналізувати мову, полегшуючи оснащення та IDE підтримку.

Сумісність із JavaScript

TypeScript є сумісним з JavaScript. Таким чином, будь-який код JavaScript також правильний для TypeScript. У TypeScript можна використовувати існуючий код JS і підключати популярні бібліотеки JavaScript. Оголошення типів для цих бібліотек часто поставляється разом з ними, або може бути написано вручну.

За замовчуванням TypeScript компілюється у сумісний з ES3 JavaScript як стандарт. За допомогою параметра –-target або його скороченої версії –t можна задати версію стандарту JavaScript, до якої компілюватиметься код TypeScript. Цей параметр може приймати такі значення: ES3 (за замовчуванням), ES5, ES6/ES2015, ES7/ES2016, ES2017, ES2018, ES2019, ES2021 або ESNext:

Оголошення типів

TypeScript забезпечує оголошення типів статичної перевірки їх узгодження. Це не є обов'язковим і може бути проігноровано використовувати звичайну динамічну типізацію JavaScript.

Існує кілька анотацій для примітивних типів: number, boolean та string. Слабо або динамічно введені структури мають тип any.

Визначення типів можуть бути експортовані в окремий файл оголошень, щоб зробити інформацію про типи доступною для сценаріїв TypeScript з використанням різних типів, що вже скомпіловані в JavaScript. Визначення можуть бути заявлені до існуючої бібліотеки JavaScript, як це було зроблено для Node.js та JQuery.

Компілятор TypeScript намагається вивести типи, коли їх не вказано явно. Наприклад, метод add у наведеному вище коді буде виводити як повернення в number, навіть якби не було передбачено жодного повернення типу у визначенні. Це засноване на статичних типах left і right numbers і знання компілятора про те,

що результат складання двох numbers завжди number. Тим не менш, прописування типу, що повертається, дозволяє компілятору перевірити правильність.

Якщо тип не може бути виведений через відсутність оголошень, за промовчанням буде динамічний тип any. Значення типу any підтримує ті самі операції, що і значення JavaScript і мінімальна статична перевірка типів виконується для операції на any.

3 ПРОЕКТНА ЧАСТИНА

3.1 Постановка завдань

За результатами огляду і аналізу предметної області, ринку подібних проектів і вимог до ПП, було виділено ряд завдань, які потребують вирішення, щоб успішно виконати проект дипломної роботи:

– спроектувати та створити користувальницький інтерфейс додатку
– спроектувати та створити системної частину додатку
– спроектувати та створити навчальну частину додатку

3.2 UML діаграма проекту

UML (англ. Unified Modeling Language – уніфікована мова моделювання) – мова графічного опису для об'єктного моделювання в області розробки програмного забезпечення, для моделювання бізнес–процесів, системного проектування та відображення організаційних структур.

UML є мовою широкого профілю, це – відкритий стандарт, який використовує графічні позначення для створення абстрактної моделі системи, званої UML–моделлю. UML був створений для визначення, візуалізації, проектування та документування, в основному, програмних систем. UML не є мовою програмування, але на підставі UML–моделей можлива генерація коду.

На рис. 2.1 зображено UML діаграму класів проекту, на якій можна побачити усі зв’язки між класами.

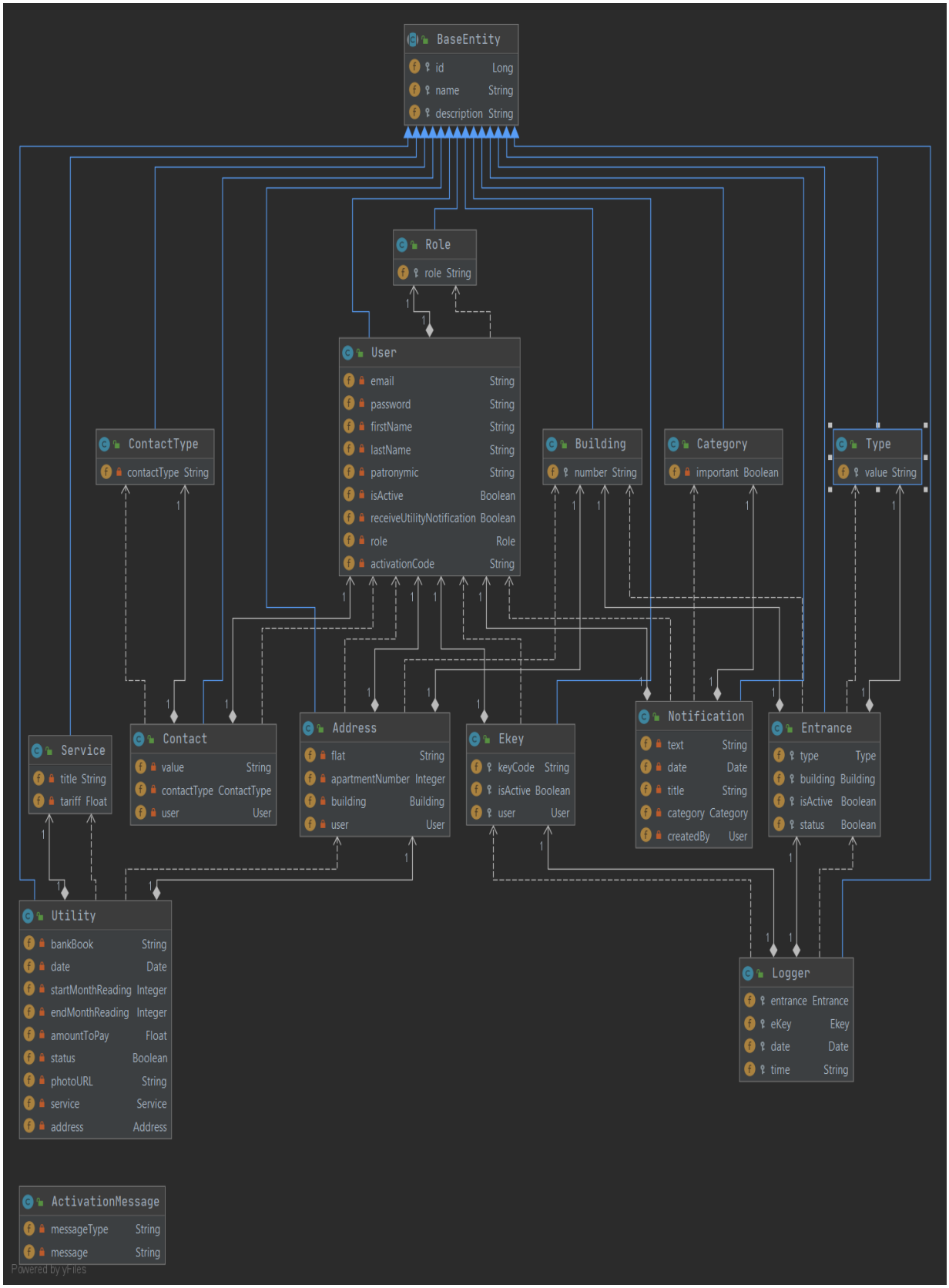

Рисунок 2.1 – Діаграма класів додатку

3.3Діаграма прецедентів

Діаграма варіантів використання (англ. use case diagram) в UML - діаграма, що відображає відносини між акторами і прецедентами і є складовою моделі прецедентів, що дозволяє описати систему на концептуальному рівні.

