НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Радіотехнічний факультет
(повна назва інституту/факультету)

Кафедра радіоконструювання та виробництва радіоапаратури
(повна назва кафедри)

«На правах рукопису»
УДК 004.42

«До захисту допущено»
Завідувач кафедри
_____ Є.А. Нелін
(підпис) (ініціали, прізвище)
"14" грудня 2018 р.

Магістерська дисертація

за спеціальністю 172 Телекомунікації та радіотехніка
за спеціалізацією Інтелектуальні технології мікросистемної радіоелектронної техніки
(код і назва спеціальності)

на тему: Система отримання, накопичення та обробки відгуків користувачів інтернет-сервісів

Виконав (-ла): студент (-ка) 6_ курсу, групи РІ-71мп
(шифр групи)
Силенко Євген Євгенійович _____
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник к.т.н. доц. Дюжаєв Л.П. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з охорони праці к.т.н., доцент Каштанов С.Ф. _____
(назва розділу) (науковий ступінь, вчене звання, , прізвище, ініціали) (підпис)

Рецензент асистент, PhD Сумко О.Ю. _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018 року

Національний технічний університет України
«Київський політехнічний інститут
імені Ігоря Сікорського»

Факультет (інститут) радіотехнічний факультет
(повна назва)

Кафедра радіоконструювання та виробництва радіоапаратури
(повна назва)

Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою
За спеціальністю 172 Телекомунікації та радіотехніка
За спеціалізацією Інтелектуальні технології мікросистемної радіоелектронної техніки (код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Є. А. Нелін
(підпис) (ініціали, прізвище)
« 30 »вересня 2017р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Силенку Євгену Євгенійовичу
прізвище, ім'я, по батькові)

1. Тема дисертації Система отримання, накопичення та обробки відгуків користувачів інтернет-сервісів.

науковий керівник дисертації Дюжаєв Л.П. к.т.н. доцент ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від « 6 »листопада 2018 р. № 4093с

2. Строк подання студентом дисертації 30 листопада 2018 року

3. Об'єкт дослідження Система отримання, накопичення та обробки відгуків користувачів

4. Предмет дослідження (вихідні дані для магістерської дисертації за освітньо-професійною програмою) аналіз функціональних можливостей створеної системи

5. Перелік завдань, які потрібно розробити 1) розробити алгоритми та моделі даних для функціонування системи отримання, накопичення та обробки відгуків користувачів; 2) розробити програму та графічний інтерфейс; 3) розробити інтернет-ресурс та перевірити його працездатність; 4) дослідити функціональних можливості створеної системи; 5) розробити стартап-проект

6. Орієнтовний перелік ілюстративного (графічного) матеріалу мультимедійна презентація

7. Орієнтовний перелік публікацій Силенко Є.Є. „Система отримання, накопичення та обробки відгуків користувачів інтернет-ресурсів". Студентський науковий семінар „Наукові розробки РТФ'2018"

8. Консультанти розділів дисертації*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
З охорони праці	Каштанов С.Ф., доцент		

9. Дата видачі завдання 30 вересня 2017 року

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	підбір джерел та виконання огляду	22.09.17 – 13.12.17	
2.	Аналіз аналогів	13.12.17 – 10.01.18	
3	розробка алгоритмів та моделей даних	10.01.18 – 20.01.18	
4	розробка попередньої версії ресурсу та тестування	20.01.18 – 26.03.18	
5	доопрацювання системи, створення інтернет-ресурсу	26.03.18 – 26.07.18	
6	створення презентації та оформлення	26.07.18 – 1.10.18	

Студент _____ (підпис) Силенко Є.Є. (ініціали, прізвище)

Науковий керівник дисертації _____ (підпис) Дюхаєв Л.П. (ініціали, прізвище)

* Консультантом не може бути зазначено наукового керівника

РЕФЕРАТ

Магістерська дисертація складається із пояснювальної записки обсягом 111 сторінок та включає 28 ілюстрацій, 6 таблиць, 30 посилань, 2 додатки.

Ключові слова: сервіс зворотного зв'язку, відгуки користувача, інтернет-ресурс, обробка відгуків.

Актуальність теми. В останні роки конкуренція в інтернеті зростає неймовірно швидко. Для того щоб бізнес розвивався і був кращим за конкурентів наразі недостатньо мати товар або надавати послуги. Відгуки користувачів можуть мати для бізнесу більшу цінність ніж власні амбіції та бажання власника бізнесу, і саме тому важливою та актуальною задачею є отримувати, накопичувати та оброблювати відгуки користувачів.

