МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧОРНОМОРСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ПЕТРА МОГИЛИ

Головатий Владислав Русланович

УДК 004.4

Автоматична генерація документації до коду на Java

Напрям підготовки 6.050101 – «Комп’ютерні науки»
ДР.ПЗ-401.10790506

Автореферат
дипломної роботи на здобуття освітньої кваліфікації
«Бакалавр комп’ютерних наук»

Миколаїв – 2019

Дипломна робота є рукопис.

Робота виконана в Чорноморському національному університеті імені Петра Могили Міністерства освіти і науки України на кафедрі інтелектуальних інформаційних систем

Науковий керівник: к.пед.н., доцент Болюбаш Надія Миколаївна.

Рецензент: ст. викладач, Бурлаченко Іван Сергійович.

Захист відбудеться «25» червня 2019 р. о $9^{\underline{30}}$ год. на засіданні екзаменаційної комісії (ауд. 2-403) у Чорноморському національному університеті імені Петра Могили за адресою: 54003, м. Миколаїв, вул. 68-ми Десантників, 10.

З дипломною роботою можна ознайомитися в бібліотеці Чорноморського національного університету імені Петра Могили за адресою: 54003, м. Миколаїв, вул. 68-ми Десантників, 10.

Автореферат представлений «20» червня 2019 р.

Секретар
екзаменаційної комісії,
ст.викл. С.В.Дворецька

ЗАГАЛЬНА ХАРАКТЕРИСТИКА РОБОТИ

Актуальність. Сучасні проекти мають у собі значний обсяг документації, який сприяє тому, що сам проект стає більшим за обсягом, його стає важко підтримувати на актуальному рівні, виникають неточності і з плином часу, коли проект розростається – проблема погіршується. На вирішення цих проблем компанії витрачають значні кошти та ресурси, які могли б витрачати на більш необхідні задачі; а програмісти витрачають свій час та компетенції на нецікаву та рутинну роботу, у той час, як програмний продукт зовсім не змінюється.

Сучасні засоби автоматичної генерації документації мають свої недоліки: якщо використовувати стандартні засоби мови Java, то проект буде перевантажений службовою документацією та спец. символами, а UI такої документації буде на низькому рівні; якщо використовувати існуючі бібліотеки (Doxygen, HappyDoc та інші), то зі зміною стандарту мови Java або інших складових – бібліотека більше не зможе виконувати свої функції. Тому виникає потреба у засобі, який буде виконувати звої функції, навіть при зміні стандарту мови Java, матиме гарний та сучасний UI і не потребуватиме додаткового коду для коректної роботи.

Деякі програмні продукти мають у собі API, яке дозволяє іншим програмістам використовувати функціонал проекту для своїх задач. А оскільки розробка та підтримка API тісно пов’язана з документацією проекту, то використання того чи іншого засобу для роботи із документацією напряму впливає на продуктивність роботи розробників та бюджет проекту. Неактуальна версія або неточності у API можуть стати причиною значних фінансових втрат.

Метою дипломної роботи є підвищення ефективності створення та підтримки документації шляхом розробки та впровадження бібліотеки, спрямованої на автоматизацію комплексної роботи з документацією.

Досягнення поставленої мети обумовлює необхідність вирішення наступних завдань:

1. Розкрити теоретичні засади автоматизації розробки документації.
2. Дослідити підходи до розробки різних аспектів документації.
3. Проаналізувати сучасний стан програмного забезпечення, яке використовують для автоматизації розробки документації.
4. Обрати на основі проведеного аналізу технології та засоби розробки бібліотеки для створення документації.
5. Розробити та здійснити програмну реалізацію бібліотеки для автоматизації створення документації.