Прецедент - можливість моделюється системи (частина її функціональності), завдяки якій користувач може отримати конкретний, вимірний і потрібний йому результат. Прецедент відповідає окремому сервісу системи, визначає один із варіантів її використання та визначає типовий спосіб взаємодії користувача з системою. Варіанти використання зазвичай застосовуються специфікації зовнішніх вимог до системи.

Основне призначення діаграми - опис функціональності та поведінки, що дозволяє замовнику, кінцевому користувачеві та розробнику спільно обговорювати проектовану або існуючу систему.

При моделюванні системи за допомогою діаграми прецедентів системний аналітик прагне:

1. чітко відокремити систему від її оточення;
2. визначити дійових осіб (акторів), їхню взаємодію із системою та очікувану функціональність системи;
3. визначити у глосарії предметної області поняття, що належать до детального опису функціональності системи (тобто прецедентів).

Робота над діаграмою може розпочатися з текстового опису, отриманого під час роботи із замовником. У цьому нефункціональні вимоги (наприклад, конкретну мову чи система програмування) під час упорядкування моделі прецедентів опускаються (їм складається інший документ).

На рисунку 2.2 зображено діаграму прецедентів

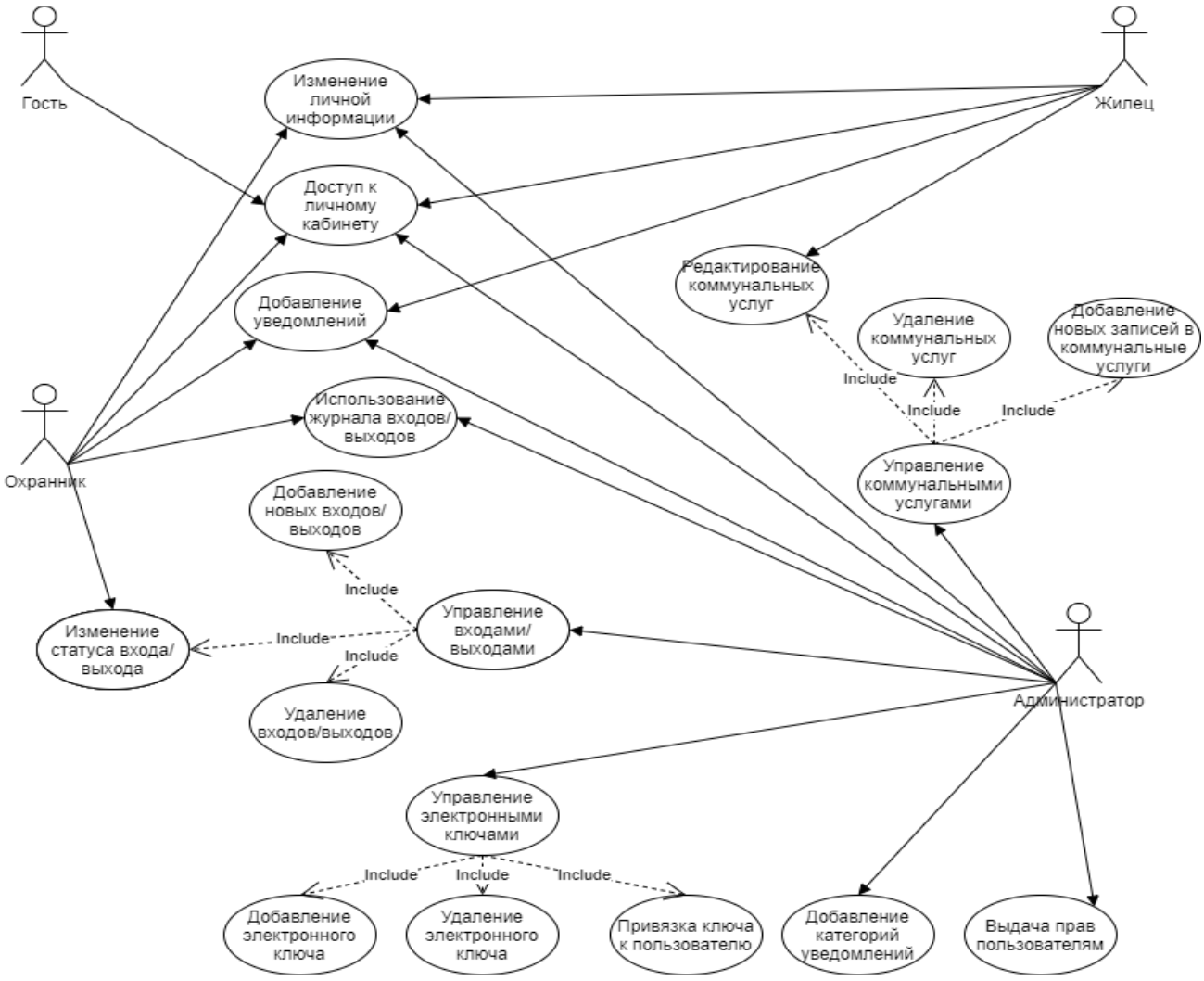

Рисунок 2.2 – Діаграма прецедентів

На діаграмі вище можна побачити всі можливі дії користувачів системи, на ній можна побачити чотири ролі: адміністратор, гість, охоронець, мешканець.

Можливості адміністратора:

1. Управління комунальними послугами:
    a. Редагування комунальних послуг
    b. Видалення комунальних послуг
    c. Додавання комунальних послуг
2. Редагування особистої інформації
3. Доступ до особистого кабінету

4. Додавання повідомлень для мешканців
5. Використання журналу входів та виходів
6. Управління входами та виходами
    a. Додавання нових входів та виходів
    b. Видалення входів та виходів
    c. Редагування статусу входів та виходів
7. Управління електронними ключами
    a. Додавання електронного ключа
    b. Видалення електронного ключа
    c. Прив’язка електронного ключа к користовачеві
8. Додавання категорій повідомлень
9. Видача прав користувачам

Можливості охоронця:

1. Редагування особистої інформації
2. Доступ до особистого кабінету
3. Додавання повідомлень для мешканців
4. Використання журналу входів та виходів
5. Редагування статусу входів та виходів

Можливості мешканця:

1. Редагування комунальних послуг
2. Редагування особистої інформації
3. Доступ до особистого кабінету
4. Додавання повідомлень для мешканців

Роль гостя була розроблена лише для людей, які приїхали у ЖК тимчасово, він може лише перевіряти свою особисту інформацію у особистому кабінеті, також, нові користувачі системи автоматично отримують роль гостя, після чого, запрошують права у адміністратора, якщо вони проживають на території ЖК.