Метою магістерської дисертації є збільшення у вільному доступі відгуків користувачів про інтернет-ресурси, зменшення необхідних ресурсів для обробки відгуків та пришвидшення процедури обробки.

Об'єктом дослідження є система отримання, накопичення та обробки відгуків користувачів (далі ресурс, система).

Предметом дослідження є аналіз функціональних можливостей створеної системи.

Наукова новизна отриманих результатів полягає у додаванні нових можливостей для отримання відгуків користувачів інтернет-ресурсів за допомогою інтеграції сервісу на власний ресурс.

Практичне значення одержаних результатів. Розроблений сервіс рекомендовано для використання в інтернет-ресурсах які прагнуть більшої задоволеності користувачів від свого продукту.

ABSTRACT

The master's dissertation consists of an explanatory note on the volume of 111 pages and includes 28 illustrations, 6 tables, 30 links, and 2 appendices.

Keywords: feedback service, user feedback, internet resource, feedback processing.

Actuality of theme. In recent years, online competition has grown incredibly fast. In order for business to develop and be better than its competitors, it is not enough to have goods or services. User feedback can have more value for business than its own ambitions and the wishes of the business owner, which is why the important and actual task is to receive, accumulate and process user reviews.

The purpose of the master's thesis is to increase the free access of user feedback about online resources, reducing the resources needed to process reviews and speed up processing.

The object of the study is the system for obtaining, storing and processing user reviews (hereinafter resource, system).

The subject of the study is an analysis of the functional capabilities of the system.

Scientific novelty of the results is to add new opportunities for receiving feedback from users of Internet resources by integrating the service into their own resources.

The practical value of the results. Developed service is recommended to use in Internet resources, which seek more customer satisfaction from their product.

ЗМІСТ

СПИСОК СКОРОЧЕНЬ

API — application programming interface — прикладний програмний інтерфейс

DNS — domain name system — доменна система імен

ПК — персональний комп’ютер

ВДТ — візуальними дисплейними терміналами

ПУЕ — Правила улаштування електроустановок

ВСТУП

На сьогодні інтернет-ресурси мають надзвичайно широкі можливості для покращення користувацького інтерфейсу, якості сервісу, збільшення аудиторії користувачів та інших важливих аспектів роботи. Конкуренція має глобальні масштаби і для того щоб користувач(клієнт, покупець) обрав саме конкретний інтернет-ресурс окрім ціни та/або якості послуги користувач також звертає увагу на велику кількість факторів. Ці фактори можуть бути не такими помітними для власника інтернет-ресурсу, так як він більше уваги приділяє самому ресурсу, його працездатності та надійності роботи. Неодмінно це є важливі фактори, але для користувача вони непомітні, до того часу як вони не проявляться у вигляді непрацюючого інтернет-ресурсу.

На жаль, користувачі, якщо не надано простого, зрозумілого та чіткого методу для зв'язку з власником інтернет-ресурсу не прагнуть до залишення свого відгуку. Більшість методів отримання відгуків є специфічними і підходять до певних конкретних ресурсів, в той час як для інших вони можуть приносити більше шкоди, ніж користі. Шкоду вони можуть наносити через те що користувач може мати певні упередження щодо обраного методу збору зворотного зв'язку. Наприклад люди можуть не мати бажання дзвонити на певні номери для залишення відгуку про інтернет-ресурс, навіть якщо даний метод є досить простим і швидким.

Інтеграція відповідного ресурсу також може підвищити кількість відгуків і тим самим суттєво допомогти ресурсу. Тому можливість інтеграції є необхідною складовою для такого ресурсу.

1 ПРИНЦИПИ ПОБУДОВИ СИСТЕМ ОТРИМАННЯ, НАКОПИЧЕННЯ ТА ОБРОБКИ ВІДГУКІВ КОРИСТУВАЧІВ ІНТЕРНЕТ-РЕСУРСІВ

Отримання зворотного зв'язку від користувачів є однією з найважливіших складових роботи будь якого сервісу. На теперішній час існує велика кількість сервісів які пропонують схожі послуги та мають схожі параметри. Саме тому саме зворотній зв'язок є тим що відрізняє хороший сервіс від поганого. Якщо автор ресурсу прислухається до своїх клієнтів то для такого сервісу є набагато більша можливість отримати прибуток з клієнту ніж у інших схожих сервісів.

