Plagiarism Detector .net

Date: June, 20 2022

Plagiarism Scan Report

2%

Plagiarised

98%

Unique

Words Statistics

Words	14937
Characters	115097

Exclude URL None

Content Checked For Plagiarism

МІНІСТЕРСТВО ОСВІТИ НАУКИ УКРАЇНИ ПрАТ «ПВНЗ "ЗАПОРІЗЬКИЙ ІНСТИТУТ ЕКОНОМІКИ ТА ІНФОРМАЦ ТЕХНОЛОГІЙ"» Кафедра економічної кібернетики та інженерії програмного забезпечення ДО ЗАХИСТУ ДОПУЩЕН кафедри _____ д.е.н., доц. С.І. Левицький БАКАЛАВРСЬКА ДИПЛОМНА РОБОТА ПЕРСОНАЛ ПОМІЧНИК УПРАВЛІННЯ РОЗКЛАДОМ ПрАТ «ПВНЗ «ЗІЕІТ» Виконав ст. гр. ІПЗ – 118 _____ М.Д. Гнутенко Науковий керівник доцент _____ О.А. Жеребцов Запоріжжя 2022 р «ПВНЗ «ЗАПОРІЗЬКИЙ ІНСТИТУТ ЕКОНОМІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ» Кафедра економічної кіберне інженерії програмного забезпечення ЗАТВЕРДЖУЮ Зав. кафедрою д.е.н., доцент Левицький С.І. _____ 17.01.2022 р. ЗАВДАННЯ НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ Студенту гр. ІПЗ – 118, спеціальності «Інж програмного забезпечення» Гнутенко Максиму Денисовичу 1.Тема: Персональний помічник управління розкладом ПВНЗ "ЗІЕІТ" затверджена наказом по інституту 06.1-50 від 15 січня 2022 р. 2. Термін здачі студентом закінченої роб червня 2022 р. 3. Перелік питань, що підлягають розробці 1. Здійснити огляд проблематики роботи з навча розкладом ПрАт ПВНЗ ЗІЕІТ. 2. Розглянути варіанти та технології щодо оптимізації механізмів управління розкла Виконати огляд та порівняння аналогів систем з інформування розкладу . 4. Здійснити обґрунтований вибір викори стеку технологій для розробки проекту . 5. Розробити алгоритми парсингу вихідної інформації розкладу. 6. Зді проектування та програмування запропонованої системи. 7. Виконати операції життєвого циклу програмного забезп щодо створеного продукту. 8. Оформити звіт за результатами роботи. 4. Календарний графік підготовки бакала дипломної роботи № етапу Зміст Терміни виконання Готовність по графіку %, підпис керівника Підпис керівника про готовність етапу, дата 1 Формулювання теми магістерської (бакалаврської) дипломної роботи, (збір практичного ма за темою магістерської дипломної роботи 2 І атестація І розділ бакалаврської дипломної роботи 3 ІІ атестація ІІ бакалаврської дипломної роботи 4 ІІІ атестація ІІІ розділ бакалаврської дипломної роботи, висновки та реком додатки, реферат, перевірка програмою «Антиплагіат» 5 Доопрацювання бакалаврської дипломної роботи, підг презентації, отримання відгуку керівника і рецензії 6 Попередній захист бакалаврської дипломної роботи 7 І бакалаврської дипломної роботи на кафедру 8 Захист бакалаврської дипломної роботи Дата видачі завдання: 17.0 р. Керівник бакалаврської роботи _____ О.А. Жеребцов (підпис) (прізвище та ініціали) За отримав до виконання _____ М.Д. Гнутенко (підпис студента) (прізвище та ініціали) РЕ Бакалаврська робота містить 0 сторінок, 0 рисунків, 0 таблиць, 0 використаних джерел. Метою розробки є ств персонального помічника для сповіщення про розклад з персональними налаштуваннями. Об'єктом дослідж сучасна система відображення розкладу занять. Предметом дослідження є чат-бот для відображення персона розкладу. Здійснено детальний огляд предметної області, та сучасних аналогів, таких як чат-бот для перевірки ро від ЗВО НУПП, та модуль розкладу АСУНЗ Національного Медичного Університету. Виявлено, що розробка с сповіщення про новий розклад є доцільною. Проект реалізовано у виді чат-боту для месенджеру Telegr використанням мови програмування Java та бібліотеки Apache POI. Отриманий програмний продукт є про використанні, та гнучким у налаштуванні. Система сповіщення дозволяє своєчасно отримувати необхідну інформа

файлів. Актуальність даної дипломної роботи полягає в удосконалені системи публікації розкладу та реа оперативного інформування викладачів та студентів за допомогою персонального помічника. РОЗДІЛ 1 (ПРЕДМЕТНОЇ ОБЛАСТІ ТА АНАЛОГІВ 1.1 Стан системи публікації розкладу ЗІЕІТ Розклад для інституту та коледж [1] складається з декількох частин: * Розклад занять. * Розклад зайнятості викладачів. * Розклад консультацій. частина – це один або декілька документів. Розклад занять, зайнятості та консультацій формується автомати допомогою системи «Деканат», у виді таблиць формату Excel (.xls або .xlsx). Після цього, при необхідності реда навчальним відділом. Така система дозволяє дуже просто публікувати розклад по двом напрямам: * Друкув вивішувати на дошці розкладу занять. * Публікувати на офіційному сайті ЗІЕІТ. Слід зауважити, що розклад зайнят публікується на сайті, а відправляється кожному викладачу індивідуально через приватні канали передачі даних. Н розклад можна завантажити у первісному форматі – xls або xlsx. Достатньо знайти необхідний курс на натисн посилання «завантажити». Вигляд меню з розкладом показано на рисунку 1.1. Рисунок 1.1 ▲ Меню розкладу на сайт Інформування про оновлення розкладу в цій системі відсутнє. За виключенням інформування викладачів за допо приватних каналів обміну інформацією. Це стан системи публікації розкладу ЗІЕІТ. Така система не но використовується достатньо давно багатьма ЗВО. Такий висновок був зроблений на підставі простого дослідження розкладу різних навчальних закладів. Про це у наступному підрозділі. 1.2 Механізми електронного відображення ро різних ЗВО На стан 2020 року в Україні нараховується більше 280 вищих навчальних закладів [2]. Кожен навчальних закладів самостійно вирішує, який спосіб електронного відображення розкладу використовувати. навчальних закладів публікують первісні Excel таблиці на офіційному сайті. Цей спосіб використовує і ЗІЕІТ. Е багато переваг, але має і недоліки. Наприклад, така система потребує розділення розкладу на частини – р студентів та розклад викладачів. Розклад зайнятості викладачів зазвичай дуже нечитабельний, особливо для навантажених тижнів. Інші використовують сторонні системи управління розкладом, такі як АСУНЗ (Автомати система управління навчальним закладом) [3]. Такі системи дозволяють формувати розклад та відображуват одразу на сайті, або в мобільному додатку, з можливістю вибирати як розклад студентів, так і розклад викладач навчальні заклади мають систему авторизації, тому оцінити їх систему публікації розкладу сторонньому спост неможливо. У якості прикладів було відібрано декілька крупних ЗВО Запоріжжя та інших міст України: * Запо Національній Університет. * Запорізький Державний Медичний Університет. * Київський Європейський Університ наприклад, Запорізький національний університет публікує розклад за схожим принципом, який використовується в публікує Excel таблиці з розкладом на сайті [4]. Студент має завантажити потрібний файл з розкладом та відкрити програму зчитування таблиць формату .xls. На рисунку 1.2 приведений знімок розділу з розкладом на їх офіційному Сам розклад має структуру, яка показана на рисунку 1.3. На рисунку приведена його частина, бо вся табл поміститься у формат Word документу. Ця структура побудована за схожим принципом, який використовується в Зверху розташовані коди груп, зліва день тижня та номер пари. На їх перетині знаходяться самі клітини з парами, зазначені ім'я, тип пари, та ПІБ викладача. Рисунок. 1.2 – Сторінка з розкладом занять на офіціальному сайті ЗНУ Р 1.3 – Частина розкладу занять інженерного факультету ЗНУ А ось приклад розкладу Запорізького Державного Мед Університету [5], на рисунку 1.4. Це також Excel таблиця, яку можна завантажити на сайті цього ЗВО. Його структур відрізняється від приведеної вище. Тут немає ділення на групи, а є лише день тижня, номер та час пари. Також, в кл з інформацією про пару, не зазначається ПІБ викладача. Рисунок. 1.4 – Приклад розкладу ЗДМУ Ще один навчальний заклад – Київський Європейський Університет [6]. Його система відображення розкладу також пол публікації Excel таблиць на офіційному сайті. Сам вид таблиці приведений на рисунку 1.5. Це розклад занять п курсу економічного факультету. Його структура схожа зі структурою розкладу ЗІЕІТ за невеликими виключе Рисунок. 1.5 – Приклад розкладу Київського Європейського Університету А ось приклад використання системи АУН яку було написано раніше. Роботу цієї системи можна побачити на прикладі Національного медичного університет О.О. Богомольця [7]. В лівому меню можна вибрати тип розкладу – студента, викладача, аудиторії, тощо. На рису можна побачити приклад відображення розкладу студентів певного курсу та певної групи, завдяки цієї системи. Р 1.6 – Приклад відображення розкладу студентів А на рисунку 1.7 можна побачити відображення розкладу виклад допомогою системи АУНЗ. Також опціонально можна вибрати діапазон дат та інші параметри. Рисунок. 1.7 – П відображення розкладу викладача Остання система, яку можна привести – це Telegram чат-бот для перегляду ро від Національного Університету "Полтавська Політехніка Імені Юрія Кондратюка" [8]. Приклад роботи з ним ви рисунку 1.8. Рисунок 1.8 – Чат-бот від НУПП Бот надає можливість перегляду розкладу різних груп. Також він має функцію нагадування про початок пари заздалегідь. Це різні системи електронного відображення розкладу навчальних закладів України. Кожна система має свої переваги та недоліки, тому вибрати кращу буде важно. Якщ яка система відображення розкладу виконує поставлені цілі та задовольняє персонал та студентів, можна вважати

правильним. 1.3 Механізми автоматичного сповіщення користувачів Механізм сповіщення призначений для операт інформування користувача про змінення стану певної системи, в нашому випадку - розкладу. Існує багато сп сповіщення користувачів. Найменш затратним та швидким можна вважати сповіщення через мережу інтернет. приведені основні способи масового сповіщення користувачів через інтернет: * електронна пошта; * спеціал додатки; * месенджери; * повідомлення від веб додатку через систему сповіщення браузера. Як і з механ відображення розкладу, кожен спосіб має переваги та недоліки, та призначений для різних задач. Одна з задач дип роботи - знайти оптимальний спосіб сповіщення користувачів про новий розклад. Для цього треба відмітити пере недоліки кожного з представлених методів. Ці переваги та недоліки виділені у таблиці 1.1. Таблиця 1.1 ▲ Перев недоліки різних методів сповіщення Метод Переваги Недоліки Електронна пошта Висока доступність Можл відправити з будь якого додатку Необхідність постійно перевіряти повідомлення на наявність нових Відсутність ш запитів до системи Спеціалізовані додатки Повідомлення одразу видно у меню повідомлень Можливість реал додатковий функціонал Необхідність встановити на телефон/пк Необхідність писати окремий кросплатформений д Месенджери Висока доступність Повідомлення одразу видно у меню повідомлень Можливість реалізувати дода функціонал за допомогою чат-боту Необхідність встановити (але більшість користувачів вже використовують месенджер) Повідомлення від веб додатку через систему сповіщення браузера Висока доступність Можл реалізувати додатковий функціонал Не всі браузери мають цю функцію або реалізують її так як треба Необхідність активувати цю функцію Повідомлення можуть мати обмежений об'єм інформації Слід зауважити, що деякі недоліки, необхідність встановлювати окремі додатки, нівелюються корисністю додатку та залежить від цілей, для як використовується. Наприклад такі додатки як месенджер встановлені майже на всіх мобільних пристроях, бо вони п та виконують багато функцій. Але може бути недоцільно використовувати окремий додаток тільки для повідомлень. перейшли до аргументації вибору системи автоматичного сповіщення користувачів про оновлений розкл Аргументація вибору системи автоматичного сповіщення та відображення розкладу У попередньому підрозділ розглянуті переваги та недоліки різних систем сповіщення користувачів через мережу інтернет. Як вже було зазн деякі переваги та деякі недоліки можуть грати ключове значення при виборі цієї системи для певних Найголовнішими критеріями обрання технічних засобів для доставки інформаційних повідомлень учасникам є шви доступність, кросплатформеність, легкість та зручність у використанні. Одна з задач дипломної роботи – в найбільш оптимальну систему для сповіщення про новий розклад. Після невеликого дослідження був вибраний передачі інформації через месенджер, а саме через чат-бот для месенджеру «Telegram». Цей підрозділ виділен аргументації цього вибору. Для вибору оптимального методу, визначимо вимоги до системи сповіщення. С сповіщення про новий розклад повинна: * Завжди відображати повідомлення про новий розклад. Виключенням мож лише відсутність підключення до мережі інтернет у користувача. * Бути доступною для більшості людей. Це в можливість отримувати сповіщення на різних мобільних або стаціонарних пристроях. * Мати можливість включ виключати повідомлення без відключення від мережі інтернет. * Відображати розклад у найбільш доступном швидкого перегляду форматі для більшості користувачів – растровому зображенні. Тепер, коли вимоги для с складені, видно, що месенджер найбільш вдало підходить до поставлених вимог. На це вказують властивос популярних месенджерів: * Повідомлення будуть завжди відображатись, за виключенням ситуацій, коли кори вручну відключив цю можливість у додатку. * Месенджери доступні для всіх користувачів, та стоять на біл мобільних пристроїв та персональних комп'ютерах. * Завдяки простим командам, які часто доступні як кнопки, активувати та деактивувати повідомлення. * Усі популярні месенджери підтримують текстові та голосові повідом відправку зображень та інших медіа файлів різних форматів. На даний момент в Україні лідирують декілька месенд Telegram [9] , Viber [10], WhatsApp [11], Facebook Messenger [12]. За даними на 2020 рік, в Україні месенд «Telegram» користується приблизно 6 млн користувачів. Це значить, що принаймні у кожного сьомого ук встановлений цей месенджер. В 2021 році статистики ще немає, але вважаючи постійний ріст кількості користувачів додатку у світі, можна зрозуміти, що кількість користувачів в Україні стала більше. Незважаючи на те, що існують і менш популярні месенджери, «Telegram» сильно виділяється від своїх конкурентів не в останню чергу з можливості створення чат-ботів. На даний момент це самий популярний месенджер з можливістю створити чат-бот яким користувачем, без реєстрації компанії або інших недоступних для більшості розробників дій. Пер використання чат-ботів можна вважати: миттєвість доставки повідомлення, доступність 24/7, швидке інформув автоматичному режимі, кросплатформеність, відсутність необхідності встановлення додаткового прогр забезпечення для більшості користувачів, можливість заміни людини експертною системою або штучним інте непомітно для кінцевого користувача. Саме цьому в якості системи сповіщення, відображення, та управління підп на розклад, був вибраний чат-бот на платформі «Telegram». 1.5 Проблеми отримання даних про розклад Весь р