3.4 Структура БД

Модель сутності – атрибута – значення (EAV) – це модель даних для ефективного використання сутностей, відносно скромна. Такі сутності відповідають математичному поняттю розрідженої матриці.

Термін «база даних EAV» відноситься до структури бази даних, де значна частина даних моделюється як EAV. Проте навіть у базі даних, описаної як «заснована на EAV», деякі таблиці у системі традиційними реляційними таблицями.

У прості конструкції EAV значення атрибута є простими або примітивними типами даних, до бази даних двигун вирівняний. Системи EAV, що використовуються для представлення, можуть мати об'єкт (примірник) можуть мати підструктуру: деякі з його атрибутів можуть мати інші види об'єктів класу, які, у свою чергу, можуть мати підструктуру, щоб довільний рівень складності. У автомобіля, наприклад, є двигун, трансмісія і т. д., а у двигуна є такі компоненти, як циліндри. (Допустима підструктура для даного класу визначена в метаданих атрибутів, як обговорюється пізніше. Таким чином, наприклад, атрибут «пам'ять з довільним доступом» може використовувати клас «комп'ютер», але не до класу «двигун».)

Для представлення підструктури включають спеціальну таблицю EAV, у якій стовпець значень містить посилання інші сутності у системі (т. е. значення зовнішнього ключа у таблиці об'єктів). Щоб отримати всю інформацію про цей об'єкт, потрібно рекурсивний обхід метаданих, за яким слідує рекурсивний обхід даних, який зупиняється, коли кожен отриманий атракціон є основним (основним). Рекурсивний обхід необхідний незалежно від того, які деталі окремого класу у традиційній формі або у формі EAV; такий обхід виконується, наприклад, у стандартних об'єктно-реляційних системах. На практиці кількість рівнів рекурсії, як правило, відносно невелика для міжнародних класів, тому втрати

продуктивності через рекурсію скромні, особливо при індексуванні використання об'єктів.

EAV/CR (EAV з класами та відносинами) відноситься до структури, яка підтримує складну підструктуру. Система EAV, яка використовується в стандартній реляційній системі, залежить від того, чи є атрибутиженними або щільними. . EAV / CR дійсно дуже докладними метаданими, які досить багаті, щоб підтримувати автоматичне створення інтерфейсу для окремих класів без необхідності писати код інтерфейсу користувача для кожного класу. В тому, що можна згенерувати пакет динамічних SQL-запитів, який не залежить від класу вихідних даних, заснованих на його метаданих та використовуючи інформацію для генерації запитів до таблиць, і деякі з цих запитів можуть бути рекурсивними. . Цей підхід добре працює для запитів «об'єкт за раз», і в веб-інтерфейсі перегляду, де при цьому на ім'я об'єкта всі деталі об'єкта надані окремі: метадані, пов'язані з класом об'єкта, також представлені уявлення деталей об'єкта, тому що воно містить заголовки окремих атрибутів, порядок, у яких вони мають бути представлені, і навіть спосіб їх угруповання.

Один з підходів до EAV / CR - дозволити стовпцям містять структури JSON, які, таким чином, забезпечують використовувані класи. Наприклад, Oracle, починаючи з версії 9.4, пропонує підтримку двійкового стовпця JSON (JSONB), що дозволяє запитувати, індексувати та об'єднувати атрибути JSON.

У своєму ПП було розроблено декілька анотацій для зв’язку з БД, а саме:

1. @ObjectType – визначення типу класу. Поля: id
2. @Attribute – для полів класу. Поля: id, valueType, clazz

На рис. 2.3 зображено схема EAV моделі.

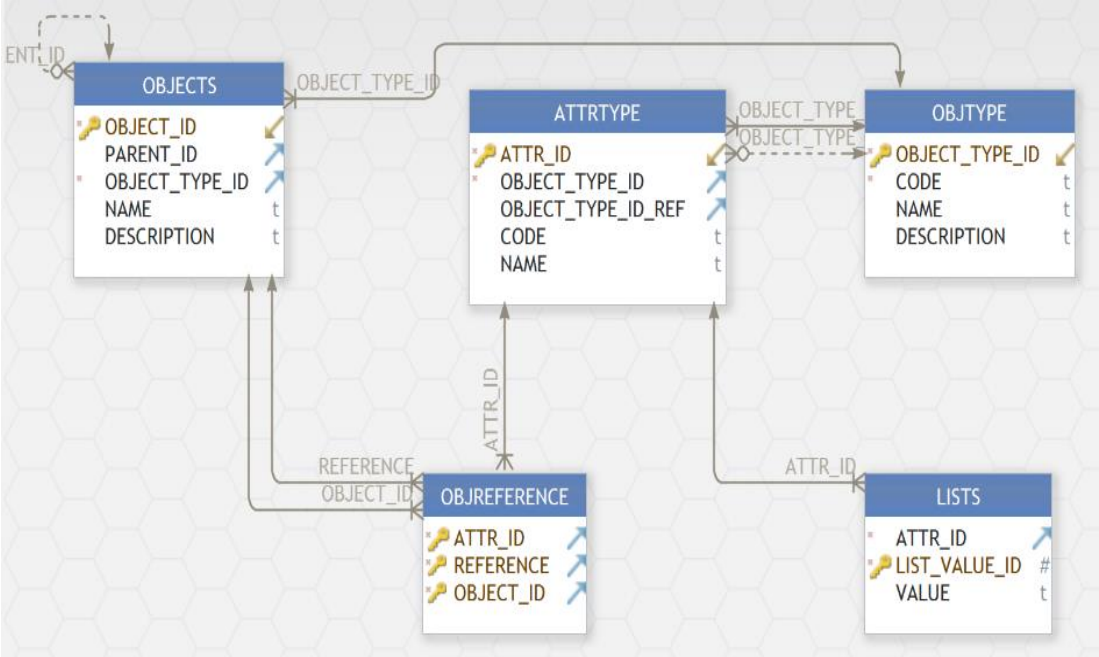

Рисунок 2.3 – Схема EAV моделі

На рис. 2.4 зображено як перераховані вище анотації використовуються у ПП.

```
6
7   @ObjectType(id = 7)
8   public class Building extends BaseEntity {
9
10      @Attribute(id = 15, valueType = ValueType.VALUE)
11      protected String number;
12
13      public String getNumber() { return number; }
16
17      public void setNumber(String number) { this.number = number; }
20  }
21
```

Рисунок 2.4 – Вигляд використання анотацій

На рисунку вище можна побачити параметри, які передаються через анотації, в анотацію @ObjectType передається ідентифікатор об’єктного типу, а в анотацію атрибута передається ідентифікатор цього атрибуту та тип його значення.