Конкуренція грає велику роль в житті будь яких сервісів через те що створити новий сайт на сьогодні є дуже просто, і тому вибір користувача серед великої кількості сервісів які надають необхідні послуги паде на той сервіс який має найбільшу кількість позитивних відгуків та задоволених клієнтів. Такими відгуками не обов'язково можуть слугувати відгуки та коментарі в інтернеті, а можуть бути надані вашими родичами, друзями, знайомими. В такому випадку навіть одного поганого відгуку може бути достатньо щоб користувач обрав інший сервіс.

Доступні наразі засоби отримання зворотного зв'язку обмежені і часто відгуки неможливо подивитись іншим користувачам, якщо у користувача будуть сумніви щодо сервісу, що зменшує їх довіру та ймовірність того що вони будуть отримувати якісь послуги саме на цьому сервісі.

Сторонні сервіси які пропонують свій функціонал для отримання зворотного зв'язку мають суттєвий мінус: в великій більшості такі сервіси є платними, або пропонують дуже обмежений функціонал в безкоштовній версії. Запропонований в цій роботі сервіс є безкоштовним, надає схожі можливості та має можливість інтеграції до сторонніх інтернет-ресурсів і тому є кращим вибором для інших інтернет-ресурсів яким потрібно швидко та ефективно збирати зворотний зв'язок. Також варто зауважити

що така модель сервісу відгуків користувачів є новою для українського мережевого простору.

Розглянемо декілька прикладів систем отримання, накопичення та обробки відгуків користувач інтернет-ресурсів.

1.1 Reformal.ru

Ця система працює з 2008 року і до 2010 року був платним, проте є найближчим аналогом до запропонованої через те що вона так само дозволяє надавати відгуки, проводити голосування за відгуки та проводити інтеграції в інші інтернет-ресурси за допомогою субдомену та віджету. Якісно цю систему відрізняє те що вона безкоштовна для звичайного використання, включаючи достатню кількість необхідного функціоналу.

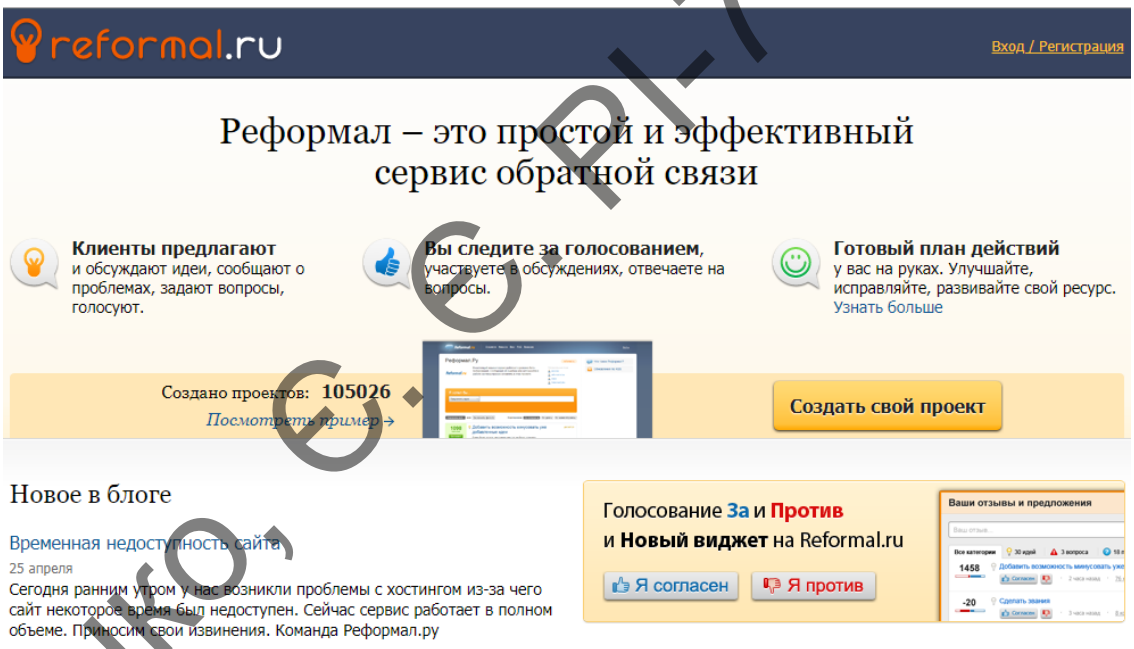

Рисунок 1.1 — Вигляд головної сторінки Reformal.ru

2014 року активний розвиток проекту вже закінчився, відповідно нового функціоналу або оновлення дизайну з того часу не відбувається. За 5 років кількість проектів зросла на 20 тисяч, коли за той самий період з початку існування сервісу було створено в 4 рази більше проектів. Проте навіть така відносно невелика цифра для певних інших систем може бути недоступною під час всього життя такої системи. Незважаючи на це багато проектів ще не бачать