ЗІЕІТ генерується автоматично системою деканат у формат Excel таблиць. Дані для генерації цих таблиць знаходя
базі даних системи деканат. На перший погляд найбільш придатним варіантом для отримання даних про розклад є
з зовнішнім API системи деканат, або з їх базою даних. Але це не так по деяким причинам. Перша причина полягає
що система деканат ЗІЕІТ не має ніякого зовнішнього API. Друга причина полягає у проблемах з актуальністю да
можуть знаходитись в базі даних системи деканат. Ця проблема виходить з того, що вихідні генеровані таблиці ч
редагуються учбовим відділом для внесення змін у порядок та дані про пари. Після таких змін єдиним актуа
джерелом даних про розклад є фінальні редаговані таблиці. По цим причинам єдиний спосіб отримання актуальних
про розклад – парсинг таблиць розкладу, які публікуються на офіційному сайті ЗІЕІТ. 1.6 Висновки розділу Було зді
огляд поточної системи публікації розкладу ЗІЕІТ та інших ЗВО. Встановлено потребу у створенні системи шв
сповіщення про новий розклад зі зручним переглядом та персональними налаштуваннями користувача. Також
здійснено огляд різних систем сповіщення користувачів та встановлено доцільність використання месенджерів. РО
МЕТОДОЛОГІЯ ТА ЗАСОБИ РОЗРОБКИ 2.1 Парсинг Парсинг (частіше називається синтаксичним аналізом [13]
процес розбору та структуризації природної або формальної мови. Як правило, результатом синтаксичного ан
синтаксична будова речення, представлена або у вигляді дерева залежностей, або у вигляді дерева складових
вигляді деякого поєднання першого та другого способів уявлення. Парсинг часто застосовується разом з лекс
аналізом. У сучасній розробці «парсингом» часто називають збір даних зі вже підготовлених структур, які пройшли
лексичного аналізу та побудову абстрактного синтаксичного дерева. До такого «парсингу» часто відносять парсин
документів. Це зумовлено тим, що для більшості формальних мов вже існують парсери, тому проводити лексич
синтаксичний аналіз мов по типу XML та HTML частіше всього не треба. Парсинг Excel таблиць схожий за цим прин
Це зумовлено тим, що формат XLSX, який використовується з 2007 року – це ZIP архів з файлами у форматі X
містять розмітку осередків. Більш детальна інформація про формат XLSX знаходиться у наступному підрозділі. П
таблиць, можна назвати парсингом частково структурованих даних, при умові, що структура та порядок осередків т
незмінні. Це зумовлено незначними періодичними змінами формату даних в осередках таблиці. Наприклад,
людський фактор, ПІБ викладача, або номер аудиторії може містити, або не містити деякі елементи, такі як сі
пробілу, крапки, тощо. Зважаючи на це, необхідно проводити додатковий синтаксичний аналіз контенту таблиц
вичленення окремих даних, таких як ПІБ викладача або номер аудиторії, з урахуванням та виправленням мож
помилок. Універсального парсеру для будь якої таблиці не існує. Для різних типів таблиць потрібен власний п
власними типами даних. Використання техніки машинного навчання може бути універсальним підходом для різни
таблиць. Але дослідження [14], які проводилися для впровадження машинного навчання у документообіг показали,
алгоритми є доцільними лише для строго структурованих даних, для отримання точних результатів. Так як розклад
на даний момент не є строго структурованим, є великий ризик отримання помилок у вихідних документах, а пом
розкладі є недопустимими. Тому машинне навчання, як універсальний метод розбору даних таблиць розкладу, на
момент не можна вважати доцільним. 2.2 Рендеринг Рендеринг [15] - термін у комп'ютерній графіці, що означає
отримання зображення за моделлю за допомогою комп'ютерної програми. Де модель - це опис будь-яких о
представлений суворо обмеженою мовою, або у вигляді структури даних. Такий опис може містити геометричн
положення точки спостерігача, інформацію про освітлення, ступінь наявності якоїсь речовини, напруженість фіз
поля та ін. У контексті рендерингу розкладу, моделлю є віртуальна таблиця, створена за допомогою даних, отрим
процесі парсингу. На даний момент розроблено безліч алгоритмів візуалізації. Існуюче програмне забезпечення
використовувати кілька алгоритмів для отримання кінцевого зображення. Для рендерингу таблиць добре під
растеризація [16]. Кінцевим результатом растеризації є растрові зображення. Рендеринг таблиці дозволяє перегля
на більшому діапазоні пристроїв, особливо мобільних, ніж перегляд у специфічному форматі, такому як xls, csv, або
2.3 Таблиці Excel Одні з основних форматів, які Excel підтримує для редагування – це XLS [17] та XLSX. Друг
представлений разом з Microsoft Excel 2007. Повна специфікація існує на сайті Microsoft та займає приблизн
сторінок [18]. Тому у цьому підрозділі буде вказана лише базова структура цих форматів. Обидва файла це ZIP арх
містить файли з інформацією про листи та їх розмітку. У той час, як XLS містить бінарні файли, XLSX зберігає інформ
форматі XML. Порожня книга, розпакована до її файлів, містить такі складові файли та папки. * [Content_Types].xm
єдиний файл, який знаходиться на базовому рівні, розпакованого zip. У ньому перелічено типи вмісту для частин у
Усі посилання на файли XML, що входять до пакета, містяться в цьому файлі XML. * _rels - Це папка Relationshi
містить один файл XML, у якому зберігаються відносини на рівні пакета. Посилання на ключові частини файл
містяться в цьому файлі як URI. Ці URI визначають тип зв'язку кожної ключової частини з пакетом. Це включає зв
основним офісним документом, розташованим як xl/workbook.xml, та іншими частинами в docProps як основн
розширеними властивостями. * docProps - Ця папка містить загальні властивості документа. Вони включають

набір основних властивостей, набір розширених властивостей або властивостей конкретної програми та попе
перегляд мініатюр документа. Порожня книга містить у цій папці два файли, а саме app.xml і core.xml. Файл co
містить таку інформацію, як автор, дата створення, збереження та зміни. App.xml містить інформацію про вміст фай
- Це основна папка, яка містить усі відомості про вміст книги. Для кожного аркуша Excel, що міститься в книзі, є один
XML. Ці файли XML можна знайти в папці xl/worksheets. Вся інформація, що міститься на аркуші, організована
розділи XML-файлу. 2.4 Регулярні вирази Регулярні вирази – це формальна мова для пошуку підстрок у стро
маніпуляцій з ними. Для пошуку використовується рядок-зразок (pattern), що складається з символів та мета сим
задає правило пошуку. Шаблон регулярного виразу складається із звичайних символів, наприклад abc, або ком
звичайних та спеціальних символів, наприклад ab*c або Chapter (\d+)\.\d*. Останній приклад включає дуж
використовуються для позначення групи. Шаблон усередині групи обробляється як єдине ціле. У разі коли
відповідності вимагає чогось більшого, ніж пряме зіставлення, наприклад знаходження послідовності символ
знаходження пробілу, шаблон включає спеціальні символи. Регулярні вирази широко використовуються для визн
правил формальних мов в різних трансляторах. Регулярні вирази добре підходять для аналізу та вичлення д
осередках таблиць розкладу. 2.5 Архітектура «Клієнт-Сервер» Архітектура клієнт-сервер – це обчислювальна мо
якій сервер розміщує, постачає та керує більшістю ресурсів і послуг, які споживає клієнт. Цей тип архітектури має од
кілька клієнтських комп'ютерів, під'єднаних до центрального сервера через локальну мережу або мережу Інтерн
система спільно використовує обчислювальні ресурси. Архітектура клієнт/сервер також відома як модель мер
обчислень або мережа клієнт/сервер, оскільки всі запити та послуги доставляються через мережу [19]. Приклад стр
клієнт-сервер показана на рисунку 2.1. Рисунок 2.1 – Архітектура Клієнт-Сервер Кожен сервер у такій архітектурі
деякий інтерфейс, за яким клієнти можуть надсилати запити до цього серверу. У разі вхідного запиту сервер обр
його, та відсилає відповідь, на яку чекає клієнт. 2.6 Архітектурні принципи REST REST (абревіатура Representation
Transfer) – це набор правил, або архітектурний стиль взаємодії компонентів додатку у мережі. Такі правила дозв
розробляти масштабовані системи, які можуть легко обмінюватись даними між собою. REST не обмежує архітек
виборі технології передачі даних, але вводить 5 обов'язкових правил для такої системи [20]: 1. Однорідність інтер
2. Архітектура Клієнт-Сервер. REST передбачає саме таку архітектуру. Це забезпечує розділення проблем, що доп
компонентам клієнта і сервера розвиватися незалежно. 3. Відсутність стану. Кожен запит від клієнта до сервера п
містити всю інформацію, необхідну для розуміння та виконання запиту. Стан може зберігати клієнт, але не сер
Кешування. Клієнт за необхідності може кешувати результат деяких запитів для підвищення швидкості роботи сист
Шари. Багатошарова система дозволяє структурована так, що кожен компонент не може бачити за межами свого ша
дозволяє масштабувати систему без риску зламати щось в іншому місті. REST не передбачає, у якому виді
передаватись дані. У сучасних системах дані частіше передаються у виді JSON строк, ріже у виді XML. У
навантажених системах часто використовують Protobuf, який дозволяє описувати дані на рідній мові програмува
серіалізувати їх у максимально стислий масив байтів. Також, REST не передбачає, який протокол передачі дани
використовуватись. Але на практиці частіше всього використовується HTTP, як один з найпопулярніших при
протоколу типу клієнт-сервер. Для бінарних даних також використовують протокол HTTP/2, так як він є
оптимізованим варіантом ніж HTTP/1.1, так я є бінарним, та підтримує потоки даних. 2.7 Чат-боти Чат-бот, або вірту
співбесідник – це програма, управління якою здійснюється через чат іншої програми, наприклад месенджеру, соц. м
тощо. Управління чат-ботом може здійснюватися різними методами, такими як текстові команди, віртуальна клав
запити на звичайній мові, голосові запити, тощо. Сьогодні чат боти застосовують для самих різних потреб як
компанії, так і для індивідуальних цілей. Для прикладу, на рисунку 2.2 приведений скриншот роботи офіційного ча
онлайн банку Монобанк [21]. Такий чат бот значно економить час користувачів. Рисунок 2.2 - Чат-бот онлайн
Монобанк Чат боти можна поділити на ті що навчаються, та «заскриптовані». Боти, що навчаються використ
машинне навчання для покращення результатів своїх відгуків на запити користувачів. Прикладом таких чат
популярний проект «A.L.I.C.E» від компанії «Яндекс». Заскриптовані чат боти зазвичай використовують в своєї
машину станів, яка є кінцевим автоматом, а значить відповіді таких ботів будуть завжди передбачуваними та
точними. Такі боти використовуються для надання послуг певного вузького спектру. Прикладом такого боту є оф
бот Telegram «BotFather» для регістрації власних чатботів. Його робота приведена на рисунку 2.3. Управління
подібними ботами зазвичай здійснюється за допомогою команд та віртуальної клавіатури. Рисунок 2.3 - Ч
BotFather 2.7 Інструментарій для розробки 2.7.1 Мова програмування Java Java – це строго типизована мова заг
призначення, розроблена компанією Sun Microsystems, та пізніше придбана компанією Oracle. Програми Java за
транслюються у спеціальний байт-код, тому вони можуть працювати на будь-якій комп'ютерній архітектурі, для як
реалізація віртуальної Java-машини. Незважаючи на те, що програми, написані на Java, компілюються не у маш