У ПП існує декілька типів значень:

1. Value – тобто є конкретне значення для цього атрибуту
2. Reference – використовується як посилання на об’єкт або атрибут
3. List value – використовується як вибір з декількох значень та присвоюється цьому атрибуту

3.5Зв’язок між лицьовою та системною частинами

Зв’язок між КІ та системною частинами зроблений за допомогою REST API.

Що таке REST API

Це англійська абревіатура, яка розшифровується та перекладається як передача стану уявлення. Web-служби, які користуються системою Representational State Transfer, застосовують термін RESTful. Відмінність цього архітектурного стилю від інших полягає в тому, що він не має єдиного стандарту, проте при цьому допустимо використовувати XML, HTTP, JSON та URL.

Підхід:

1. компоненти систем взаємодіють у значно більшому масштабі;
2. всі спільні інтерфейси;
3. частини можна запроваджувати незалежно одну від іншої;
4. є проміжні елементи, які знижують відсоток затримки та посилюють безпеку з'єднання.

Суть роботи алгоритму полягає у парі дій, залежно від типу запиту. Від роботи сервера залежить функціонал та можливості архітектури. Є 4 основних види щодо інформації:

1. get – отримання, просто передача;
2. delete - Видалення, надалі вони не відображаються;
3. post - реєстрація або додавання, реєстрація;
4. update - оновлення, регулярна операція, бази стають актуальними та свіжими.

Як пакет зазвичай відправляється JSON масив на вказаний конкретний URL. Там спрацьовує так звана функція, а залежно від вже надісланих даних та поточного запиту починається певна дія. При цьому не має значення, з якого пристрою надіслана інформація - мобільний додаток або браузер комп'ютера.

3.5.1 Розробка REST API на лицьовій стороні ПП

Найпоширеніший спосіб отримати дані від web-служб - це через HttpClient сервіс, доступний для впровадження залежностей у ваших компонентах. Angular HttpClient є досить простим. Все, що нам потрібно зробити, це викликати метод get і передати йому url. Цей метод get повертає об'єкт Observable. Цей клас є частиною бібліотеки rxjs, яка використовується у багатьох місцях Angular'a.

```
// rest.service.ts
@Injectable()
export class RestService {

  constructor(private httpClient: HttpClient) {}

  public getByObservable(url: string): Observable<any> {
      return this.httpClient.get(url);
  }

  public getByPromise(url: string): Promise<any> {
      return this.httpClient.get(url).toPromise();
  }

}
```

Подібно до обіцянки (Promise), спостерігач (Observable) не містить у собі відразу значення. Натомість у нього є метод підписки (subscribe), де ми можемо

зареєструвати зворотній виклик (callback). Цей callback викликається, як результат буде доступний. Крім обіцянки, Observable може повернути більше значення. Можна повернути собі поток результатів. Але це не має значення у цьому випадку. У нашому випадку Observable повертає лише одне значення.

```
// my-component.component.ts
@Component({ /* .. */ })
export class MyComponent {

  constructor(private rest: RestService) {}

  // Observable classic examples
  public getFields() {
    this.rest.getByObservable('http://anyurl.com').subscibe(value
=>{
      // value - результат
    },
    error => {
      // error - об'єкт помилки
    });
  }

  // Promise classic examples
  public async getAsyncField() {
    try {
      // value - результат
      const value = await this.rest.getByPro-
mise('http://anyurl.com');
    } catch (error) {
      // error - об'єкт помилки
    }
  }
}
```

3.5.2 Розробка REST API на системній стороні ПП

Розглянемо клас, який виступає контролером

```
@RequestMapping("role")
@RestController
public class RoleController {

    @Autowired
```

```
    private final RoleService service;

    public RoleController(RoleService service) {
        this.service = service;
    }

    @GetMapping
    public Page<Role> getAll(@RequestParam(value = "page", required
= false) Integer page,
                             @RequestParam(value = "size", required
= false) Integer size,
                             @RequestParam(value = "role", required
= false) String role,
                             @RequestParam(value = "sort", required
= false) String sort) {
        List<SearchCriteria> filters = new ArrayList<>();
        Pageable pageable = null;
        if (page == null && size != null) {
            pageable = PageRequest.of(0, size);
        }
        if (page != null && size != null) {
            pageable = PageRequest.of(page, size);
        }
        if (role != null) {
            filters.add(new SearchCriteria("flat", role));
        }
        return service.getAll(pageable, filters, new SortCrite-
ria(sort));
    }

    @PostMapping("/add")
    public boolean createRole(@RequestBody Role role) {
        return service.create(role);
    }

    @DeleteMapping("{id}")
    public boolean deleteRole(@PathVariable("id") Long roleId) {
        return service.delete(roleId);
    }

    @PutMapping
    public boolean updateRole(@RequestBody Role role) {
        return service.update(role);
    }
```

Завдяки таким класам, на системній стороні ПП оброблюються запроси з лицьової сторони.

4 ОГЛЯД ПРОГРАМНИХ ЕЛЕМЕНТІВ

4.1 Опис інтерфейсу гостя

Розглянемо інтерфейс гостю.

4.1.1 Особистий кабінет гостя.

На рис 3.1 зображено особистий кабінет гостя

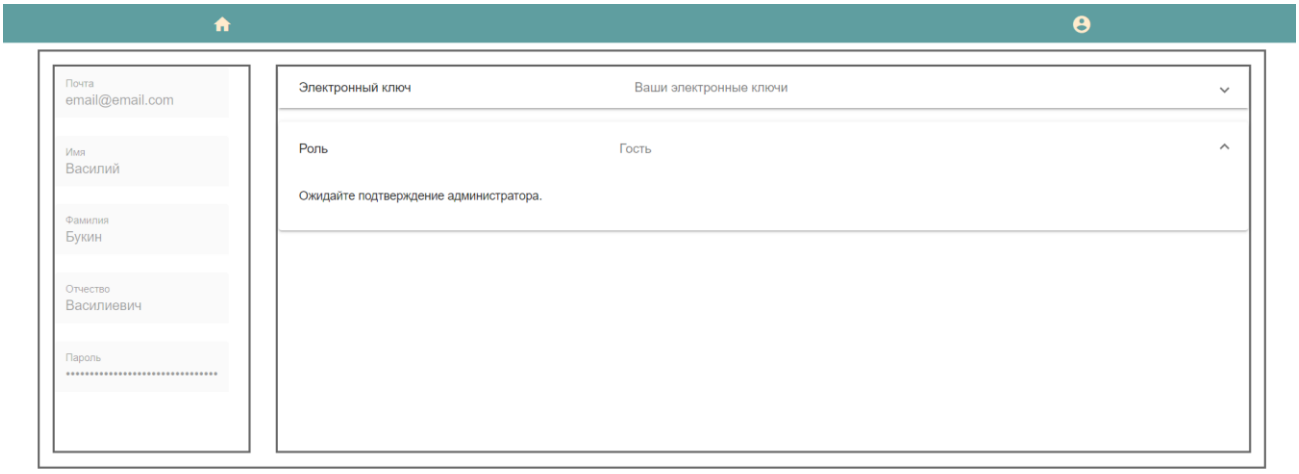

Рисунок 3.1 – Особистий кабінет гостя

На рисунку вище можна побачити, що гість не може змінювати особову інформацію. Гість не має електронних ключів та не може нічого робити окрім огляду своєї інформації.