конкурентних пропозицій і тому використовують цей ресурс в якості основного для отримання відгуків від користувачів.

1.2 Revizion

Сервіс який пропонує отримувати лише реальні відгуки від користувачів. За описом це забезпечується постійною перевіркою повідомлень за допомогою спеціалістів. Також це може забезпечуватись тим що немає прямої можливості надіслати відгук без необхідної інформації, тобто, наприклад, посиланням яке розташовується в закладі який працює з цією системою. Інша можливість залишити відгук – перейшовши за посиланням яке надходить у відповідному повідомленні на пошту після отримання замовлення в інтернет-магазині. За допомогою таких шляхів система обіцяє тільки реальні відгуки, що збільшує якість відгуків, проте не дає надати відгук користувачу який з певних причин не зміг, або не захотів зробити замовлення в інтернет магазині або відвідати певний заклад.

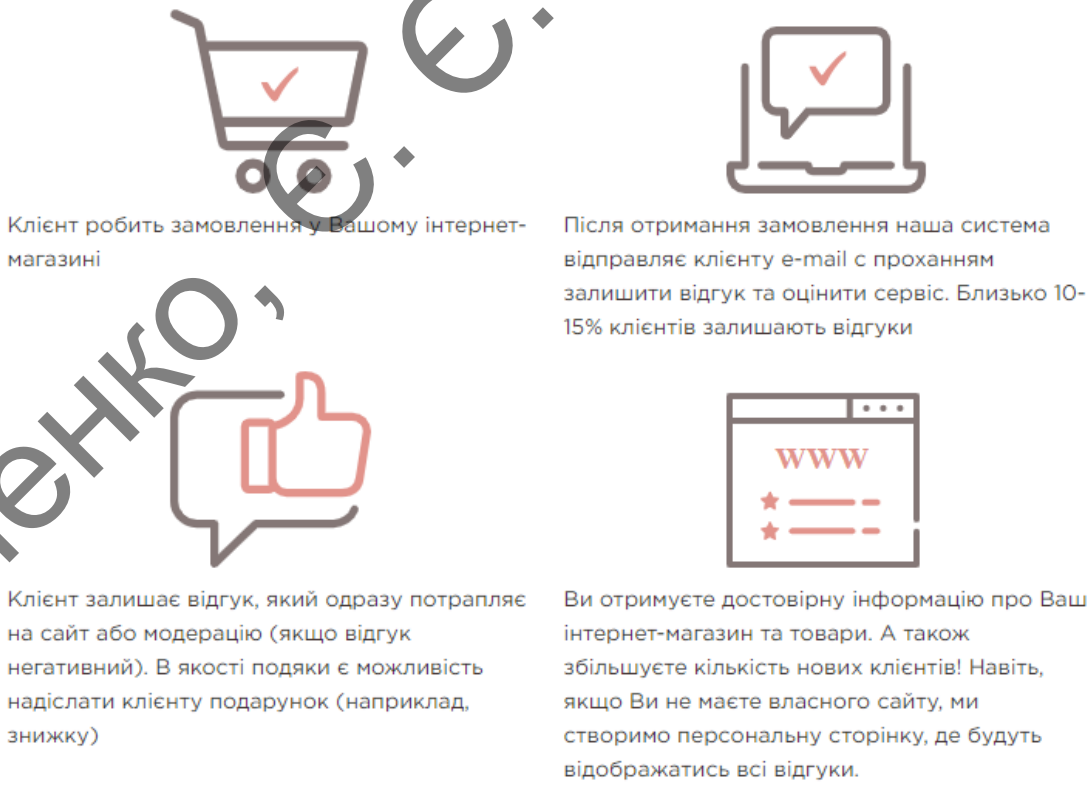

Рисунок 1.2 — Принцип роботи Revizion для інтернет-магазину

З іншої сторони система пропонує певну кількість унікальних можливостей для інтернет-ресурсів які підлючені до них. Наприклад можливість надати подяку клієнту який надав відгук у вигляді знижки на наступну покупку. Також сервіс надає аналіз результатів, тобто без додаткової оплати сервіс надає спеціалістів які дадуть відповідь на будь яке питання по сервісу або нададуть відповідні результати аналізу даних. Цікавою можливістю є реклама від блогерів з якими співпрацює сервіс. Тобто певні блогери можуть надати відгук про ваш заклад або інтернет-магазин та додатково його прорекламувати, що не пропонує більше жоден з інших розглянутих сервісів.

Ці можливості надаються системою за певну платню, проте для державних організацій цей сервіс є безкоштовним.

1.3 UserEcho

Система, яка більше направлена на амереканський ринок через те що вона не така відома за її межами. Як і більшість аналогів є платним сервісом, проте на відміну від інших має досить великий список можливостей які не обмежуються отриманням відгуків.

До таких можливостей відноситься створення форуму запитань і відповідей, теги, категорії, створення опитувань, можливість чатів в реальному часі та достатня кількість інтеграцій, включно з прикладним програмним інтерфейсом (API). Через достатньо низьку ціну в порівнянні з конкурентами та велику кількість можливості цей сервіс є кращим для інтернет-ресурсів, і тому має таких клієнтів як SublimeText, Instagrille та Anturis.

З унікальних можливостей які надаються цією системою можна віднести чат в реальному часі між клієнтом та співробітником інтернет-ресурсу та моніторинг соціальних мереж. Таким чином якщо користувач надасть відгук в одній з соціальних мереж він може бути автоматично доданим на сайт цієї

системи і після відповіді на цей відгук він так само автоматично буде доданий як відповідь до відповідного повідомлення в соціальній мережі.