код, а у код, який може виконувати лише віртуальна машина, швидкість таких програм не дуже відрізняєть аналогічних, написаних на мовах, які транслюються у машинний код. Це можливо завдяки JIT компілятору, який тра байткод Java машини у машинний код під час роботи програми. Зараз Java широко використовується для ств backend-у та у Enterprise розробці. Також неможливо не згадати про розробку під Android, де Java була осн мовою для написання додатків, поки на її місто не прийшла мова Kotlin. Однак, попри це, програми для Android во є програмами для віртуальної машини Java, тільки не для стандартної JVM, а для Android Runtime (раніше Da мові Java неможливо явне видалення об'єкта з пам'яті - натомість реалізовано складання сміття. Традиційним при що дає збирачеві сміття «натяк» на необхідність звільнення пам'яті, є привласнення змінної порожнього значення n може виявитися ефективним при необхідності звільнити об'єкт, що не потрібний, посилання на який зберігає довгоживучому об'єкті. Це, однак, не означає, що об'єкт, замінений значенням null, буде неодмінно і негайно вилу але є гарантія, що цей об'єкт буде видалено саме в майбутньому. Даний прийом лише усуває посилання на об'єкт відв'язує покажчик від об'єкта в пам'яті. Об'єкт не буде видалений збирачем сміття, поки на нього вказує хоча посилання зі змінних або об'єктів, що використовуються. Java є об'єктно орієнтованою мовою. Будь-яка функція існувати лише всередині класу. На методи перетворилися й стандартні функції. Наприклад, Java немає функції sin метод Math.sin() класу Math (що містить, крім sin(), методи cos(), exp(), sqrt(), abs() і багато інших). Констр Java не вважаються методами. Деструкторів у Java немає. 2.7.2 Мова розмітки YAML YAML – це мова для структу даних, розроблений у 2001 році Кларком Евансом. Широко використовується як мова для опису конфігураційних для різних програм. Ця мова відрізняється від інших високою читаністю та простотою опису різних структур даних. О елементи YAML [22]: * Потоки YAML використовують друковані Unicode-символи, як UTF-8, так і UTF-16. * Відс прогалин (символи табуляції не допускаються) використовуються для позначення структури. * Коментарі починак символу решітки (#), можуть починатися в будь-якому місці рядка і продовжуються до кінця рядка. * позначаються початковим дефісом (-) з одним членом списку на рядок, або члени списку укладаються у квадратн ([ ]) і поділяються комою та пробілом (,). * Асоціативні масиви представлені двокрапкою з пробілом (:) у вигляді значення», по одній парі ключ-значення на рядок, або у вигляді пар, укладених у фігурні дужки та розділених ко пробілом (,). * Ключ в асоціативному масиві може мати як префікс знак (?), що дозволяє вказати складний наприклад представлений у вигляді списку. * Рядки записуються без лапок, однак можуть бути укладені в одино подвійні лапки. * Всередині подвійних лапок можуть бути використані екрановані символи в C-стилі, що починаю зворотною косою (\). * YAML дозволяє задавати підстановки за допомогою якорів & та псевдонімів (*). У лісти приведений приклад із базових конструкції цієї мови розмітки. Інші елементи та синтаксичні конструкції мови можна у повній специфікації на офіційному сайті [23]. Лістинг 2.1 – Базові елементи мови string: "Строка" numb floating_point: 0.8 map: key: "Значення" key2: 16 string_list: - "Строковий елемент 1" - "Строковий елемент 2" mc - name: "Елемент 1" position: 1 - name: "Елемент 2" position: 2 2.7.3 Збиральник проектів Gradle Програми на компілюються за допомогою компілятора javac. Для середніх та великих програм, які складаються з десятків, або класів, та мають залежність від інших компонентів, ручне збирання за допомогою тільки компілятора дуже складне доцільне. Для рішення цієї проблеми були створені різні збиральники проектів. Збиральник проектів – це програ автоматизує більшість рутинних завдань зі збірки проекту, такі як управляння залежностями, цілями для оптимізація збірки, тощо. Для Java існує чимало збиральників. Їх можна поділити на імперативні та декларати імперативних можна віднести вже майже невикористовуваний збиральник Apache Ant. Він приймає на вхід XML детальним описом задач для збірки. Імперативним він є тому, що розробнику необхідно чітко вказати як потрібно проект. Після Apache Ant мейнстримом став декларативний збірник Maven. Він також використовує XML для проекту, але на відміну від Ant, для базових проектів Maven дозволяє описати лише базові речі, наприклад ім'я п список залежностей, та інші властивості проекту. Для більш складних проектів є безліч плагінів, як можна підключ проекту у тому ж файлі. Але використання XML для опису збірки більш великих проектів приводить до громізд складно читаних файлів. На даний час ще одною з найбільш популярних систем збірки проектів є Gradle, який в більшість проблем, присутніх у попередніх збірниках. Gradle — система автоматичної збірки, побудована на при Ant та Maven, але надає DSL на мовах Groovy або Kotlin замість традиційної XML-подібної форми предста конфігурації проекту. Gradle був розроблений для багатопроектних збірок, що розширюються, і підтримує ка модель розробки, визначаючи, які компоненти дерева збірки не змінилися і які завдання, залежні від цих час вимагають перезапуску. У світі збиральників проектів це називається інкрементальною збіркою. Він також під використання репозиторіїв Maven для підключення залежностей. Писати скрипти збірки з використанням Grac дуже просто та ефективно, навіть якщо для збірки необхідно виконувати нестандартні задачі. 2.7.4 Серед розробки Intellij IDEA У професійній розробці, для підвищення зручності та швидкості роботи з програмним

проектуванням та рефакторингом, застосовуються програмні засоби, які називаються IDE (Integrated Develo Environment). Для мови програмування Java існує чимало IDE різної якості. За даними опиту [24] за 2020 рік, на домінують три Java IDE – Intellij IDEA (62%), Eclipse (20%), та NetBeans (10%). Популярність першої обґрунтован час роботи, IDEA сильно виділяється такими зручними можливостями, як: * Розуміння контексту. IDE розуміє, частина програмного коду редагується у даний час, без необхідності виділяти її. * Розумне авто доповнення. IDE авто доповнення з розумінням контексту, пропонуючи ім’я змінних, функцій та інших частин коду, зважаючи на типи їх ім’я, попередній досвід, та інші фактори. * Language injection. Коли в одному файлі редагується код на різних IDEA розуміє це, та продовжує надавати розумні доповнення та перевірки коду. * Потужний рефакторинг. Дет аналіз коду дає можливість проводити рефакторинг, який буде каскадно змінювати весь вихідний код при необхідн тому числі, якщо він на різних мовах. * Різні мілкі, але дуже зручні особливості поведінки, такі як авто допо синтаксичних конструкцій кавичок, дужок, відступів, автоматичне збереження файлу без натискання Ctrl+S, тощ особливості, багато інтеграцій з іншими сервісами, зручний інструментарій для проектування, та стабільність роблять IDEA майже стандартом для професійної розробки Java в більшості компаній на даний момент. Звичай можливості не безкоштовні для апаратного забезпечення. IDEA потребує значно більше оперативної пам’яті, ніж її ан Це зумовлено детальним аналізом всього проекту для підтримки інструментарію, описаного раніше. Зазвича потребує від 1.5 до 2 ГБ оперативної пам’яті. Але на сучасних комп’ютерах ця вимога більш ніж виконувана. 2.8 використовуваних бібліотек 2.8.1 Guice Guice - це простий DI (Dependency Injection) фреймфорк для програмування Java, розроблений компанією Google [25]. Dependency Injection – це патерн, який ц використовується при написані середніх та великих додатків на різних мовах програмування, в тому числі і на Java патерн значно полегшує управління залежностями, коли програма складається з великої кількості класів, які за один від одного. Guice, як і більшість DI фреймворків для Java, працює за допомогою рефлексії, інструменту Jav надає можливість зчитувати та модифікувати мета інформацію класів, методів, атрибутів та інших частин програми її виконання. При використанні патерну DI, об'єкти приймають свої залежності через конструктор. Тому, щоб ст об'єкт, спочатку потрібно створити його залежність. Однак, щоб створити кожну залежність окремо, потрібно, у свою створити її залежність. Таким чином, насправді нам потрібно створити граф залежностей. Створення графів о вручну дуже трудомістко, та може призвести до помилок і ускладнює тестування. Guice може створити граф об'є розробника. Але спочатку Guice потрібно налаштувати, щоб він створив граф саме так, як потрібно [26]. 2.8.2 Log4J – це бібліотека для логування (запису подій програми у деякий файл), створена організацією Apache. показує, що логування є важливою складовою циклу розробки, бо воно надає точний контекст про запуск та програми. Логінг має свої недоліки. Він може уповільнити роботу програми. Якщо записувати події занадто часто, ц призвести до небажаного переповнення журналу подій. Щоб усунути ці проблеми, log4j розроблено як надійний, шв розширюваний. Оскільки ведення журналів рідко є основним напрямком програми, API log4j прагне бути прост розуміння та використання. API для Log4j відокремлений від реалізації [27], що дає зрозуміти розробникам дода класи та методи вони можуть використовувати, забезпечуючи сумісність наперед. Це дозволяє розробникам покращити впровадження безпечно та сумісним чином. Основним об’єктом для логування є Logger. Він має свої для запису подій у журнал. Кожна подія має свій рівень. Рівень події визначає її вивимість у журналу або у термін певних налаштуваннях. Бібліотека має наступні стандартні рівні: DEBUG, INFO, WARN, ERROR, FATAL. Програ використовують API Log4j2, запитуватимуть об’єкт Logger із певним іменем у LogManager. LogManager відповідний LoggerContext, а потім отримає з нього об’єкт Logger. Якщо Logger необхідно створити, він буде пов’ з LoggerConfig, який містить те саме ім’я, що й Logger, або ім’я батьківського пакета, або кореневий LoggerC Об’єкти LoggerConfig створюються з декларацій Logger у конфігурації. LoggerConfig пов'язаний з додатка фактично доставляють LogEvents. Приклад отримання об’єкту логеру приведений у лістингу 2.2. Лістинг 2.2 – П створення об’єкту Logger. Logger rootLog = LogManager.getRootLogger(); Logger namedLo LogManager.getLogger("wombat"); Logger classLog = LogManager.getLogger(SomeClass.class); Після істо знайдені вразливості цієї бібліотеки, рекомендується використовувати лише останню версію, де ці вразл вважаються виправленими. 2.8.3 Hibernate Hibernate — бібліотека для мови програмування Java, призначе вирішення задач об’єктно-реляційного відображення (ORM). Є самою популярною реалізацією специфікації JPA Persistence API). Бібліотека не тільки вирішує задачу зв'язку класів Java з таблицями бази даних (типами даних типами даних SQL), а також надає засоби для автоматичного генерування та оновлення набору таблиць, побудови з та обробки отриманих даних. Бібліотека дозволяє значно зменшити час розробки, яке зазвичай витрачається на описування коду SQL та JDBC. Hibernate забезпечує використання мови, схожої на SQL, названої Hibernate Language (HQL), яка дозволяє виконувати SQL-подібні запити, записані поруч з об’єктами даних Hibernate.

критеріїв надаються як об'єктно-орієнтована альтернатива HQL. Відображення Java класів із таблицями баз здійснюється за допомогою Java-анотацій, або конфігураційних XML-файлів (застарілий метод). Використовуючи XML Hibernate, можна створити скелет вихідного коду для класу тривалого зберігання. В цьому немає необхідност використовується анотація. Hibernate може використовувати файл XML або анотації для підтримки схем баз дани 2.8.4 Apache POI Apache POI – це проект [29], який надає бібліотеки Java для читання та запису файлів у фо Microsoft Office, таких як Word, PowerPoint та Excel. Бібліотека для роботи з форматами Excel надає єдиний інте для різних форматів, таких як XLS та XLSX. Apache POI це не одна бібліотека. Вона складається з наступних компо які ожна використовувати разом або окремо: * HSSF (Horrible Spreadsheet Format). Компонент читання та файлів MS-Excel, формат XLS. * XSSF (XML Spreadsheet Format). Компонент читання та запису файлів MS формат XLSX. * HPSF (Horrible Property Set Format). Компонент отримання наборів властивостей файлів MS-O HWPF (Horrible Word Processor Format). Компонент для читання та запису файлів MS-Word, формат DOC. * (XML Word Processor Format). Компонент читання та запису файлів MS-Word, формат DOCX. * HSLF (Horribl Layout Format). Компонент для читання та запису файлів PowerPoint, формат PPT. * XSLF (XML Slide Layout Fo Компонент для читання та запису файлів PowerPoint, формат PPTX. * HDGF Horrible DiaGram Format Компонент з файлами MS-Visio, формат VSD. * XDGF XML DiaGram Format Компонент роботи з файлами MS-Visio, формат Apache POI надає вам такі інтерфейси, як Workbook, Sheet, Row, Cell, тощо. Ці інтерфейси реалізовані конкр класами, такими як HSSFWorkbook, HSSFSheet, HSSFRow, HSSFCell. Класи, які реалізують роботу з новим формато мають схожу назву, але з префікс XSSF. 2.8.5 TelegramBots Telegram API побудоване на протоколі HTTP. Взає цим API проходить через посилання HTTP запитів на певний адрес з тілом у форматі JSON. Для того, щоб спрости процес, та зменшити кількість помилок, було створено багато бібліотек для різних мов програмування, які на абстракції для роботи з API не напряму, а через певний інтерфейс. Для мови програмування Java така бібліотека існує. Вона називається TelegramBots. Бібліотека TelegramBots створена для розробки чат ботів для месен Telegram. Вона підтримує 2 методи отримання оновлень (запитів) від користувачів: 1. Webhooks. Отримання ми оновлень на власний сервер за допомогою веб хуків. Для цього обов'язково треба мати HTTP сервер та сертифік нього, для роботи через протокол HTTPS. 2. Long Polling. Отримання оновлень методом періодичних запитів до се Не потребує розгортання HTTP серверу, але отримання оновлень не миттєве та більш затратно через постійне від та закриття з'єднання. Бібліотека підтримує останні версії Telegram API та дозволяє працювати з повідомле прикріпленими файлами, аудіо та відео, отримувати інформацію про користувача, який розмовляє з ботом, тощ Висновки розділу Було здійснено огляд різних методів та засобів розробки, таких як парсинг, рендеринг, регулярні в Здійснено огляд двох форматів таблиць Excel та їх внутрішню структуру. Обґрунтовано вибір мови програмува бібліотек для розробки програми. Для розробки використовується мова програмування Java, з використанням бібліотек як TelegramBots, Apache POI, Hibernate, Log4j, Guice. В якості збиральника проекту було вибрано G Здійснено огляд цих бібліотек та виявлено особливості роботи з ними. Здійснено огляд середовища розробки Intel Community Edition, яка була вибрана для розробки програмного продукту. Для створення конфігурації програм обрано мову розмітки YAML. РОЗДІЛ 3 РОЗРОБКА ТА ІНТЕГРАЦІЯ ПРОГРАМНОГО ПРОДУКТУ 3.1 Проектуван Загальна архітектура До архітектури чат боту були поставлені наступні вимоги: 1. Масштабованість. 2. Розд обов'язків. 3. Незалежність окремих частин (модулей). Програма буде складатись з окремих модулей, які взаємодіє собою, та мають внутрішні компоненти. Всього можна виділити 5 модуля: 1. Модуль даних. 2. Завантажувач розкл Рендер розкладу. 4. Ядро боту. 5. Веб сервер. Модуль даних відповідає за доступ до даних користувачів. Він до іншим модулям абстрагуватись від способу збереження даних та має зв'язок з базою даних певного типу. Для збере даних буде використовуватись реляційна база даних. Важливо враховувати, що тип бази даних може бути змін часом, тому тісного зв'язку з певною СУБД не повинно бути. Для цього модуль даних можна поділити на декілька На першому рівні буде робота з чистими SQL запитами. Для цього в Java для існує стандарт JDBC. На другом знаходиться ORM фреймфорк Hibernate. Він дозволить абстрагуватись від певного типу БД, бо має власну мову, с SQL. На верхньому рівні розташовані DAO класи. Саме з цими класами буде працювати інші модулі та сервіси. Та дозволять абстрагувати роботу з даними та зробить модуль даних більш гнучким. Завантажувач розкладу відпов завантаження розкладу з репозиторію, та його парсинг. Сбір даних з таблиць розкладу проводиться за допо бібліотеки Apache POI, яка описувалась в 2 розділі. Рендер розкладу відповідає за представлення розкладу таблиці, яку бачить кінцевий користувач. Він залежить від певного типу рендеру. За умовченням для ренд використовується безкоштовна версія бібліотеки Aspose Cells. Ядро боту відповідає за взаємодію з кін користувачем через Telegram API. Воно включає такі компоненти як командний інтерфейс, граф команд представляє собою машину станів, та таймер, який взаємодіє з менеджером розкладу та відправляє оновлений р