Гість повинен дочекатися підтвердження, що він мешканець від адміністратору.

4.2 Опис інтерфейсу мешканцю

Розглянемо можливості мешканцю.

4.2.1 Вітальний екран мешканцю

На рисунку 3.2 зображено вітальний екран

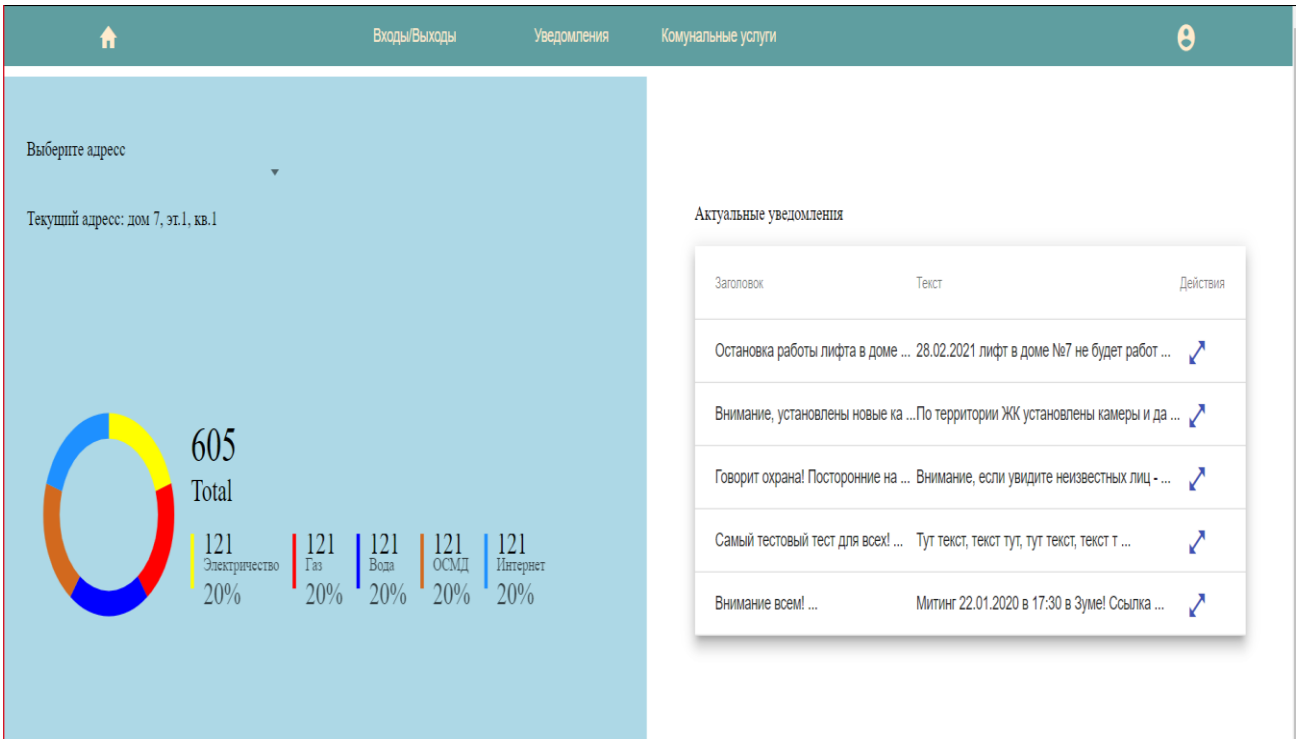

Рисунок 3.2 – Вітальний екран

На рисунку вище можна побачити актуальні повідомлення, а також, скільки грошей мешканець заплатив за комунальні послуги. Також, якщо у мешканця є декілька домівок – він може вибрати їх у меню. Отже мешканець може:

1. Перегляд актуальних повідомлень
2. Вибір адреси
3. Переглядати діаграму комунальних послуг

4.2.2 Комунальні послуги мешканцю

На рис. 3.3 зображено КІ комунальних послуг, на якому можна побачити особовий рахунок газу, води, електрики та адресу мешканця

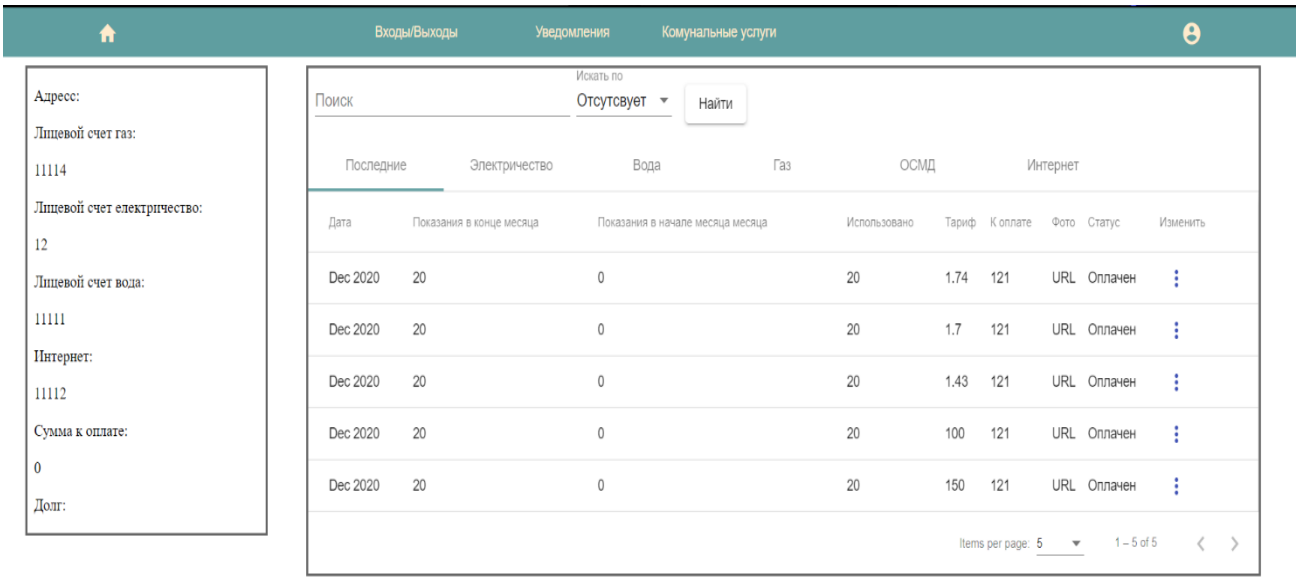

Рисунок 3.3 – Користувальницький інтерфейс комунальних послуг

На даній сторінці мешканцю доступне:

1. Управління даними про комунальні послуги, поділені за категоріями
2. Перегляд підсумкового особового рахунку
3. Пошук запису за ключовим словом

4.2.3 Повідомлення мешканцю

На рис. 3.4 зображено КІ повідомлень.