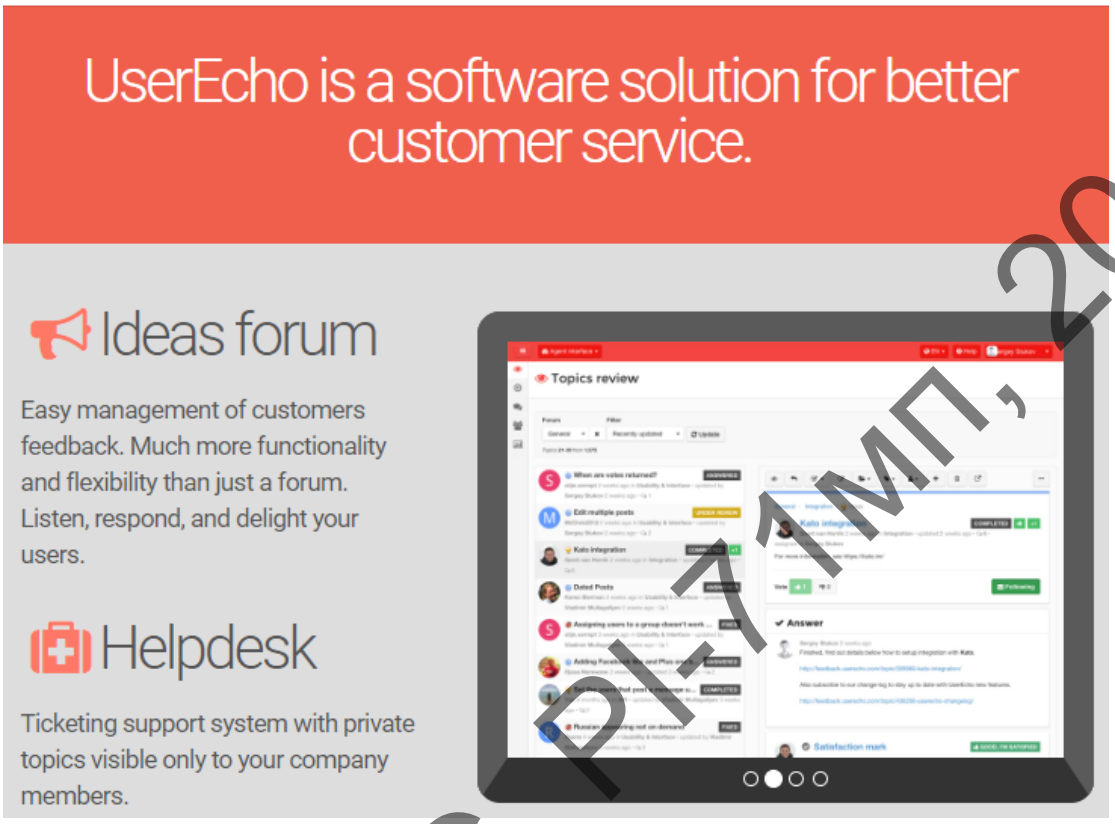

Рисунок 1.3 — Головна сторінка UserEcho

Зважаючи на досить велику кількість інтеграцій, віджет та доступність у вільному доступі відгуків та їх створення – система UserEcho є досить хорошим вибором ще й через те що вона надає можливість давати відгуки ще з доступного функціоналу є створення форумів, баз знань та опитувань, що є хорошим доповненням до можливостей інтернет-ресурсу що вже має підписку на подібний сервіс. А можливість для створення відгуків за допомогою авторизації через інтернет-ресурс з існуючими там користувачами та можливість живого чату з клієнтами ще більше спрощує використання цього ресурсу.

2 ОСОБЛИВОСТІ МОВИ ПРОГРАМУВАННЯ

Проект буде розроблятись з використанням мови програмування Golang. Ця мова була вибрана через свою достатню простоту, достатню широкість можливостей, можливість їх розширення через пакети, велику підтримку серед користувачів, високу надійність та швидкодію.

Легкість мови дозволяє досить швидко почати на ній писати, а доступні можливості значно полегшують розробку складних сервісів.

2.1 Синтаксис, пакети, структури, утиліти

Синтаксис. Синтаксис мови програмування Golang потребує закінчувати рядок крапкою з комою, проте через вживання певних правил компілятор автоматично їх розставляє [14]. Є два правила які описуються в специфікаціях мови, після знаходження яких вставляється крапка з комою:

- Якщо останній токен є типом даних, числом будь якого типу, символом, або рядком тексту, ключовими словами break, continue, fallthrough або return, одним з операторів або пунктуацією [15] ++, --,),], }
- Щоб дозволити складні однорядкові вирази – перед символами закриття “)” або “}”

В інакшому синтаксис цієї мови схожий з усіма С-подібними мовами і навіть людина яка немає досвіду у цій мові програмування може зрозуміти більшість коду без потреби в додатковій інформації.