кінцевому користувачу. Через обмеження платформи Telegram, яка дозволяє відправляти лише 30 повідом
секунду, модуль має мати інтерфейс відправки повідомлень який буде складувати їх у чергу, для відправки, врах
це обмеження. Веб сервер відповідає за обробку запитів з панелі керування. Він має бути побудований за шаблоно
та працювати на протоколі HTTP/1.1. Для обробки запитів використовуються контролери, які посилають команди на
менеджеру розкладу, якщо запит стосується зміни у налаштуваннях. Ці модулі не взаємодіють між собою безпосер
а використовують для цього окремі сервіси. Кожен сервіс відповідає за окрему групу функцій. Так, можна виді
сервіса: 1. Менеджер підписок. Відповідає за доступ до підписок користувачів на розклад. «Розмовляє» з модулем
так як треба отримувати та змінювати дані про підписки на розклад. 2. Менеджер розкладу. Відповідає за дос
розкладу, який був розпарсений модулем Schedule Loader. Важливо враховувати, що бот повинен отримувати да
запити від інших, віддалених сервісів. В архітектурі позначимо їх як зовнішні модулі. До таких модулей віднося
Віддалений репозиторій з розкладом занять. 2. Сервер Telegram. Віддає дані про запити користувачів під час «ро
з ботом. 3. Панель керування. Є клієнтом, в не сервером, але важливо враховувати як ще одну точку входу. На рису
візуально представлена архітектура зі всіма модулями, які були виділені раніше. Рисунок 3.1 – Архітектура чат бо
База даних Для зберігання даних використовується реляційна модель. На рисунку 3.2 позначена структура таблиц
даних бота та їх відносини. Рисунок 3.2 – Структура таблиць БД Таблиця «users» представляє користувача чат бот
зберігається ідентифікатор користувача месенджеру, деякі його персональні дані, та налаштування. Так як
користувач може бути підписаний лише один раз на певний тип розкладу, зв'язок з іншими таблицями представле
«один до одного». Таблиця «subs_teachers» представляє підписку на розклад викладача. Вона має зовнішній кл
зв'язку з таблицею користувачів. Колонки «first_name» та «last_name» це не дублювання даних з таблиці корист
а дані викладача, на розклад якого оформлена підписка. Таблиця «subs_courses» представляє підписку на р
курсу. Таблиця «subs_groups» представляє підписку на розклад певної групи. Таблиця «subs_points» предс
підписку на власні оцінки. На відміну від інших підписок, надсилання інформації про оновлення оцінок не треба, том
«notified» відсутнє. Таблиця «hashes» зберігає хеш суми файлів розкладу для порівняння їх з актуальними даним
пошуку оновлених файлів. Хеш сума представлена у виді строки, тому і поле для її зберігання відповідного типу. Та
«api_users» необхідна для зберігання даних про користувачів панелі керування. Для безпеки даних, існує
користувачів – звичайні та адміністратори. Кожен користувач може мати багато сесій в один час, тому має зв
таблицею «api_sessions» у виді «один до багатьох». Таблиця «api_sessions» зберігає поточні сесії користувачів
керування. Кожна сесія має унікальний токен доступу, який є первинним ключом у таблиці. Тому знаючи токен, який
користувач при кожному запиті, можна отримати інформацію про користувача та його права доступу. Поле «expiry_
зберігає інформацію про час закінчення дії сесії. Це число, яке представляє час у форматі «Unix». 3.1.3 REST API
керування складається з двох частин – бекенду та фронтенду. Бекенд для панелі проектується за принципами RES
API буде працювати поверх протоколу HTTP з використанням повідомлень у форматі JSON. Кожен запит в
здійснюється на певний адрес. Для розділення різних типів запитів використовуються так звані точки прийому
(endpoints). По суті це URL, на який здійснюється запит. Зазвичай для описування кінцевої точки, базови
ігнорується, бо він спільний для всіх точок. Для панелі керування можна виділити наступні точки прийому запиту: *
– здійснити аутентифікацію та отримати токен нової сесії. * /logout – видалити власну сесію. * /stats – отрима
про статистику користування. * /properties – отримати або змінити налаштування базові налаштування. * /tec
отримати або змінити налаштування розкладу викладачів. * /consult - отримати або змінити налаштування ро
консультацій. * /courses - отримати або змінити налаштування розкладу курсів. * /rendering - отримати або з
налаштування рендерингу. * /user/list – отримати список акаунтів корситувачів панелі. * /user/sessions – от
список активних сесій. * /user/endSession – завершити сесію. * /user/create – створити нового користу
/user/edit – змінити заді певного користувача. * /user/delete – видалити користувача панелі. Всі запит
аутентифікації, повинні мати токен доступу в заголовку. Цей токен необхідний для авторизації користувач
авторизації в заголовку повинен бути атрибут Authorization: , де «access_token» - це токен доступу, який
отримати після запиту до точки «/login». Якщо сесія за даним токеном не знайдена, клієнт отримає відповідь зі ст
401 Unauthorized. На будь-який запит може прийти помилка замість відповіді. Повідомлення помилки має такий ф
{ "error": "", "message": "" } Де «error» - код помилки у строковому форматі, а «message» - більш докладни
помилки. Для вказаних раніше точок можна виділити наступні види помилок: * invalid_login – невірний ло
аутентифікації. * invalid_password – невірний пароль при аутентифікації. * unefined_message – нем
прочитати повідомлення через нестачу даних. * invalid_token – невірний токен при завершенні сесії. * login_to
користувач з вказаним логіном вже існує. * password_short – пароль надто короткий. * user_not_found – корис
вказаним логіном не знайдений. Повну специфікація REST API з описом та структурою всіх повідомлень можна зна

посиланням [30] в списку посилань. 3.1.4 Модель взаємодії з ботом В якості моделі для взаємодії з ботом була ви модель кінцевого автомату. Кожен кінцевий автомат може бути представлений як граф. Кожен стан, який може пр бот під час «розмови» - це вершина графу, а команди, які посилає користувач, тобто переходи між станами – це графу. На рисунку 3.3 зображений графік цих станів та переходів між ними. Переходи, які починаються з симво (слеш) - це команди. Ланцюг станів завжди починається з команди. До деяких станів можуть вести одразу де переходів, наприклад для виведення списку викладачів користувач може ввести одну з 4 команд, але від типу ко вирішується який тип розкладу вивести. Кожен стан супроводжується певною дією з боку бота. Часто це вивід інфо про поточний стан, або відповідь на певний запит. Кожен кінцевий стан має перехід на початок (стан «Старт»). На г це не позначено щоб уникнути графічної плутанини з переходами. Рисунок 3.3 – Граф станів та переходів між ним Структура проекту Структуру проекту певною мірою диктують система збірки. Система збірки «Gradle», яка описа розділі, дозволяє розбивати проект на модулі. Всього можна виділити 3 модуля: 1. Модуль «api» зі спільними клас утилітами. 2. Модуль «parser», представлення сутностей розкладу та його парсингу. 3. Головний модуль «bot», як точку входу в програму, реалізацію інтерфейсів модуля «api», та логіку взаємодії з Telegram API. Кожен мо одиницею компіляції, тому це також дозволить швидше збирати проект, завдяки інкрементальній збірці. Ра необхідними для системи збірки файлами та директоріями, проект буде мати структуру, яка позначена на рисун Рисунок 3.4 – Структура проекту 3.1.6 Проектування конфігурації програми Конфігурація програми складається з 3 (файлів): 1. Головна конфігурація. 2. Конфігурація розкладу. 3. Конфігурація мови. Головна конфігурація відпові базове налаштування програми, та містить наступні елементи: 1. Налаштування підключення до бази да Налаштування підключення до Telegram API. 3. Налаштування регулярних виразів. 4. Налаштування підключе серверу оцінок успіху. 5. Налаштування API панелі керування. У лістингу 3.1 приведений приклад головної конф бота. Лістинг 3.1 – Головна конфігурація бота database: 'hibernate.dialect': "org.hibernate.dialect.H2D 'hibernate.connection.driver_class': "org.h2.Driver" 'hibernate.connection.url': "jdbc:h2:./data 'hibernate.connection.username': "user" 'hibernate.connection.password': "user" 'hibernate.show_sql': 'hibernate.hbm2ddl.auto': "update" 'hibernate.jdbc.batch_size': "50" telegram: bot_name: "VasyaSchedu token: "TOKEN" regex: teacher_default: "([А-ЯЁЇІЄҐ][а-яёїієґ]{0,32})\\s*([А-ЯЁЇІЄҐ])\\.\\s*([А-ЯЁЇІЄҐ teacher_inline: "([А-ЯЁЇІЄҐ][а-яёїієґ]{1,32}\\s*[А-ЯЁЇІЄҐ]\\.\\s*[А-ЯЁЇІЄҐ]\\.*)\\s*(ауд\\.{1,3})" classroo яёїієґ]*\\.*\\s*([0-9]{3})" points: url: "http://77.93.42.90:81/uspeh/index.php" login_ "action=login&lastname=${lastname}&n1=${n1}&n2=${n2}&password=${password}&submit=%C2%EE%E9% login_success: 302 timeout: 3 thread_pool_size: 6 Блок «database» налаштовує підключення до БД та по фреймворку Hibernate, та є словарем де ключ це ім'я властивості конфігурації Hibernate, а ключ – значен властивості. Така конфігурація дозволяє дуже гнучко змінювати поведінку взаємодії з БД, або навіть тип цієї БД «telegram» має поля для налаштування підключення до Telegram API. Блок «regex» дозволяє налаштувати ре вирази для парсингу деяких частин розкладу. Ці вирази виділені в конфігурацію через можливу зміну або модифіка частин у майбутньому. Блок «points» має поля для налаштування підключення до серверу з результатами студентів. Поле «thread_pool_size» визначає кількість потоків для паралельної обробки запитів до бота від корист Конфігурація розкладу відповідає за налаштування мета інформації про різні типи розкладу та його рендер структура приведена у лістингу 3.2. Лістинг 3.2 – Конфігурація розкладу check_rate: 20 day_indexes: "Понеді "Вівторок": 1 "Середа": 2 "Четвер": 3 "П'ятниця": 4 "Субота": 5 "Неділя": 6 teachers: url: "https://.../Teach associations: 'і': "1" 'К9': "1к" 'Кол4': "4к" 'і(з)': "zo:1 курс" consult: url: "https://.../consultation.xls" day_point: col: 1 teacher_point: col: 0 row: 3 courses: - url: "https://.../1.xls" name: "Інститут 1 курс" day_point: col: 0 group_point: col: 4 row: 5 render: format: JPEG dpi: 175 Поле «check_rate» виставляє період перевірки розк секундах. Блок «day_indexes» налаштовує зіставлення імені дня до його індексу. Індекс першого дня є нулем «teachers», «consult» та «courses» налаштовують доступ до розкладу відповідного типу. Блок «teachers» тако налаштування асоціацій скорочень, які можуть використовуватись у розкладі викладачів, з певним розкладом курс «render» налаштовує рендеринг розкладу, у саме формат вихідного зображення та його якість (кількість точок на Конфігурація мови має найбільш просту структуру. Це файл з великою кількістю рядків у форматі ключ-значен ключом є деяке унікальне ім'я строки, а значенням сама строка. Доступ до цих строк у коді здійснюється саме унікальне ім'я. Це дозволить змінювати мову або окремі строки у програмі без її перекомпіляції. 3.2 Програмуванн Зчитування конфігурації додатку Для зручної роботи з конфігурацією програми, необхідно провести г конфігураційних файлів у рідні для мови програмування структури. Для цього треба створити клас для кожно конфігурації. На рисунку 3.5 приведена структура класів конфігурації. Рисунок 3.5 – Структура класів конфігураці AbstractConfig є базою для інших класів конфігурації. Він має посилання на дерево конфігурації, з якого інш