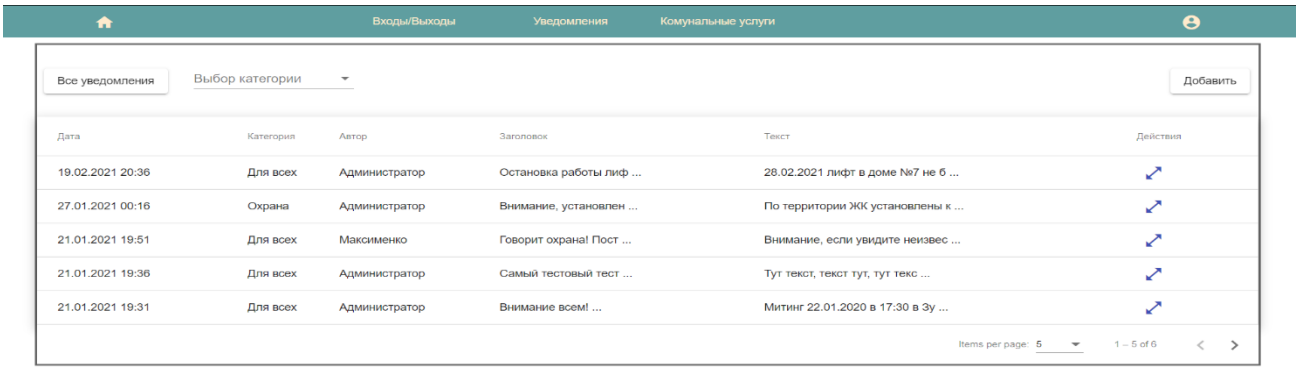

Рисунок 3.4 – КІ повідомлень

На цій сторінці мешканцю дозволено:

1. Перегляд дошки оголошень
2. Сортування оголошень за категоріями
3. Додавання оголошень

4.2.2 Входи та виходи мешканцю

На рис 3.5 зображено КІ входів та виходів

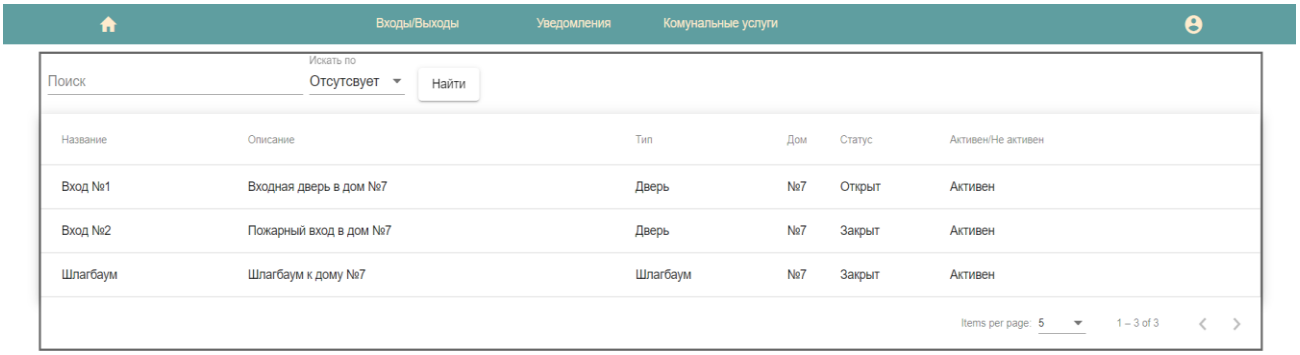

Рисунок 3.5 – КІ входів та виходів

На цій сторінці мешканцю дозволено:

1. Перегляд загальної інформації про входи/виходи комплексу
2. Пошук за назвою входу

4.3 Опис інтерфейсу охоронця

Розглянемо КІ охоронця.

4.3.1 Опис входів та виходів охоронця

На рис. 3.6 зображено КІ входів та виходів для охоронця. Охоронець може відкривати та закривати двері. Це було зроблено, для ситуацій, коли мешканець загубив свій електронний ключ, або для гостів.

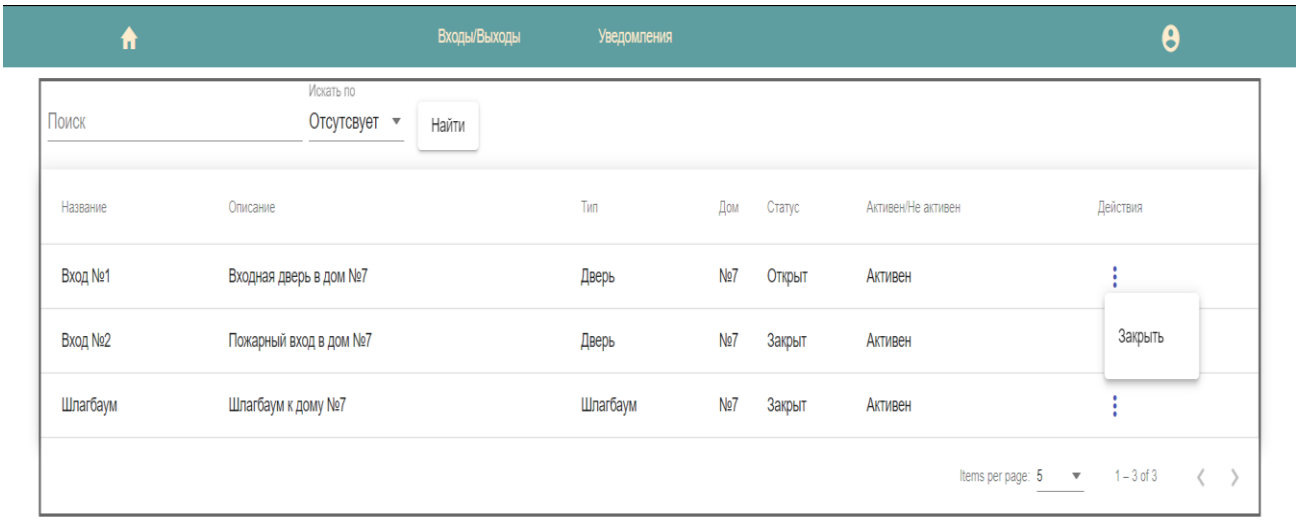

Рисунок 3.6 – КІ входів та виходів для охоронця

Охоронець може:

1. Перегляд та додавання повідомлень

2. Перегляд загальної інформації та управління входами/виходами комплексу
3. Пошук входів за ключовим словом

4.4 Опис інтерфейсу адміністратору

Розглянемо КІ адміністратору.