Практичне значення отриманих результатів. Автоматична генерація документації дасть смогу витрачати значно менше часу, людських ресурсів та компанії. Враховуючи те, що у сучасному світі програмні продукти розробляються великою кількістю людей, а компанії витрачають на підтримку, ведення впровадження та оновлення документації значні фінансові та людські ресурси – автоматича генерація документації дасть смогу витрачати ресурси на інші задачі, де необхідні висококласні навички розробників програмного забезпечення. У той час, як при нинішніх реаліях нудна та монотонна робота займає час програмістів. Також існує вірогідність допустити помилку, яка може призвести до значних фінансових, репутаційних та небажаних втрат. Додаткова робота витрачається на пошук помилок у документації та їх виправлення, а підтримувати документацію актуальною не завжди можливо.

Структура дипломної роботи. Пояснювальна записка до дипломної роботи складається із вступу, 3 розділів, висновків, додатків. Загальний обсяг роботи складає 73 сторінок, 5 рисунків та 28 посилань на літературні джерела.

ОСНОВНИЙ ЗМІСТ РОБОТИ

У вступі подано загальну характеристику досліджуваної теми, обґрунтовано актуальність дипломної роботи, сформульовано мету, завдання розробки, визначено основні етапи роботи над дипломним проектом.

У першому розділі визначено та дається тлумачення поняттям, які є основоположними для розглянутої теми, аби при розробці системи уникнути неоднозначності поставлених до системи вимог; розглядаються аналогічні програми та web-додатки, наводиться повний аналіз усіх актуальних способів ведення документації та підтримання її у актуальному стані. Розглядаються плюси і мінуси кожного способу та пропонуються методи вирішення проблемних частин. Наводяться основні генератори документації, які є актуальними у даний момент часу. Розглядається роль API у сучасних програмних застосунках, взаємозв’язок між документацією та API.

У другому розділі виконано більш детальний аналіз поставленої задачі з точки зору її математичного вирішення, виконаний аналіз існуючих засобів для виправлення недоліків аналогічних додатків та інструментів, розглянуто основні математичні методи її вирішення; розглянуто такі важливі поняття для вирішення задачі прогнозування, проаналізовано деякі критерії прийняття рішень та умови їх використання; зроблено висновки щодо того які з розглянутих методів використати для вирішення поставленої задачі та приведено аргументи щодо причин прийняття кожного рішення.

Розглянутий технічний стек та засоби розробки програмного забезпечення. Приводяться методи покращення процесу розробки додатку через збільшення продуктивності розширеного типу архітектури, наприклад, дані можуть бути передані як JSON. Ця високопродуктивна архітектура веб-додатків вимагає серверних технологій у веб-сервісі на стороні сервера, а також використання фреймворків JavaScript на стороні клієнта. Зв'язок цих моделей архітектури веб-додатків вимагає особливого механізму, наприклад, часто використовується Hash-Bang. Робота в режимі офлайн не завжди є успішною, оскільки існує проблема оновлення або кешування інформації про віджет. Звичайно, механізм маніфесту застосовується тут, але він все ще

вимагає додаткового часу та зусиль для взаємодії між файлом шаблону віджетів і логікою поведінки сторінки. Архітектура сервера додатків і веб-сервера - послідовне розташування веб-сервера. Вона забезпечує проектування, розробку та розгортання серверів. Основним завданням сервера прикладних програм і архітектури веб-сервера є задоволення вимог користувача щодо загального досвіду та зручності інтерфейсу. Ці програми існують на сервері під час роботи операційних систем і гарантують підключення до мережі. Вони встановлюють різні завдання, застосовуючи протоколи HTTP для ресурсу сторінки або REST API. Продуктивність, пам'ять, зберігання, рівні додатків і обчислювальна потужність є необхідними компонентами фізичної потужності сервера. Структура архітектури веб-сервера охоплює ці пункти для завершення запитів користувача до повного розширення. Архітектура веб-додатків Java зарекомендувала себе як універсальний фреймворк. Сьогодні вона займає провідні позиції серед мов програмування. З одного боку, ви можете мати справу з простим, описовим програмним забезпеченням, з іншого боку, це не проблема для управління жорсткими, багатошаровими. В обох випадках ви будете застосовувати високопродуктивну архітектуру веб-додатків і отримувати бажані результати. Розробники включають багаторівневу архітектуру в веб-додатках Java. Отже, складність структури архітектури веб-додатків залежить від встановлених вимог. Слід зазначити, що ця структура забезпечує високу продуктивність. Вона включає і керує кількістю наборів інструментів Java для створення рішень, які працюють у хмарі. У вас є можливість змінювати його інструменти і фреймворки для шаруватої архітектури в програмах Java, щоб отримувати різні операції, починаючи з найпростіших до складних повноцінних систем. Архітектура веб-додатків на основі Java зазвичай використовується в корпоративних рішеннях. Зроблений аналіз роботи JSF. JavaServer Faces (JSF) - це специфікація Java для побудови інтерфейсів користувача на основі компонентів для веб-додатків і формалізована як стандарт через процес спільноти Java, який є частиною