можуть зчитувати або записувати дані. Також він має абстрактний метод load, який визивається під час к перезавантаження конфігурації. Для зручності роботи з деякими даними, він має метод «loadProperties» перетворює вузел конфігурації на структуру и вигляді ключ-значення. У Java така структура називається Prop Об'єкт цього класу приймає наприклад фреймфорк Hibernate для внутрішнього налаштування, тому цей мето корисним у майбутньому. У лістингу 3.3 приведене тіло цього методу. Лістинг 3.3 – Код методу loadProperties pro Properties loadProperties(ConfigurationNode node) { Properties properties = new Properties(); for (var node.getChildrenMap().entrySet()) { String key = entry.getKey().toString(); String valu entry.getValue().getString(); properties.put(key, value); } return properties; } Клас MainConfig, як слідує з представляє головну частину конфігурації програми. Він наслідується від AbstractConfig та реалізує його метод l тілі цього метода відбувається зчитування даних з конфігурації за їх шляхом (шляхом є послідовність ключів, яка в певного поля), та їх запис у відповідне поле об'єкту. Надалі досуп до даних конфігурації здійснюється через ці точніше через getter-методи для цих полей. Клас ScheduleConfig працює так само, як головний клас конфігура зчитує інший файл. Слід зауважити, що цей клас працює з більш складними конструкціями, власними типами дани для кожного типу, який зчитується з конфігурації, заздалегідь створено так званий клас-десеріалізатор. Це кла відповідає за побудову більш складного об'єкту з примітивів, які були зчитані з файлу конфігурації. 3.2.1 Програм типів розкладу Перш за все, необхідно створити систему ідентифікації розкладу. За цим ідентифіктором можн однозначно ідентифікувати таблицю розкладу. Кожна таблиця розкладу знаходиться у певному файлі та ар унікальним ім'ям. Тому для ідентифікації достатньо знати ім'я файлу розкладу та ім'я аркуша. Слід зауважити, що п та наявність деяких аркушів з розкладом не є гарантованою, тому порядковий номер аркуша не підійде для іденти розкладу. Тому ідентифікатор розкладу буде мати строковий тип, та записуватись у форматі <ім'я файлу>:<ім'я ар де ім'я файлу є так званим простором імен, бо може мати більше ніж один ключ, а ім'я аркушу є ключем дост розкладу. Також, важливо щоб і простір імен і ключ були завжди в одному регістрі, щоб запобігти помилок порівнянні. Для зберігання такого ідентифікатора був створений клас NamespacedKey. Його структура привед рисунку 3.6. Рисунок 3.6 – Клас ідентифікатора розкладу Перш за все клас перевизначає методи equal hashCode() для правильного порівняння об'єктів цього класу при пошуку. Для перетворення об'єкту класу на клас перевизначає метод toString(). Його тіло приведено у лістингу 3.4. Лістинг 3.4 – Метод toString() public toString() { return key.isEmpty() ? namespace : String.format("%s:%s", namespace, key); } Для створення класу зі строки, був створений статичний метод parse(), який приймає певну строку та повертає екзе ідентифікатора. Тіло метода приведено у лістингу 3.5. Лістинг 3.5 – Метод parse() public static Namespac parse(String str) { Preconditions.checkNotNull(str, "Raw NamespaceKey cannot be null"); String[] arr = str.sp if (arr.length == 1) return of(arr[0]); if (arr.length != 2) throw new IllegalArgumentException("Invalid names key: " + str); return of(arr[0], arr[1]); } Слідуючи об'єктно орієнтованому підходу, відповідальність за збереже роботу з інформацією про розклад буде розділена по класам. Інформацію про розклад можна розділити на два Інформація, яка відома заздалегідь. Це метаінформація розкладу. До неї відносяться посилання на розклад, та ключових точок. * Інформація, яка отримується після парсингу. До цієї інформації відносяться такі дані, як спис викладачів, або інших даних, відповідних певному типу розкладу. Відповідно до типів розкладу, можна виділити на класи метаінформації розкладу: * TeacherScheduleInfo – метаінформація розкладу викладачів. * CourseSchedul метаінформація розкладу курсу. * ConsultScheuleInfo - метаінформація розкладу консультацій. Для слід принципам DRY, для цих класів був написаний базовий (абстрактний) клас. Структура цього класу позначена на р 3.7. Рисунок 3.7 – Структура класу AbstarctScheduleInfo Клас має два поля – «id» та «url». Ідентифікатор ма унікальним для кожного розкладу. За замовченням ідентифікатором є ім'я файлу розкладу. Структура всіх метаінформації про розклад мають структуру, яка позначена на рисунку 3.8. Рисунок 3.8 – Структура метаінформації Клас TeacherScheduleInfo має поле «associations» типу Map, в якому зберігається інформа відношення скорочення у таблиці зайнятості до певного ідентифікатору розкладу. За цим ідентифікатором можн знайти потрібний розпарсений розклад. Клас CourseScheduleInfo має поля для ключових точок парсингу розкладу, яке буде відображатись користувачу. Клас ConsultSchedule має поля для ключових точок парсингу фай додаткових полей та методів. Наступним кроком є створення класів самого розкладу. Відповідно до типів розкладу, виділити наступні класи: * TeacherSchedule – розклад викладачів. * CourseSchedule – розклад ку ConsultSchedule – розклад консультацій. * ClassroomSchedule – розклад зайнятості аудиторії. Ці класи збе відповідають за інформацію про розпарсений розклад. Як і класи метаінформації, вони наслідують від базового AbstractSchedule. Структура базового класу розкладу позначена на рисунку 3.9. Рисунок 3.9 – Базовий клас ро Клас має посилання на певний клас-рендер, зображення всього документу, яке формується після парсингу, та унік

ідентифікатор розкладу. Для рендерингу певної частини розкладу (частіше всього персональний розклад корист створений абстрактний метод getPersonalRenderer(), який повертає екземпляр класу, який реалізує інте ScheduleRenderer. Стурктура класів для рендерингу описана у наступній частині підрозділу. Всі раніше описан розкладу наслідують від базового класу. Тож їх структура виглядає так, як показано на рисунку 3.10. Детально коже буде розглядатись далі. Рисунок 3.10 – Структура класів розкладу Клас TeacherSchedule представляє р викладачей (всіх разом). Він має структуру, позначену на рисунку 3.11. Клас зберігає інформацію про розклад к викладача у мапі (клас реалізуючий інтерфейс Map), де ключом є деякий клас Person, який буде описаний значенням є список днів викладача. Рисунок 3.11 – Клас TeacherSchedule Клас Person, який використовується у ба класах, в тому числі у розкладі викладачів, представляє певного учасника системи, людину. Його об'єкт зберіга прізвище та по-батькові людини, та дозволяє безпечно порівнювати їх, в тому числі проводити приблизне порів Структура цього класу позначена на рисунку 3.12. Рисунок 3.12 – Клас Person Більше всього цікавий метод isSimila приймає екземпляр цього ж класу, та повертає булево значення, яке говорить чи схожі порівняні персони. Степінь сх розраховується за допомогою алгоритму «Відстань Левенштейну», використовуючи прізвище персони. Тіло цього можна побачити у лістингу 3.7. Лістинг 3.7 – Метод isSimilar() класу Person public boolean isSimilar(Person ano if (isEmpty()) return false; int ln = Levenshtein.calcDistance(this.lastName(), another.lastName()); return MAX_SIMILARITY && this.firstName().equals(another.firstName()) this.patronymic().equals(another.patronymic()); } Бачимо, що результатом порівняння строк за Левенштейном є число, яке означає кількість модифікацій однієї строки, для повної відповідності другої. Максимальна кількість моди позначена у константі MAX_SIMILARITY, та має значення 1 (один). Порівняння персон за схожістю необхідно для викладачів, прізвища яких мають різне написання у різних таблицях розкладу. 3.2.1 Програмування парсеру табли кожного типу розкладу необхідний власний клас з реалізацією парсингу цього розкладу. Відповідно до типів ро можна виділити наступні класи парсерів: * TeacherScheduleLoader – парсер розкладу викладач CourseScheduleLoader – парсер розкладу курсів. * ConsultScheduleLoader – парсер розкладу консультацій. Буд парсер має спільну логіку, яку можна винести у абстрактний клас, від якого будуть наслідувати всі інші. Структура класу показана на рисунку 3.13. Рисунок 3.13 – Клас AbstractScheduleLoader Метод loadWorkbook() зава документ Excel за посиланням, яке вказано у метаінформації розкладу. Він має тіло, яке приведено у лістингу 3.8. 3.8 – Метод loadWorkbook() protected Workbook loadWorkbook(ScheduleInfo info) ScheduleParseException { try (InputStream in = info.getUrl().openStream()) { String filenam info.getUrl().toString(); Workbook workbook = null; if (filename.endsWith(".xlsx")) { workbook = XSSFWorkbook(in); } else if (filename.endsWith(".xls")) { workbook = new HSSFWorkbook(in); } return workb catch (IOException e) { throw new ScheduleParseException("Cannot load " + info.getUrl().toString(), e); } } бібліотека Apache POI не має механізму для виявлення формату документу, це робиться вручну, за допомогою пер розширення файлу. Класи-парсери, які наслідують від базового, мають структуру, яка позначена на рисунку 3.14. Р 3.14 – Структура класів-парсерів Парсинг розкладу виконується за допомогою певних вказівників на осередки. П розкладу викладачів починається з пошуку першого осередка з ім'ям викладача та створенням вказівника нього. можливий завдяки координатам, які заздалегідь вписані у файл конфігурації програми. Кожен раз, коли вказів осередок викладача переміщується вниз, робиться ітерація по всім осередкам, які мають інформацію про пари. відповідає реалізація метода load(), який визначений у базовому класі парсера. Частина цього метода приве лістингу 3.9. Лістинг 3.9 – Обхід таблиці викладачів Map> teachers = new HashMap<>(); int r POINT_TEACHER.row(); Cell teacherCell = getCell(sheet, row, POINT_TEACHER.col()); (!ExcelUtil.isEmptyCell(teacherCell)) { Person person = Person.teacher(ExcelUtil.getCellValue(teacherCell) days = getDays(sheet, row); teachers.put(person, days); row++; teacherCell = getCell(sheet, POINT_TEACHER.col()); } За обхід днів та пар розкладу викладача відповідає метод getDays(). Він має цик інкрементує позицію вказівника на пару після запису даних попередньої пари. 3.2.3 Програмування рендеру ро Рендеринг розкладу може проводитись різними методами. Для уніфікації цього процесу був створений абстрактни SheetRenderer. Він має методи, які повинен реалізувати кожен backend клас для рендерингу листа. В якості бекен рендерингу на даний час використовується бібліотека Aspose Cells, яка має відповідний функціонал. Для тестува експериментів бус також створений власний клас для рендерингу за допомогою вбудованого в стандартну біб Java API для малювання AWT. Структура всіх класів рендерингу позначена на рисунку 3.15. Рисунок 3.15 – Стр класів для рендерингу За допомогою єдиного API рендерингу, всі класи для малювання таблицы розкладу працювати з одним інтерфейсом не зважаючи на те, яким методом він буде малюватись. Рендеринг розкладу почин з будування його моделі. Модель будується на основі отриманих за допомогою парсингу даних, які зберігаються

об'єктах, класи яких були описані раніше, це TeacherSchedule, CourseSchedule та інші. Відповідно до клас об'єктів можна виділити наступні класи для рендерингу: * TeacherScheduleRenderer – відповідає за будівниц рендеринг персонального розкладу викладача. * CourseScheduleRenderer – відповідає за будівництво та рен розкладу курсу. * GroupScheduleRenderer – відповідає за будівництво та рендеринг розкладу гр ConsultScheduleRenderer – відповідає за будівництво та рендеринг персонального розкладу консульта ClassroomScheduleRenderer – відповідає за будівництво та рендеринг розкладу зайнятості певної аудиторії. Т загальна логіка та дані для всіх цих класів, можна виділити абстрактний клас. Його структура позначена на рисун Рисунок 3.16 – Клас AbstractScheduleRenderer Клас має методи для ініціалізації шрифтів, та допоміжні мето будуть часто використовуватись при рендерингу розкладу. Рендеринг розкладу викладача починається зі ств шапки. Шапка має статичні та динамічні дані. За створення шапки відповідає метод drawHeader(). Якщо убрати к оформлення, код заповнення осередків шапки даними приведений у лістингу 3.10. Лістинг 3.10 – Створення розкладу викладача Cell titleCell = getOrCreateCell(sheet, 0, 0); Cell nameCell = getOrCreateCell(sheet, 1, 0 dayTitleCell = getOrCreateCell(sheet, 2, 0); Cell classNumTitleCell = getOrCreateCell(sheet, 2, 1) classTimeTitleCell = getOrCreateCell(sheet, 2, 2); Cell classesTitleCell = getOrCreateCell(sheet, titleCell.setCellValue(schedule.getTitle()); nameCell.setCellValue(person.toStr dayTitleCell.setCellValue(lang.of("schedule.render.head.days")); classNumTitleCell.setCellValue(lang.of("schedule.render.head.classnum")); classTimeTitleCell.setCellValue(lang.of("schedule.render.head.classtime")); classesTitleCell.setCellValue(lang.of("schedule.render.head.classes")); Рендеринг тіла таблиці робиться методами – drawDay() і drawClass(). Метод drawDay() має цикл для ітерації по дням тижня певного викладача. drawClass() відповідає за малювання осередку певної пари. Для цього на основі скорочення та асоціац налаштовані для розкладу викладачів, робиться пошук по даним з розкладу курсу, на який є посилання у т зайнятості. Для більш розумного пошуку є три допоміжні методи: getScheduleByCourses(), findClasse getAlternateClasses(). Метод getScheduleByCourses() приймає дані про пару з таблиці зайнятості, та по колекцію з розкладу курсів, на які посилається таблиця зайнятості. Його тіло приведено у лістингу 3.11. Лістинг 3.11 – getScheduleByCourses() private Collection getScheduleByCourses(TeacherClass teacherClass) { List schec new LinkedList<>(); for (String course : teacherClass.courses()) { NamespacedKey scheduleK this.schedule.getInfo().getAssociation(course); if (scheduleKey == null) // Try to find schedule by raw "p scheduleKey = this.schedule.getInfo().getAssociation(teacherClass.raw()); if (scheduleKey != nu CourseSchedule sch = (CourseSchedule) manager.getCourseSchedule(scheduleKey); if (sch != schedules.add(sch); } } return schedules; } Пошук за «сирим» вказівником був добавлений через неоднозн формату запису посилань на розклад курсів. Так, зазвичай, курси відділені комою, але іноді посилання може мат тому стандартна операція split() над строкою може віддати невірний результат. Пошук за «сирим» посиланням повернути правильний результат, при умові, що для цього посилання існує асоціація. Метод findClasses() пр екземпляр розкладу курсу, дані викладача та номер пари, та повертає всі пари, які призначені на запрошу викладача. Ці пари і виводяться у таблиці персонального розкладу, замість скорочень. Тіло методу приведене у л 3.13. Лістинг 3.12 – Метод findClasses() private Collection findClasses(CourseSchedule schedule, TeacherDa int classNum, Person teacher, boolean retAlternate) { int dayIndex = TimeTable.getDayIndex(day.getNa Optional dayOpt = schedule.getDay(dayIndex); if (dayOpt.isPresent()) { Collection class dayOpt.get().getClasses(classNum, person); if (!classes.isEmpty()) return classes; } return retAlterr getAlternateClasses(schedule, day, classNum, teacher) : Collections.emptyList(); } Якщо ні одна пара за посил не була знайдена, визивається метод getAlternateClasses(). Він приймає ті ж аргументи, що і метод findClasses робить пошук за всіма таблицями розкладу у файлі. Деякі таблиці розкладу, наприклад розклад заочного відд знаходиться в одному файлі в різних листах. У разі відсутності хоч одної пари викладача, робиться припущен інформація про пару може бути у інших листах. Цей пошук є найбільш затратним, але і визивається він не част методу getAlternateClasses() приведене у лістингу 3.13. Лістинг 3.13 – Метод getAlternateClasses() private Coll getAlternateClasses(CourseSchedule schedule, TeacherDay day, int classNum, Person teacher) (CourseSchedule member : schedule.getFileGroup()) { Collection classes = findClasses(member, day, clas teacher, false); if (!classes.isEmpty()) return classes; } return Collections.emptyList(); } Наприкінці створення т розкладу, якщо ні одна пара не була знайдена, на її місці виводиться посилання з таблиці зайнятості, для ручного викладачем. Після того, як модель розкладу була побудована, вона відправляється на фінальний рендеринг у ра зображення. Побудова та рендеринг розкладу інших категорій майже нічим не відрізняється від побудови ро