4.4.1 Опис інтерфейсу електронних ключів адміністратора

На рисунку 3.7 зображено КІ електронних ключів

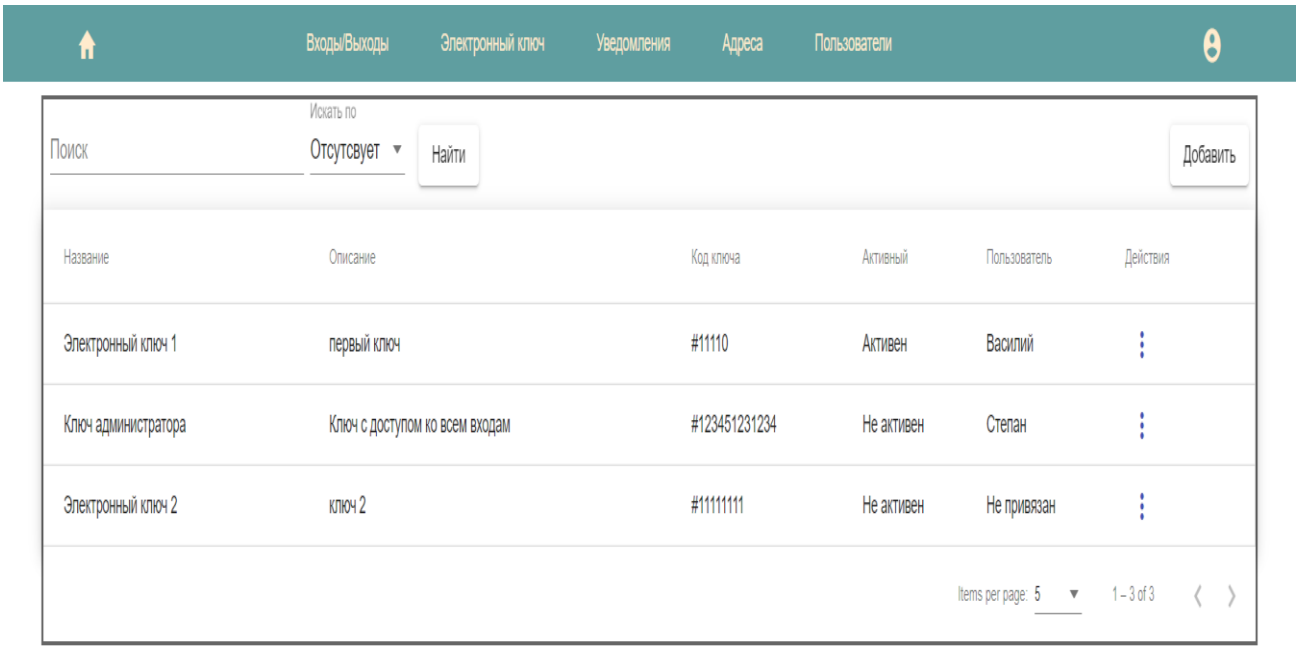

Рисунок 3.7 КІ електронних ключів

На цьому КІ адміністратор може:

1. Управління електронними ключами:
    a. Додавання електронного ключа
    b. Видалення електронного ключа
    c. Прив’язка електронного ключа к користовачеві
2. Переглядати інформацію про електронні ключі

Також на цій сторінці адміністратор може видати у тимчасове використання електронний ключ гостю або мешканцю

4.4.2 Опис інтерфейсу адрес

На рисунку 3.8 зображено КІ адрес.

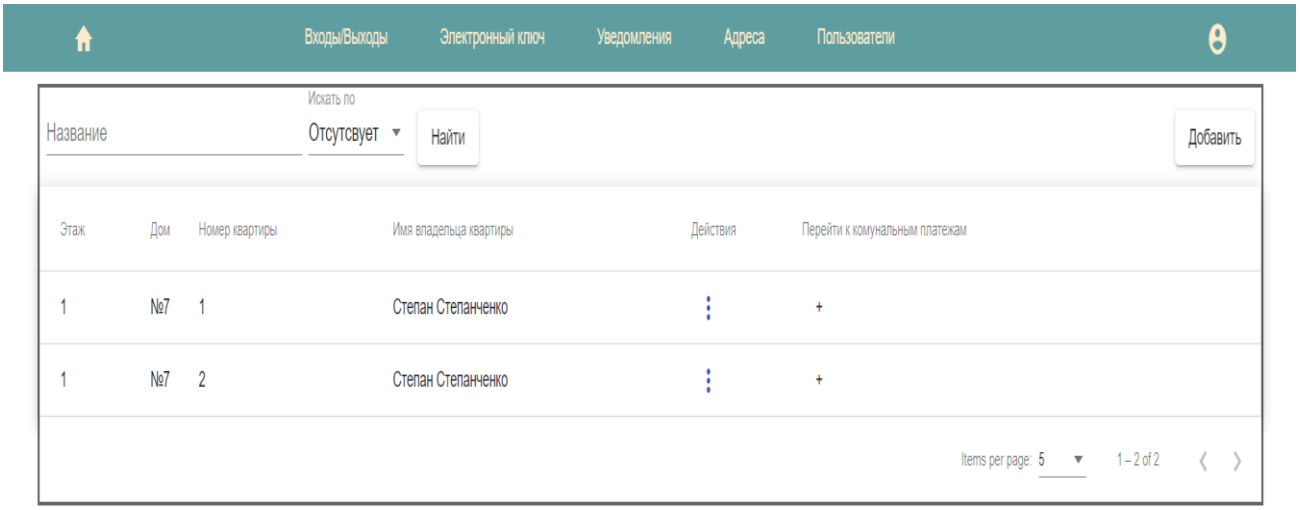

Рисунок 3.8 – КІ адрес

На цій сторінці адміністратор може:

1. Додавати адреси
2. Редагувати
3. Видаляти

Також з цієї сторінці можна зайти на сторінку комунальних платежів та потрапити на обрану адресу.

4.4.3 Опис інтерфейсу користувачів

На рисунку 3.9 можна побачити КІ користувачів.