платформи Java, Enterprise Edition. Це також веб-фреймворк MVC, який спрощує побудову користувацьких інтерфейсів (UI) для серверних додатків, використовуючи повторно використовувані компоненти UI на сторінці. JSF 2 використовує Facelets як систему шаблонів за замовчуванням. Інші технології перегляду, такі як XUL або звичайний Java , також можуть використовуватися. На відміну від цього, JSF 1.x використовує JavaServer Pages (JSP) як систему шаблонів за замовчуванням. На основі компонентно-орієнтованої дизайнерської моделі користувальницького інтерфейсу JavaServer Faces використовує файли XML, звані шаблонами перегляду або переглядами Facelets. FacesServlet обробляє запити, завантажує відповідний шаблон перегляду, створює дерево компонентів, обробляє події і передає відповідь (зазвичай на мові HTML) на клієнта. Стан компонентів інтерфейсу та інших об'єктів інтересів області зберігається в кінці кожного запиту в процесі, який називається stateSaving (примітка: transient true), і відновлюється при наступному створенні цього перегляду. Або клієнт, або сервер може зберігати об'єкти та стани. JSF часто використовується разом з Ajax, Rich технологією розробки інтернет-додатків. Ajax - це комбінація методів і технологій веб-розробки, які дозволяють створювати багаті користувальницькі інтерфейси. Компоненти інтерфейсу користувача в Mojarra (посилання на JSF) і Apache MyFaces були спочатку розроблені тільки для HTML, і Ajax потрібно було додати через JavaScript. Однак це змінилося: Оскільки JSF підтримує кілька форматів виводу, компоненти Ajax можуть бути легко додані, щоб збагатити інтерфейси користувача на основі JSF. Специфікація JSF 2.0 надає вбудовану підтримку Ajax шляхом стандартизації життєвого циклу запитів Ajax і надання простих інтерфейсів розробки до подій Ajax, дозволяючи будь-якій події, викликаної клієнтом, пройти належну перевірку, перетворення і, нарешті, виклик методу, перш ніж повернути результат через браузер через оновлення XML DOM. Розроблений аналіз роботи шаблонізатора Thymeleaf. Thymeleaf - це шаблон XML Java, XHTML, HTML5, який може працювати як у веб-середовищі (на основі

сервлетів), так і в інших мережах. Він краще підходить для обслуговування XHTML, HTML5 у шарі перегляду веб-додатків на основі MVC, але він може обробляти будь-який файл XML навіть у автономному середовищі. Вона забезпечує повну інтеграцію Spring Framework. У веб-додатках Thymeleaf прагне бути повноцінною заміною сторінок JavaServer Pages (JSP) і реалізує концепцію природних шаблонів: файли шаблонів, які можна безпосередньо відкривати в браузерах і які все ще відображаються правильно як веб-сторінки.

- Шаблон Java для XML, XHTML і HTML5.
- Працює як в Інтернеті, так і в інших мережах (в автономному режимі). Немає жорсткої залежності від API Servlet.
- На основі модульних наборів функцій називають діалекти.
- Можливості діалекту (наприклад, оцінка, ітерація тощо) застосовуються, пов'язуючи їх з тегами та / або атрибутами шаблону.
- Два діалекти доступні поза коробкою: Standard і SpringStandard (для додатків MVC для весни, такий же синтаксис як стандартний).