викладача, за винятком відсутності системи пошуку пар. 3.2.4 Налаштування Guice DI Для спрощення дост
залежностей (об'єктів) використовується техніка Dependency Injection за допомогою бібліотеки Guice, яка
описана у другому розділі. Для організації джерел залежностей в Guice треба створити модулі. Кожен моду
перевизначити метод configure(), в якому . Для поточного проекту було створено модулі BaseM
PersistenceModule, ServicesModule, WebModule. Модуль BaseModule визначає базові залежності, такі як конфі
програми. Його код приведений у лістингу 3.14. Лістинг 3.14 – Метод configure() модуля BaseModule protecte
configure() { bind(Path.class) .annotatedWith(Names.named("appDir")) .toInstance(ro
bind(MainConfig.class).toInstance(conf); bind(ScheduleConfig.class).toInstance(schedule
bind(Language.class).toInstance(Language.builder() .source(ConfigSources.resource("/lang.yml",
.copyTo(rootDir)) .build()); } Модуль PersistenceModule визначає залежності, які відносяться до бази даних. До
залежностей выдноситься фабрика сесій, Dao класи, тощо. Його код приведений у лістингу 3.15. Лістинг 3.15 –
configure() модуля PersistenceModule protected void configure() { SessionFactory sessionFacto
initHibernate(); bind(SessionFactory.class).toInstance(sessionFactory); bind(TeacherSubsDao.
bind(ConsultSubsDao.class); bind(CourseSubsDao.class); bind(GroupSubsDao.class); bind(PointsSubsDao.
bind(NoticesDao.class); bind(HashesDao.class); bind(ApiUserDao.class); bind(ApiSessionDao.class); }
initHibernate() ініціалізує фабрику сесій Hibernate. Його тіло приведено у лістингу 3.16. Лістинг 3.16 – Тіло
initHibernate() Configuration configuration = new Configura
configuration.addProperties(conf.getDbProperties()); configuration.addAnnotatedClass(SubscriptionPoints.
configuration.addAnnotatedClass(SubscriptionTeacher.class);
configuration.addAnnotatedClass(SubscriptionConsult.class);
configuration.addAnnotatedClass(SubscriptionCourse.class);
configuration.addAnnotatedClass(SubscriptionGroup.class);
configuration.addAnnotatedClass(ScheduleHash.class); configuration.addAnnotatedClass(TeacherNotice.
configuration.addAnnotatedClass(ApiUser.class); configuration.addAnnotatedClass(ApiSession.
StandardServiceRegistryBuilder builder = new StandardServiceRegistryBu
.applySettings(configuration.getProperties()); SessionFactory sessionFactory
configuration.buildSessionFactory(builder.build()); Визовом метода addAnnotatedClass() ми додаємо клас
сутності БД у конфігурацію Hibernate. Більш детально про налаштування Hibernate мова йдеться у наступній ч
цього підрозділу. Модуль ServicesModule визначає залежності які відносяться до певних сервісів (менеджерів), та
менеджер розкладу, менеджер часу та ін. Код налаштування залежностей у цьому модулі приведено у лістин
Лістинг 3.17 – Метод configure() модуля ServicesModule protected void configure
bind(SubsService.class).in(Scopes.SINGLETON); bind(ScheduleService.class) .to(ScheduleServiceImpl
.in(Scopes.SINGLETON); bind(PointsService.class).in(Scopes.SINGL
bind(SchedulerBot.class).in(Scopes.SINGLETON); bind(TimerService.class).in(Scopes.SINGL
bind(ApiUserService.class).in(Scopes.SINGLETON); } Надалі, для доступу до цих залежностей достатньо лише в
анотацію @Inject у конструкторі класу, в який треба запровадити залежність. Для впровадження залежності об'єкт
класу, його треба створити не через ключове слово «new», а використовуючи об'єкт Injector, який надає бібліотека
Запит залежностей через конструктор приведений на прикладі менеджеру підписок у лістингу 3.18. Лістинг 3.18 –
залежностей через конструктор public final class SubsService { private final TeacherSubsDao teacherDao; p
final ConsultSubsDao consultDao; private final CourseSubsDao coursesDao; private final PointsSu
pointsDao; private final NoticesDao noticesDao; private final GroupSubsDao groupsDao; @Inject
SubsService(TeacherSubsDao teacherDao, ConsultSubsDao consultDao, CourseSubsDao cours
PointsSubsDao pointsDao, NoticesDao noticesDao, GroupSubsDao groupsDao) { this.teacherDao = teach
this.consultDao = consultDao; this.coursesDao = coursesDao; this.pointsDao = pointsDao; this.notices
noticesDao; this.groupsDao = groupsDao; } } 3.2.4 Програмування взаємодії з БД Взаємодія з базою
відбувається за допомогою фреймворку Hibernate, який описувався у другому розділі. Першим кроком впров
цього фреймворку буде створення сутностей, які представляють записи у таблицях бази даних. Відповід
спроектованих раніше таблиць, можна виділити наступні класи сутностей: * BotUser – представляє користувача с
* SubsTeacher – представляє підписку на розклад викладача. * SubsConsult – представляє підписку на р
консультацій. * SubsCourse – представляє підписку на розклад курсу. * SubsGroup – представляє підписку на р
групи. * SubsPoints – представляє запис даних для авторизації на сервері успішності. * ScheduleHash – предс
останню хеш суму файлу розкладу. * ApiUser – представляє користувача панелі керування. * ApiSession – предс

сесію користувача панелі керування. Кожна сутність має бути відмічена спеціальними анотаціями, які визн структуру майбутньої таблиці БД, зв'язки з іншими сутностями, або навіть власні типи та дескриптори до них. П класу, анотованого таким чином, видно на рисунку 3.17. Рисунок 3.17 – Анотований клас BotUser Створені класи сут необхідно зареєструвати для обробки доданих аннотацій. Враховуючи, що в проекті використовується Dependency Injection, буде доцільно ініціалізувати Hibernate та його сутності у модулі PersistenceModule відповідає за реєстрацію залежностей, які відносяться до роботи за базою даних. Ініціалізація сутностей приве лістингу 3.19. Лістинг 3.19 – Ініціалізація сутностей Hibernate Configuration configuration = new Configura configuration.addProperties(conf.getDbProperties()); configuration.addAnnotatedClass(SubsPoint. configuration.addAnnotatedClass(SubsTeacher.class); configuration.addAnnotatedClass(SubsConsult. configuration.addAnnotatedClass(SubsCourse.class); configuration.addAnnotatedClass(SubsGroup. configuration.addAnnotatedClass(ScheduleHash.class); configuration.addAnnotatedClass(BotUser. StandardServiceRegistryBuilder builder = new StandardServiceRegistryBu .applySettings(configuration.getProperties()); SessionFactory sessionFactory configuration.buildSessionFactory(builder.build()); Для абстрагування управління сутностями було ви використовувати паттерн DAO. Він передбачає створення додаткових класів які інкапсулюють логіку роботи (наприклад сирі SQL запити), та надають інтерфейс для роботи з БД простими методами, властивими для програмування. Для кожного типу сутності треба створити DAO клас. Тоді можна виділити наступні класи: * User робота з сутностями користувачів. * SubsTeacherDao – робота з сутностями підписки на розклад виклад SubsConsultDao - робота з сутностями підписки на розклад консультацій. * SubsCourseDao - робота з сутн підписки на розклад курсів. * SubsGroupDao - робота з сутностями підписки на розклад груп. * HashesDao – ро хеш сумами файлів розкладу. Базовим класом для всіх DAO класів є абстрактний клас Dao, який має мето створення та автоматичного закриття сесій (підключень). Його структура позначена на рисунку 3.17. Рисунок 3.18 Dao Так, наприклад, він має метод withSession() для створення сесії без повертання результату. Код цього приведений у лістингу 3.20. Важливим є те, що використовуючи try-with-resources, сесія автоматично закрив після виходу з try блоку. Лістинг 3.20 – метод withSession() protected void withSession(Consumer consume (Session session = factory.openSession()) { Transaction t = session.beginTransac consumer.accept(session); t.commit(); } } Робота з базою даних через Hibernate та створений абстрактни видна на прикладі класу SubsTeacherDao. Наприклад, для автоматичного сповіщення є задача знайти всіх корист які не отримали сповіщення. Для цього є метод findNotNotified(), який також приймає ліміт на кількість зна сутностей. Його код приведений у лістингу 3.21. Лістинг 3.21 – Метод findNotNotified() public Coll findNotNotified(int limit) { return useSession(session -> { Query q = session.createQuery("from SubsTeacher notified = false"); q.setMaxResults(limit); return (Collection) q.list(); }); } Як і інші подібні методи, він використов (Hibernate Query Language) замість SQL для можливості змінити тип бази даних на інший у будь який час. О особливостей цієї мови запитів є використання назви сутності замість назви таблиці, тому вибірка йде з сутносте SubsTeacher. Інші DAO класи та їх методи майже не відрізняються один від одного та мають подібні до предста CRUD методи. Для використання створених класів в інших місцях, їх необхідно зареєструвати для впровадження Guice DI. Для цього ми вже створювали PersistenceModule. Після реєстрації всіх конпонентів, метод confi виглядає так, як позначено на рисунку 3.19. Рисунок 3.19 – Реєстрація компонентів в Guice 3.2.5 Програм менеджеру підписок Менеджер підписок надає доступ та можливість управління підписками користувачів на розкл реалізований у класі SubsService. Конструктор та залежності цього класу вже описувались у підрозділі про впров залежностей. Клас залежить від DAO класів, які надають доступ до таблиць БД з інформацією про підписки. Стр цього класу позначена на рисунку 3.21. Рисунок 3.21 – Структура класу SubsService Суть цього менеджеру в н інтерфейсу взаємодії з підписками без безпосередньої роботи з CRUD методами від DAO класів. 3.2.6 Програм командної системи Система «розмови» з ботом складається з двох частин – команд та запитів. Команда почина символу «/» (слеш) та не має аргументів для простоти використання на мобільних пристроях. Запит може бути вводу простого тексту або натискання на кнопку віртуальної клавіатури в чаті. Відповідно до спроектованої моделі, графу команд, для створення станів та переходів між ними, були написані класи State та ChatInput відповідн зберігання сесії «розмови» створений клас ChatSession. Структура цих класів позначена на рисунку 3.22. Рисунок Структура класів командної системи Клас State є абстрактним, та представляє стан діалогу з ботом. Він має посила наступний стан та два головних метода – activate() та input(). Метод activate() визивається автоматично, кол діалогу оновлюється на до певного стану. Поведінка при активації визначається реалізацією цього метода у класі п стану, але частіше всього цей метод виводить в чат інформацію про зміну стану, наприклад запит за ввід