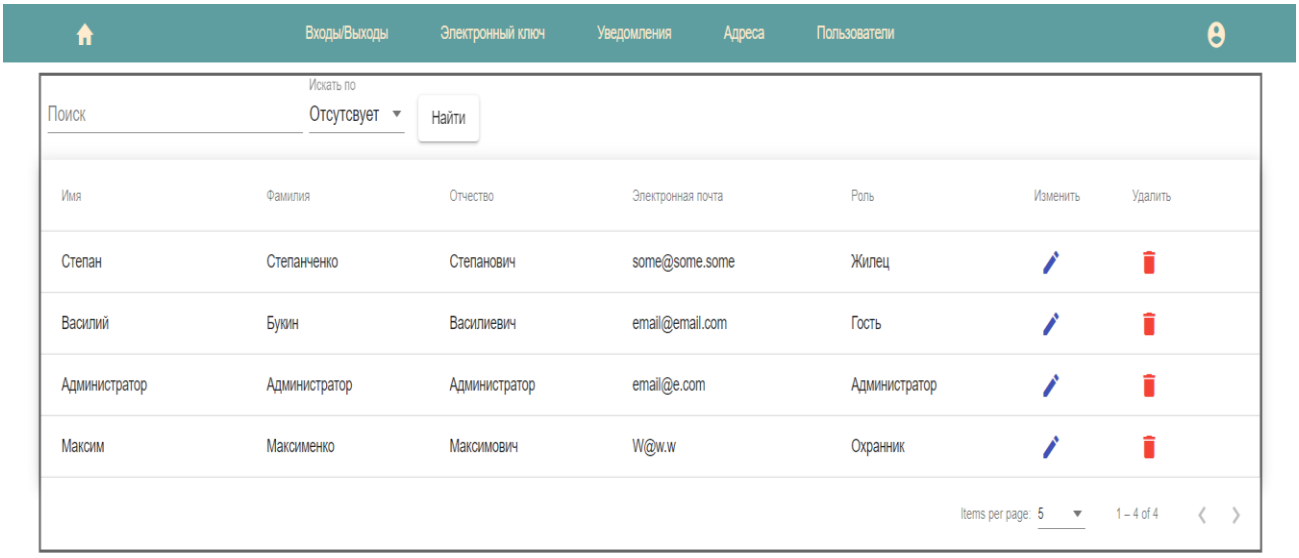

Рисунок 3.9 – КІ користувачів

На цій сторінці адміністратор може:

1. Видаляти мешканців та гостей
2. Прив’язати електронний ключ для мешканця
3. Надати роль мешканця для гостя

Також можна сортувати гостей по ролях, та шукати по критеріях.

ВИСНОВКИ

У магістерській роботі було відзначено переваги та недоліки розробленої системи для автоматизації деяких процесів у сфері діяльності житлового комплексу та визначено методи вирішення проблематики та модернізування сучасних технологій моніторингу. Оскільки дослідження в цій галузі були узагальненими, з урахуванням цих обставин література розглянута в цьому дослідженні, розкриває загальні переваги та наслідки застосування системи для автоматизації. Тому можна підкреслити, що вона допомагає керувати зростаючими вимогами до робочого навантаження з обмеженою робочою силою. Завдяки проведеним дослідженням вивчення переваг та недоліків використання програми в конкретній сфері застосування, було досягнуто ефективного додатку з вирішеною проблемою для майбутньої робочої сили.

Як підсумок впровадження інформаційно-комунікаційної технології житлового комплексу з реалізованою автоматизацією функцій зазначених у даній роботі, спостерігається підвищення ефективності роботи працівників житлового комплексу та якості обслуговування мешканців, зі скороченням ресурсоспоживання та розширенням інформаційної обізнаності. Як вихідний продукт ми маємо систему менеджменту ЖК, яка здатна керувати середовищем території з деякими функціями безпеки та житлово-комунальними послугами. Незважаючи на те, що велику кількість функцій можна додати у майбутньому, готовий продукт вже на даному етапі є доступною програмою для загального користування.

На основі аналізу були встановлені основні фактори, що впливають на розвиток автоматизованих систем будівельних комплексів. Виділено основні завдання для будівельного комплексу, а саме: підвищення якості життєзабезпечення мешканців, швидкий обмін інформацією, доступ до управління головних дверей всім користувачам та зручне використання житлово-комунальних послуг.

Отже, було сформовано ефективну цілісну систему управління будівельним комплексом, яка спрямована на подолання будівельного спаду, а також задоволення та покращення життєдіяльності населення на основі прийняття ефективних економічних, фінансових рішень для можливої реалізації прийнятих рішень. У подальшому вдосконаленню програми є можливим, на основі вивчення зарубіжного та вітчизняного досвіду, розробити організаційно-комунікаційну модель управління будівельним комплексом з більшою кількістю доступних функцій.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Установка JDK URL: https://khasang.io/courses/java0/lectures/1214779 (дата звертання 17.01.2021)
2. MySql – система управления базы данных – « Веб Креатор». URL: https://web–creator.ru/articles/mysql (дата звернення 25.02.2021).
3. СУБД Oracle. URL (дата звернення 25.02.2021).: https://ru.wikipedia.org/wiki/Oracle
4. СУБД SQLite. URL (дата звернення 25.02.2021).: https://ru.wikipedia.org/wiki/SQLite
5. Java програмна платформа – Вікіпедія. URL: https://ru.wikipedia.org/wiki/ Java_(программная_платформа) (дата звернення 2.02.2021).
6. Java Runtime Environment (сокр. JRE; укр. середовище виконання для Java) – Вікіпедія. URL: https://ru.wikipedia.org/wiki/Java_Runtime_Environment (дата звернення 2.02.2021).
7. Начало работы с .NET Framework | Microsoft Docs. URL: https://docs.microsoft.com/ru–ru/dotnet/framework/get–started/ (дата звернення 15.02.2021).
8. Eclipse – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Eclipse (дата звернення 10.03.2021).
9. UML – Википедия URL: https://ru.wikipedia.org/wiki/UML (дата звернення 12.03.2021)
10. NetBeans – Википедия URL: https://ru.wikipedia.org/wiki/NetBeans (дата звернення 25.02.2021).
11. Java Runtime Environment (сокр. JRE; укр. середовище виконання для Java) – Вікіпедія. URL:

https://ru.wikipedia.org/wiki/Java_Runtime_Environment (дата звернення 20.02.2021)

12. IntelliJ Idea – Вікіпедія. URL: https://uk.wikipedia.org/wiki/IntelliJ_IDEA (дата звернення 10.03.2021)
13. Spring Framework – Вікіпедія. URL: https://ru.wikipedia.org/wiki/Spring_Framework (дата звернення 25.03.2021).
14. Angular Framework – Вікіпедія. URL: https://ru.wikipedia.org/wiki/Angular_Framework (дата звернення 25.02.2021).
15. Angular Material – Офіційний сайт. URL: https://material.angular.io/ (дата звернення 26.03.2021).
16. TypeScript – Вікіпедія. URL: https://ru.wikipedia.org/wiki/TypeScript (дата звернення 26.03.2021).
17. EAV/CR сутність БД. URL: https://star-wiki.ru/wiki/Entity%E2%80%93attribute%E2%80%93value_model