Пакети. Певний набір файлів можуть містити одну ціль, бути пов’язані між собою або мати декілька варіантів реалізації інтерфейсів, такі файлі можуть мати спільну назву пакету, для того щоб об’єднати всі функції, структури та змінні в одну групу, яка буде мати назву імені пакету.

Це дозволяє більше розділяти код, використовуючи, наприклад, композицію. Також таке розділення дозволяє використовувати в коді тільки те, що потрібно, таким чином зменшуючи кодову базу програми.

Доступність пакетів для використання в коді забезпечується їх імпортуванням. Імпортування може мати два вигляди: імпортування пакету з стандартної бібліотеки або відносного шляху до коду. Тоді імпортування має наступний вигляд:

import "math"

Тут пакет знаходиться в списку стандартних пакетів і тому не потрібно використовувати ні префіксів, ні адрес. Другий вигляд – імпорт сторонніх пакетів які знаходяться в інтернеті. Таке імпортування має наступний вигляд:

import "github.com/go-sql-driver/mysql"

Про цей пакет можна сказати що він був завантажений з домену github.com, а остання тека в цьому пакеті – mysql. Утиліта go-get дозволяє завантажувати пакети з інтернету, де вони зберігаються в якійсь системі контролю версій(VCS). В цьому випадку можна сказати що для скачування цього пакету використовувалась VCS під назвою git.

Директорія в якій міститься пакет не завжди може називатись за назвою пакету, і мовою програмування дозволено задавати «аліаси», тобто синоніми, пакетів. Такі синоніми дозволяють задати іменування яке потрібне для конкретного програмного продукту, або для задання більш інтуїтивної назви.

Імпорт який задає синонім виглядає наступним чином:

import driver "github.com/go-sql-driver/mysql"

В цьому прикладі імпортуючи пакет з директорії github.com/go-sql-driver/mysql в коді на нього посилання будуть починатись з назви driver. Це дозволяє використовувати пакети з різних ресурсів, з різним функціоналом, проте з однаковими назвами пакетів, або задати ім’я пакету якщо воно не співпадає з назвою теки в якій він знаходиться для зручності майбутнього використання.