інформації. Метод input() визивається, коли користувач робить ввід даних, поки знаходиться на поточному Сигнатуру метода видно на рисунку 3.22, він приймає об'єкт класу ChatInput, та поточну сесію діалогу, як інформацію про користувача та діалог. Метод повертає результат вводу у виді одного з трьох варіантів перечис NEXT, WRONG та STAY. Від результату обчислення залежить подальша поведінка боту. При результаті NEXT, б перейти до наступного стану, якщо він існує, тоді у наступного стану буде визваний метод activate(). При рез WRONG бот виведе помилку користувачу, вірогідно через неправильній ввід даних. При результаті STAY бот н робити ніяких дій. Стан, який частіше всього використовується це список варіантів вибору. Його реалізація знаходи класі ChoiceState. Його структура позначена на рисунку 3.23. Рисунок 3.23 – Клас ChoiceState Клас наслідує від та додає декілька методів, які мають реалізувати класи, які будуть наслідувати від нього. Найважливішим методом є buildKeyboard() який приймає список пар типу строчка-строчка як аргумент та віддає екземпляр клавіа варіантами вибору, яку виведе бот. Першою строчкою у парі є назва кнопки, яка буде виведена, а другою – те команда, яка буде виконана при натисканні. Код генерування клавіатури вибору приведений у лістингу 3.22. Лістинг Код генерування клавіатури protected InlineKeyboardMarkup buildKeyboard(int page, List> data) { var bu InlineKeyboardMarkup.builder(); int pages = (int) Math.ceil((float) data.size() / elemOnPage()); int fr Math.max(0, elemOnPage() * page); int to = Math.min(from + elemOnPage(), data.size()); List> pageD data.subList(from, to); List row = new LinkedList<>(); int i = 0; for (Pair pair : pageData) { if (i % 2 = builder.keyboardRow(row); row = new LinkedList<>(); } row.add(InlineKeyboardButton.builder() .text(pair .callbackData(pair.value()) .build()); i++; } if (!row.isEmpty()) builder.keyboardRow(row); if (data.si elemOnPage()) { if (page >= pages - 1) { // Last page builder.keyboardRow(Arrays. getPageBtn(firstPageText, 0), getPageBtn(prevPageText, page - 1))); } else if (page == 0) { // First builder.keyboardRow(Arrays.asList(getPageBtn(nextPageText, page + 1), getPageBtn(lastPageText, pages - else { builder.keyboardRow(Arrays.asList(getPageBtn(firstPageText, 0), getPageBtn(prevPageText, page getPageBtn(nextPageText, page + 1), getPageBtn(lastPageText, pages - 1))); } } return builder.build(); } Важ частиною тут є аргумент page, за допомогою якого можна розрахувати, які елементи треба вивести. Це зумовл обмеженням на кількість кнопок у клавіатурі, так і зручністю використання. Всі стани діалогу зберігаються у ст HashMap, де ключом є назва команди, а значенням стартовий стан. Ці данні зберігає об'єкт класу ChatManager цей клас відповідає за управління діалогами. Його структура позначена на рисунку 3.24. Рисунок 3.24 - ChatManager Найбільш важливим тут є метод handleUpdate(), в який передаються дані про запит до бота. У л 3.23 приводиться частина коду цього метода, яка управляє ходом діалогу за результатом обробки запиту. Лістинг Управління діалогом за результатом запиту State state = session.getState(); if (state != null) { InputResult re state.input(input, session); if (result.equals(InputResult.STAY)) return; if (result.equals(InputResult.NE session.updateState(state.getNext()); return; } if (!state.hasNext()) endSession(chatId) bot.sendMessage(session, bot.getLang().of("cmd.unsupported")); 3.2.7 Програмування таймеру розкладу відповідає за перевірку оновлення розкладу та синхронізацію масової розсилку цього розкладу. Для цього був ств клас TimerService. Його структура позначена на рисунку 3.25. Рисунок 3.25 – Клас TimerService Клас містить 2 о таймери – таймер перевірки оновлень та таймер розсилки розкладу. Їх ініціалізація відбувається у методі start(приведений у лістингу 3.24. Лістинг 3.24 – Метод start() класу TimerService public void start() { checkT timer.scheduleWithFixedDelay(this::check, 0L, conf.getCheckRate(), TimeUnit.SECONDS); sendTas timer.scheduleWithFixedDelay(this::mailSubscribers, 0L, 2L, TimeUnit.SECONDS); } Перший таймер, з пе зазначеним у конфігурації, визиває метод check() для перевірки оновлень розкладу. Метод за допомогою мен розкладу робить запит на оновлення розкладу. Якщо розклад був оновлений, для кожного користувача, як підписаний на цей розклад, видаляється флаг про закінчення сповіщення. Приклад коду який це робить привед лістингу 3.25. Лістинг 3.25 – Перевірка оновлення розкладу викладачів (scheduleService.reloadTeacherSchedule(false)) { subsService.resetTeacherMailing(); logger.info("Te schedule reloaded. Marked everyone for mailing"); } Після того як флаг оповіщення був знятий, таймер розсил наступній перевірці не сповіщених користувачів, відправить їм оновлений розклад, бо він був оновлений ще при перевірці. Так, раз в 2 секунди визивається метод mailSubscribers(), який послідовно оповіщає всіх хто підписа відповідний розклад. Приклад сповіщення приведений у виді коду сповіщення викладачів у лістингу 3.26. Лістинг Сповіщення викладачів Collection notMailed = subsService.getNotMailedTeacherSubs(); if (!notMailed.isEmp for (SubscriptionTeacher sub : notMailed) { ScheduleRenderer renderer = scheduleService.getTeacherSche .getPersonalRenderer(sub.getTeacher(), scheduleService); sendPhoto(sub.getTelegra renderer.renderBytes(), bot.getLang().of("mailing.teacher")); sub.setReceivedMailing(true);

subsService.updateTeacherSubs(notMailed); logger.info("Sent " + notMailed.size() + " messages during tea mailing"); } Як видно, першим кроком є знаходження всіх підписаних користувачів зі знятою міткою про заве сповіщення. Якщо такі користувачі знайдені, для кожного рендериться розклад, та відсилається через чат-бот. тіл прикінці розсилки всі сповіщені користувачі відмічаються як сповіщені одним запитом до БД. Така систем розсилки її більш гарантованою, у випадку якщо під час розсилки трапиться критичний сбій, тому що невідмічені про у сповіщення користувачі все одно будуть сповіщені при наступній роботі таймеру. 3.2.8 Програмування головного бота Головним класом боту є SchedulerBot, який наслідує від бібліотечного класу TelegramLongPollingB відповідає за взаємодію з Telegram API за допомогою бібліотеки TelegramBots. Його структура позначена на р 3.26. Клас має методи для відправки повідомлень різного типу, та слухає запити від користувачів. Для прослух запитів користувачів, треба перевизначити метод onUpdateReceived(). Його код приведений у лістингу 3.27 Лісти – Прослухування запитів public void onUpdateReceived(Update update) { CompletableFuture.runAsync(chatManager.handleUpdate(update), threadPool); } Для ефективності обробки запитів, був створений пул п Кількість потоків у ньому налаштовується у конфігурації програми. Таким чином, кожен запит виконується не бл головний поток. Рисунок 3.26 – Клас SchedulerBot Також клас здійснює управління відправкою повідомлень, обмеження Telegram у 30 повідомлень в секунду. Для цього була створена черга повідомлень (поле sendQ позначене у структурі класу), та таймер, який кожну секунду забирає до 30 повідомлень з черги та відправляє безпосередньо до користувачів. За це відповідає метод sendFromQueue(), який і визивається таймером. Йо приведений у лістингу 3.28. Лістинг 3.28 – Метод sendFromQueue() private void sendFromQueue() (!exe.isShutdown()) { for (int i = 0; i < MAX_PER_SECOND; i++) { SendMethod method = sendQueue.po (method != null) sendNow(method.session(), method.method()); } } } В свою чергу, метод send(), який визив у багатьох містах, не відправляє повідомлення одразу, а лише додає його до черги, використовуючи стандартний класу Queue - offer(), який додає елемент у кінець черги. Код цього метода приведений у лістингу 3.29. Лістинг Метод send() public void send(ChatSession session, Object method) { if (method != null) sendQueue.offe SendMethod(session, method)); } 3.3 Розгортання Процес розгортання будь якого серверу починається з пер апаратного та програмного комплексу. Для встановлення чат боту, інститут надав доступ до серверу ProLiant DL 3 який підтримує до двох процесорів Intel Xeon з технологією EM 64 T, що розширюється до 12 Гб пам'ять PC 2-3200 2 з частотою 400 МГц, три повнорозмірні слоти розширення, гігабітний мережевий контролер і вбудований кон SmartArray 6 і RAID. Крім того, цей сервер оснащений оригінальним дисковим кошиком, який забезпечує роботу жо дисків SCSI на одному або двох каналах. На час розгортання, на даному сервері були встановлені такі апаратні еле як: * Процесор Intel Xeon 3.3 ГГц * RAM об'ємом 3 ГБ В якості операційної системи, на сервері встановлено U Server версії 16.04. Для управління сервером віддалено використовується протокол SSH. Сервер, який був на інститутом, вже має налаштований SSH сервер. Також, для передачі файлів необхідний доступ до файлової системи протокол FTP, який також встановлений та налаштований на даному інститутом сервері. Діаграма розгортання поз на рисунку 3.27. Головним вузлом є «Application Server», який має три модуля – середовище виконання, викон файл, та локальну базу даних. Він автоматично підключається до зовнішніх вузлів по протоколу HTTPS, тому гол задачею є встановлення вузлу «Application Server». Рисунок 3.27 – Діаграма розгортання серверу 3.3.1 Встано середовища виконання програми Персональний помічник являє собою програму, написану на мові Java, тому розгортання необхідне середовище виконування, яке називається JVM (Java Virtual Machine). Ubuntu Serv встановлена на фізичному сервері інституту, використовує пакетний менеджер APT (Advanced Packaging Tool вся перевірка залежностей будет проходити автоматично. Першим кроком при встановленні пакету є оновлення репозиторіїв. Це робиться командою apt update, як показано на рисунку 3.28 Для багатьох операції використовувати права адміністратора. Рисунок 3.28 – Оновлення списку репозиторіїв Після оновлення необхідно ім'я пакету останньої версії JVM. На момент розгортання остання версія – 17. Для цього можна використати ко пошуку, яка позначена на рисунку 3.29 Ця команда виводить знайдені пакети за ім'ям, введений у команду. Рисунок Пошук потрібного пакету Так як для роботи чат боту необхідна лише JVM, доцільно вибрати пакет з мінімальним на компонентів. Згідно стандарту, пакети з приставкою «jre» (Java Runtime Environment) містять лише JVM та не для її роботи бібліотеки, без компілятору та інших інструментів. Тому для встановлення будемо використовувати openjdk-17-jre. На рисунку 3.30 приведена команда, яка використовується для встановлення нового пакету. Рисун – Встановлення пакету Після погодження зі встановленням, та деякого часу, пакет та всі необхідні залежності встановлені на сервері. Для перевірки наявності віртуальної машини у системі, та правильно налагоджений з середовища, можна ввести команду для перевірки версії віртуальної машини, яка приведена на рисунку 3.31. Рисун – Перевірка наявності віртуальної машини у системі Після того, як версія встановленої машини виводиться,

вважати, що середовище для розгортання чат боту успішно налагоджене. Наступний крок – встановлення чат бот
Встановлення чат боту Для встановлення чат боту використовується заздалегідь скомпільований .jar файл,
головним виконуваним файлом. Для переміщення файлів між сервером та локальною машиною використов
протокол FTP. Для цього можна використати будь який FTP клієнт. Для роботи чат боту необхідно налаштувати базу
Було вирішено використовувати локальну легку базу даних H2. Для того, щоб мати доступ до неї, боту необхідний
драйвер, який заздалегідь був завантажений з офіційного сайту програми. Тож для розгортання необхідно скопіюв
сервер два файли: * Головний виконуваний файл scheduler.jar * h2-2.1.100.jar – JDBC драйвер для управління БД
того, як файли скопійовані на сервер, необхідно провести перший запуск програми для того, щоб вона створила
конфігурації. Для цього можна ввести коротку команду для запуску Java програми, яка приведена на рисунк
Рисунок 3.32 – Перший старт програми Після першого запуску були створені всі необхідні файли. Перший раз
може підключитись до БД та API Telegram, бо вони ще не налаштовані. Для початку налаштування основних пара
треба відкрити файл config.yml. Для редагування файлів на сервері можна використовувати утиліту «nanc
налаштування підключення до бази даних, необхідно змінити властивості у блоці «database», як показано на р
3.33. Всі можливі властивості та їх значення можна знайти у документації до фреймворку Hibernate. Рисунок
Налаштування доступу до БД Для отримання списку запитів до боту, необхідно вписати його токен та ім’я у від
поля блоку «telegram», як показано на рисунку 3.34. Рисунок 3.34 – Налаштування токену чат бота Після цього мож
раз запустити бота та перевірити що він провантажується до кінця, як показано на рисунку 3.35. Рисунок 3.35 –
запуск бота Для стабільної та безперебійної роботи бота, треба налагодити його автоматичний перезапуск при авар
виході програми або перезавантажені операційної системи. Для цього використовується стандартний для Ubu
багатьох інших дистрибутивів Linux, підсистема ініціалізації та управління службами systemd. Для того, щоб ст
нову службу, треба написати скрипт для її запуску у директорії /etc/systemd/system. Кожен файл цієї дирек
розширенням .service є окремою службою. Для створення служби, яка буде контролювати життєвий цикл ча
створемо файл який буде називатись scheduler.service. Далі додамо інструкції, які позначені на рисунку 3.36. Р
3.36 – Скрипт для запуску сервісу Параметр ExecStart це шлях до скрипту запуску Java програми start.sh. Він м
строку, яка запускає програму: java –jar scheduler.jar Для старту загрузки сервісу у підсистему system, треба
наступну команду: sudo systemctl enable scheduler.service Після того, як сервіс доданий, його треба акти
наступною командою: sudo systemctl start scheduler Після старту сервісу, його статус можна подивитись за допо
команди systemctl status scheduler. Статус чат боту приведений на рисунку 3.37. В даному випадку сер
запущений коректно та працює. Рисунок 3.37 – Статус сервісу чат боту Тепер чат бот буде автоматично завантажу
при старті системи, та при аварійному відключені програми. 3.3.3 Перевірка робото спроможності розгорнутої с
Після розгортання необхідно перевірити робото спроможність системи. Для цього було створено 12 тест кейсів.
Назва: Базова робота команд. Очікуваний результат: Вивід списку доступних команд. Отриманий результат: Вивід
доступних команд. Рисунок 3.38 - Вивід програми Case 2 Назва: Робота розкладу курсів. Очікуваний результат
розкладу обраного курсу у виді зображення. Отриманий результат: Вивід розкладу обраного курсу у виді зобра
Рисунок 3.39 - Результат запиту Case 3 Назва: Робота розкладу зайнятості викладача. Очікуваний результат
зайнятості обраного викладача у виді зображення. Отриманий результат: Вивід зайнятості обраного викладача
зображення. Рисунок 3.40 - Результат запиту Case 4 Назва: Процес підписки на розклад викладача. Очік
результат: Вивід зайнятості обраного викладача у виді зображення. Повідомлення про оформлену підписку на о
розклад. Отриманий результат: Вивід зайнятості обраного викладача у виді зображення. Повідомлення про офор
підписку на обраний розклад. Рисунок 3.41 - Результат запиту Case 5 Назва: Робота підписки на розклад викл
Очікуваний результат: Вивід розкладу викладача, на якого була підписка. Отриманий результат: Вивід розкладу викл
на якого була підписка. Рисунок 3.42 - Результат запиту Case 6 Назва: Автоматичне отримання розкладу виклада
оновлені. Очікуваний результат: Бот автоматично присилає оновлений розклад. Отриманий результат: Бот автом
присилає оновлений розклад. Рисунок 3.43 - Результат запиту Case 7 Назва: Вивід розкладу групи. Очікуваний рез
Виведений розклад обраної групи. Отриманий результат: Виведений розклад обраної групи. Рисунок 3.44 - Ре
запиту Case 8 Назва: Вивід розкладу консультацій. Очікуваний результат: Виведений розклад консультацій об
викладача. Отриманий результат: Виведений розклад консультацій обраного викладача. Рисунок 3.45 - Результат
Case 9 Назва: Підписка на розклад консультацій. Очікуваний результат: Виведений розклад консультацій об
викладача. Повідомлення про успішну підписку. Отриманий результат: Виведений розклад консультацій об
викладача. Повідомлення про успішну підписку. Рисунок 3.46 - Результат запиту Case 10 Назва: Підписка на р
групи. Очікуваний результат: Виведений розклад обраної групи. Повідомлення про успішну підписку. Отриманий рез
Виведений розклад обраної групи. Повідомлення про успішну підписку. Рисунок 3.47 - Результат запиту Case 11