У веб-додатках Thymeleaf прагне бути повноцінною заміною сторінок JavaServer Pages (JSP) і реалізує концепцію природних шаблонів: файли шаблонів, які можна безпосередньо відкривати в браузерах і які все ще відображаються правильно як веб-сторінки. Зроблений аналіз роботи хмарної платформи Heroku. Heroku - хмарна платформа як послуга (PaaS), що підтримує кілька мов програмування. Програми, які виконуються в Heroku, зазвичай мають унікальний домен (як правило, "applicationname.herokuapp.com"), який використовується для маршрутизації запитів HTTP до правильного dyno. Кожен з контейнерів додатків або dynos, поширюються на "сітку дино", яка складається з декількох серверів. Сервер Git у Heroku обробляє репозиторій додатків від дозволених користувачів. Основним змістом розробки є вихідний код, пов'язані залежності, якщо вони існують, і Procfile для команди. Програма надсилається до Heroku за допомогою будь-якого з наступних: Git, GitHub, Dropbox або через API.

Існують пакети, які приймають додаток разом з усіма залежностями, і мовою часу виконання, і виробляють слимаки. Вони називаються build-packs і є засобом для процесу компіляції slug. Slug - це комбінація розшарування вихідного коду, вбудованих залежностей, середовища виконання та компільованого генерованого виводу системи збирання, яка готова до виконання. Далі йде конфігурація, що містить настроювані конфігураційні дані, які можна змінювати незалежно від вихідного коду. Додаткові компоненти - це хмарні сервіси третьої сторони, спеціалізовані та додаткові, які легко додаються до програми, розширюючи її функціональність. Реліз - це комбінація slug (додатка), конфігурації і доповнення. Heroku підтримує журнал, відомий як головна книга випусків, що розробляється розробником. Основним блоком, який забезпечує середовище запуску, є Dynos, які є ізольованими, віртуалізованими контейнерами Unix. Формування Dyno програми є загальною кількістю виконуваних в даний час dynos, розділених між різними типами процесів, які розробник масштабував. Менеджер Dyno відповідає за управління dynos у всіх додатках, що працюють на Heroku. Програми, які використовують вільний тип dyno, будуть спати після 30 хвилин бездіяльності. Масштабування до декількох web-диносів або іншого типу dyno уникне цього. Одноразовий Dynos є тимчасовим динозом, який виконується з їх входом і виходом, приєднаним до локального терміналу. Вони завантажуються з останньою версією. Кожен дин отримує власну ефемерну файлову систему зі свіжою копією останнього випуску. Він може використовуватися як тимчасовий блокнот, але зміни до файлової системи не відображаються в інших dynos. Logplex автоматично зіставляє записи журналу з усіма запущеними диносами програми, а також іншими компонентами, такими як маршрутизатори, забезпечуючи єдине джерело діяльності. Масштабування програми передбачає зміну кількості динозів кожного типу процесу.

У третьому розділі наведений аналіз роботи використаних технолгій.

Технологія розробки програмного забезпечення розглядає питання управління проектуванням систем програмного забезпечення, а також кошти і стандарти розробки програм. Технологія розробки програмного забезпечення визначає деяку професійну культуру роботи фахівців (не тільки програмістів), що забезпечує заданий рівень продуктивності праці і якості одержуваної в результаті програмної продукції. Технологія розробки програмного забезпечення охоплює процес розробки програмного забезпечення від появи потреби в ньому до його виготовлення, передачі користувачеві, модифікації в процесі експлуатації і припинення його використання внаслідок морального старіння. Програмне забезпечення може бути розроблено для різноманітних цілей, причому три найбільш поширені для задоволення конкретних потреб конкретного клієнта / бізнесу (у випадку з користувальницьким програмним забезпеченням), для задоволення усвідомленої потреби в певному наборі потенційних користувачів (у випадку з комерційними та програмне забезпечення з відкритим вихідним кодом) або для особистого використання (наприклад, учений може написати програмне забезпечення для автоматизації звичайного завдання). Розробка вбудованого програмного забезпечення, тобто розробка вбудованого програмного забезпечення, наприклад, для контролю споживчих товарів, вимагає інтеграції процесу розробки з розвитком контрольованого фізичного продукту. Системне програмне забезпечення лежить в основі застосування і самого процесу програмування, і часто розробляється окремо. Необхідність кращого контролю якості процесу розробки програмного забезпечення породила дисципліну інженерії програмного забезпечення, яка має на меті застосувати системний підхід, що ілюструється інженерною парадигмою, до процесу розробки програмного забезпечення. Існує багато підходів до управління проектами програмного забезпечення, відомих як моделі життєвого циклу розробки програмного забезпечення, методології, процеси або моделі. Модель водоспаду є традиційною версією, яка контрастує з більш пізньою інновацією гнучкої розробки програмного забезпечення.

Обґрунтовано необхідність аналізу існуючих, розробки та важливість дотримання необхідних умов для безпеки та здоров’я програміста; розглянуто впливи на здоров’я та ефективність роботи небезпечних факторів, які можуть виникнути на робочому місці програміста; розраховано освітленість робочого приміщення та необхідної кількості штучного освітлення, та обрано необхідне для його забезпечення обладнання; розглянуто допустимі значення кожного з небезпечних чинників; визначено необхідний режим праці та ергономіку робочого місця.

ЗАГАЛЬНІ ВИСНОВКИ

За результатами виконання поставлених задач та проведеної роботи сформульовані такі висновки:

– здійснено визначення та аналіз особливостей документації до коду;

– розглянуто основоположні поняття для постановки задачі;

– проведено пошук та аналіз існуючих аналогів;

– сформульовано постановку задачі та, задля полегшення пошуку математичних методів вирішення поставленої задачі;

– вибраний необхідний стек технологій для розробки додатку;

– визначено та проаналізовано методи, що можуть використовуватися при розв’язанні поставленої задачі;

– обрано методи, які було б найбільш доцільно та ефективно використати для вирішення поставленої задачі;

– створено систему підтримки прийняття рішень, яка відповідає поставленій меті, та продемонстровано її роботу;

– розглянуто шкідливі фактори та необхідні умови праці, визначені нормативними документами;

– виконано розрахунок кількості необхідного штучного освітлення для робочого приміщення.

АНОТАЦІЯ

Головатий Владислав Русланович. Автоматична генерація документації до коду на Java. – На правах рукопису.

Дипломна робота на здобуття освітньої кваліфікації «Бакалавр комп’ютерних наук». – Чорноморський національний університет імені Петра Могили, Миколаїв, 2019.

Дипломна робота присвячена розробці та здійсненню програмної реалізації автоматичної генерації документації до коду на Java.. Додано матеріали з реалізацією основних функцій. Представлено свій погляд на подальший розвиток розробленої системи та її можливу модернізацію.

Пояснювальна записка до дипломної роботи складається із вступу, 4Зрозділи, висновків, додатків. Загальний обсяг роботи складає 73 сторінок, 5 рисунків, 2 додатоків та 28 посилань на літературні джерела. Спеціальна частина з техніки безпеки та цивільної оборони складає 11 сторінок.

Ключові слова: автоматизація, документація, підвищення ефективності, web-застосунок.

ABSTRACT

V.R. Holovatyi. Generation of documents is automatic to Java code. - On the rights of the manuscript.

Diploma work on obtaining an educational qualification "Bachelor of Computer Science". - Black Sea National University named after Petro Mohyla, Nikolaev, 2019.

Thesis is devoted to the development and implementation of software implementation of automatic generation of documentation to the code in Java .. Added materials with the implementation of the main functions. The view of further development of the developed system and its possible modernization is presented.

Explanatory note to the thesis consists of the introduction, 43 sections, conclusions, annexes. The total volume of work is 73 pages, 5 figures, 2 add-ons and 28 references to literary sources. The special part of safety and civil defense is 11 pages.

Key words: automation, documentation, increase of efficiency, web-application.