Перевірка підписки на розклад консультацій. Очікуваний результат: Виведений розклад консультацій, на яки підписка. Отриманий результат: Програма не відповідає, навіть після декількох запитів. На інші запити у той сам відповідь коректна. Рисунок 3.48 - Результат виконання команди декілька разів Case 12 Назва: Перевірка навант при великій кількості запитів. Очікуваний результат: З інтервалом 50 мс, програма не повинна використовувати біль 600 МБ оперативної пам'яті, та 80% ЦП. Отриманий результат: Максимально програма використовувала приблиз МБ оперативної пам'яті, та 75-80% ЦП. Використання ресурсів до та під час навантаження вказано на рисунках 3 3.50 відповідно. Рисунок 3.49 - Використання ресурсів без навантаження Рисунок 3.50 - Використання рес навантаженням 3.4 Висновки розділу Відповідно до висунутих вимог, було спроектовано систему модульного Спроектовано структура таблиць бази даних для зберігання даних про користувачів, підписки на розклад, та слу інформацію. Також було спроектовано конфігурацію програми та базову структуру проекту. Спроектована систем запрограмована з використанням обраних технологій. Було запрограмовано парсинг різних типів таблиць, їх ренд сутності та взаємодію з базою даних, різні менеджери та парсинг конфігурації. Запрограмована система протестована та розгорнута на базі інституту ЗІЕІТ. ВИСНОВКИ Було здійснено огляд поточної системи пуб розкладу ЗІЕІТ та інших ЗВО. Встановлено потребу у створенні системи швидкого сповіщення про новий розк зручним переглядом та персональними налаштуваннями користувача. Здійснено огляд систем електр відображення розкладу різних ЗВО, та найбільш ефективних механізмів автоматичного сповіщення корист Встановлено, що сповіщення через месенджер є найбільш доцільним для даної задачі. Враховуючи вимоги до шв сповіщення та легкості доступу до даних про розклад для найбільшої групи людей, вирішено реалізувати систему чат-боту для месенджеру. Здійснено порівняльну характеристику найбільш популярних месенджерів, в першу че платформ для створення чат-ботів, та вирішено використовувати месенджер Telegram. Проведено дослідження можливості отримувати актуальні дані про розклад занять, та виявлено, що парсинг публікованих таблиць є на доцільним. Для розробки чат-боту було обрано наступний стек технологій: Java 17, Apache POI, Hibe TelegramBots. В якості середовища розробки була обрана Intellij IDEA Community Edition. Спроектована систем запрограмована з використанням обраного технологічного стеку. Готовий чат-бот був протестований та розгорну базі інституту ЗІЕІТ. Розроблений програмний продукт відповідає поставленим до нього цілям, а саме надає ш доступ до розкладу занять з можливістю автоматичного сповіщення, та має простий у використанні інтерфейс. ПЕ ВИКОРИСТАНИХ ДЖЕРЕЛ 1. Офіційний сайт Запорізького Інститут Економіки та Інформаційних Технологій. [Електр ресурс]. – Режим доступу: www. URL: https://www.zieit.edu.ua/ (дата звернення: 01.02.22) 2. Державна с статистики України. [Електронний ресурс]. – Режим доступу: www. URL: https://ukrstat.gov.ua 3. Автомати система управління навчальним закладом. [Електронний ресурс]. – Режим доступу: www. URL: http://mkr.org Офіційний сайт Запорізького Національного Університету. [Електронний ресурс]. – Режим доступу: www https://www.znu.edu.ua/ 5. Офіційний сайт Запорізького Державного Медичного Університету. [Електронний рес Режим доступу: www. URL: https://zsmu.edu.ua/ 6. Офіційний сайт Київського Європейського Універ [Електронний ресурс]. – Режим доступу: www. URL: https://e-u.edu.ua/ 7. Офіційний сайт Національного мед університету імені О.О. Богомольця. [Електронний ресурс]. – Режим доступу: www. URL: http://nmuofficial.com/ бот від Національного Університету "Полтавська Політехніка Імені Юрія Кондратюка". [Електронний ресурс]. – доступу: www. https://nupp.edu.ua/news/rozklad-zanyat-zavzhdi-v-telefoni-universitet-zapustiv-na-tele chat-bot-dlya-studentiv.html 9. Telegram Messenger. [Електронний ресурс]. – Режим доступу: www https://telegram.org/ 10. Viber Messenger. [Електронний ресурс]. – Режим доступу: www. https://www.viber.com/ 11. WhatsApp Messenger. [Електронний ресурс]. – Режим доступу: www https://www.whatsapp.com/ 12. Facebook Messenger. [Електронний ресурс]. – Режим доступу: www https://www.messenger.com/ 13. Визначення парсингу. [Електронний ресурс]. – Режим доступу: www https://uk.wikipedia.org/wiki/Синтаксичний_аналіз 14. Впровадження машинного навчання в докумен [Електронний ресурс]. – Режим доступу: www. URL: https://www.tadviser.ru/a/417209/ 15. Rendering. [Елект ресурс]. – Режим доступу: www. URL: https://en.wikipedia.org/wiki/Rendering_(computer_graphics Rasterization. [Електронний ресурс]. – Режим доступу: www. URL: https://en.wikipedia.org/wiki/Rasterisat Microsoft XLS format specification. [Електронний ресурс]. – Режим доступу: www. https://interoperability.blob.core.windows.net/files/MS-XLS/%5bMS-XLS%5d.pdf 18. Microsoft XLSX f specification. [Електронний ресурс]. – Режим доступу: www. https://interoperability.blob.core.windows.net/files/MS-XLSX/%5bMS-XLSX%5d.pdf 19. Client-Server archite [Електронний ресурс]. – Режим доступу: www. URL: https://cio-wiki.org/wiki/Client_Server_Architecture/ 20. W REST. [Електронний ресурс]. – Режим доступу: www. URL: https://restfulapi.net/ 21. Чат бот онлайн банку Мо

[Електронний ресурс]. – Режим доступу: www. URL: https://www.monobank.ua/contacts/ 22. Мова розмітки [Електронний ресурс]. – Режим доступу: www. URL: https://wikipedia.org/wiki/YAML 23. Специфікація мови [Електронний ресурс]. – Режим доступу: www. URL: https://yaml.org/spec/1.2.2/ 24. Популярність IDE для [Електронний ресурс]. – Режим доступу: www. URL: https://snyk.io/blog/intellij-idea-dominates-the-ide-m with-62-adoption-among-jvm-developers/ 25. Guice Framework. [Електронний ресурс]. – Режим доступу: URL: https://github.com/google/guice/ 26. Керівництво користувача Guice. [Електронний ресурс]. – Режим д www. URL: https://netvl.github.io/guice/users-guide.html 27. Log4j2 Manual. [Електронний ресурс]. – Режим д www. URL: https://logging.apache.org/log4j/2.x/manual/index.html 28. Бібліотека Hibernate. [Електронний р – Режим доступу: www. URL: https://ru.wikipedia.org/wiki/Hibernate_(библиотека) 29. Apache POI. [Елект ресурс]. – Режим доступу: www. URL: https://poi.apache.org/components/index.html 30. Специфікація RE панелі керування. [Електронний ресурс]. – Режим доступу: www. https://github.com/Nan1t/Scheduler/blob/rest/REST.md ДОДАТОК А ВИХІДНИЙ ПРОГРАМНИЙ КОД

Source

РЕЗЮМЕ Бакалаврська робота містить 53 сторінок пояснюючої записки, 1... 0.13%

РЕЗЮМЕ Бакалаврська робота містить 53 сторінок пояснюючої записки, 15 рисунків, 8 таблиць, 2 додатки.

http://dspace.wunu.edu.ua/bitstream/316497/39117/1/%D0%91%D0%BE%D0%BD%D0%B4%D0%B0%D1%80%D0%B5%D1%86__%D0%94%D0%9F_2019.pdf

Що таке. Рендер – це що таке? Як його налаштувати? 3d ... 0.13%

Що таке. Рендер – це що таке? Як його налаштувати? 3d ...

https://gstore03.ru/uk/chto-takoe-render

Регулярний вираз - Вікіпедія 0.13%

Регулярний вираз - Вікіпедія

https://www.wiki.uk-ua.nina.az/%D0%A0%D0%B5%D0%B3%D1%83%D0%BB%D1%8F%D1%80%D0%BD%D0%B8%D0%B9_%D0%B2%D0%B8%D1%80%D0%B0%D0%B7.html

by IM Зурілов · 2021 — Відсутність стану – кожен запит від клієнта до серв... 0.13%

by IM Зурілов · 2021 — Відсутність стану – кожен запит від клієнта до сервера повинен містити всю інформацію, необхідну для розуміння запиту, і не може використовувати.

https://krs.chmnu.edu.ua/jspui/bitstream/123456789/1922/1/%D0%90%D0%B2%D1%82%D0%BE%D1%80%D0%B5%D1%84%D0%B5%D1%80%D0%B0%D1%82%20%D0%97%D1%83%D1%80%D1%96%D0%BB%D0%BE%D0%B2%20%D0%9C.%20%D0%86..pdf

Мова java навіщо використовується. Мова програмування Java ... 0.13%

Мова java навіщо використовується. Мова програмування Java ...

https://gamessoon.ru/uk/the-java-language-for-what-is-used-java-programming-language/

Мова програмування Java: з чого розпочати вивчення. Де ... 0.26%

Мова програмування Java: з чого розпочати вивчення. Де ...

https://ruketa.ru/uk/yazyk-programmirovaniya-java-s-chego-nachat-izuchenie-gde-primenyaetsya-java.html

by MO Омельянюк · 2020 — Java-анотацій. При використанні файлу XML ... 0.13%

by MO Омельянюк · 2020 — Java-анотацій. При використанні файлу XML Hibernate може генерувати скелет вихідного коду для класів тривалого зберігання. У.

https://er.nau.edu.ua/bitstream/NAU/41940/1/%D0%A4%D0%9A%D0%9A%D0%9F%D0%86_2020_122_%D0%9E%D0%BC%D0%B5%D0%BB%D1%8C%D1%8F%D0%BD%D1%8E%D0%BA%D0%9C%D0%9E.pdf

by IA Татошвілі · 2020 — якщо використовується анотація. Hibernate мож... 0.13%

by IA Татошвілі · 2020 — якщо використовується анотація. Hibernate може використовувати файл XML або анотації для підтримки схеми бази даних.

https://ela.kpi.ua/bitstream/123456789/35028/1/Tatoshvili-I-A_bakalavr.pdf

[jboss-cvs] Picketbox SVN: r276 - in trunk 0.13%

[jboss-cvs] Picketbox SVN: r276 - in trunk

https://lists.jboss.org/pipermail/jboss-cvs-commits/2011-October/129365.html

Запорізький інститут економіки та інформаційних технологій 0.13%

Запорізький інститут економіки та інформаційних технологій

https://uk.wikipedia.org/wiki/%D0%97%D0%B0%D0%BF%D0%BE%D1%80%D1%96%D0%B7%D1%8C%D0%BA%D0%B8%D0%B9_%D1%96%D0%BD%D1%81%D1%82%D0%B8%D1%82%D1%83%D1%82_%D0%B5%D0%BA%D0%BE%D0%BD%D0%BE%D0%BC%D1%96%D0%BA%D0%B8_%D1%82%D0%B0_%D1%96%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D0%B8%D1%85_%D1%82%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D1%96%D0%B9

Національний медичний університет імені О. О. Богомольця 0.13%

Національний медичний університет імені О. О. Богомольця

https://uk.wikipedia.org/wiki/%D0%9D%D0%B0%D1%86%D1%96%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%B8%D0%B9_%D0%BC%D0%B5%D0%B4%D0%B8%D1%87%D0%BD%D0%B8%D0%B9_%D1%83%D0%BD%D1%96%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%82%D0%B5%D1%82_%D1%96%D0%BC%D0%B5%D0%BD%D1%96_%D0%9E._%D0%9E._%D0%91%D0%BE%D0%B3%D0%BE%D0%BC%D0%BE%D0%BB%D1%8C%D1%86%D1%8F

Національний університет "Полтавська політехніка імені ... 0.13%

Національний університет "Полтавська політехніка імені ...

https://www.facebook.com/poltava.polytechnic.edu/