Спеціальним випадком, який є підвипадком синонімів для імпортів, є імпортування пакетів тільки для «побічних ефектів». Під терміном «побічні ефекти» мається на увазі що пакет не обов’язково несе якийсь корисний функціонал який користувач може використовувати, а лише виконувану функцію під назвою init і можливо набір внутрішніх станів та

змінних які будуть виконані під час запуску програмного забезпечення. Це досить часто використовується для того щоб ініціалізувати потрібні зміні при запуску програми. У випадку якщо пакет має створити лише побічний ефект то він імпортується з синонімом «_», тобто нижнім підкресленням:

```
import _ "github.com/go-sql-driver/mysql"
```

При такому імпортуванні пакет може може мати побічні ефекти, проте він не доступний для використання напряму користувачем.

Імпорт пакету який містить побічні ефекти не обов'язково мають мати нижнє підкреслення як синонім. У випадку якщо пакет має як корисний користувачеві функціонал та побічні ефекти користувач може також використати стандартний імпорт, або імпорт з будь яким іншим синонімом. Тоді побічні ефекти також будуть мати місце, але при цьому користувач зобов'язаний тоді використовувати цей пакет в коді програми.

Імпортовані пакети обов'язково мають бути використані, в інакшому випадку програма не скомпілюється. У випадках коли користувач розроблює програмне забезпечення і потреба на певний час у вибраному пакеті зникає – користувач може використати нижнє підкреслення як синонім при імпорті, що дозволить залишити цей імпорт, але не потребуватиме його використання.

Мова програмування Golang вже надає певний перелік доступних пакетів для того щоб пришвидшити розробку та виконання типових задач. Наприклад в стандартній бібліотеці є пакети для роботи з веб-ресурсами, для роботи з математичними операціями, файловою системою, криптографією та іншими речами.

Підхід розділення пакетів та використання «побічних ефектів» можна побачити в пакеті database/sql. При імпорті цього пакету також необхідно імпортувати драйвер, який і буде мати реалізацію комунікації з певним типом баз даних. Приклад такого використання може мати наступний вигляд:

```
package main

import (
  "database/sql"
```

```
  _ "github.com/go-sql-driver/mysql"
)

var driver string = "mysql"
// Змінна dataSourceName заповнюється або при компіляції,
// або іншими функціями пакету
var dataSourceName string

func main() {
  db, err := sql.Open(driver, dataSourceName)
  // Використання змінних db та err опущенно для простоти
}
```

Тут імпортування пакету database/sql потрібне для того щоб надати потрібні інтерфейси для роботи з базою даних, проте вона не має їх конкретних реалізацій. Наступний імпортований пакет, а саме "github.com/go-sql-driver/mysql" вже містить конкретну реалізацію та через побічні ефекти реєструє її.

Існує велика кількість пакетів які використовують побічні ефекти, і для реалізацій протоколів баз даних існує окремий список [16] в якому можна знайти реалізацію для конкретної бази даних. Також до широковідомого випадку використання побічних ефектів є пакет net/http/pprof [17] який використовується для профілювання додатків які працюють як веб сервіси і не тільки. Якщо використовується стандартний пакет для створення http-серверу то потрібно лише імпортувати пакет з нижнім підкресленням у якості синоніму і тоді додаток матиме велику кількість корисної інформації при переході на певну адресу.

Структури. В мові програмування Golang немає класів як в іншим об’єктно орієнтовних мовах, проте найближчою заміною є структури. Структури це основний структурний блок для програм в цій мові і на їх можливостях зосереджено досить багато уваги.

Структури як і класи мають внутрішній стан, вони так само мають методи якими вони можуть змінювати внутрішній стан або зовнішні

змінні. На відміну від мови програмування C++ у структур немає конструкторів і деструкторів що деяк спрощують розробку програмного забезпечення, проте ці речі можна замінити за допомогою інших патернів та внутрішніх можливостей мови. Наприклад деструктор можна замінити викликом методу runtime.SetFinalizer, в яку передається об'єкт та функція, яка буде викликатись при знищенні відповідного об'єкту. Проте використання такого методу є поганою практикою серед розробників.

Структури можуть містити поля всіх доступних типів, включно з іншими структурами. Наприклад об'явлення структури під назвою Main та властивостями Age та Name з типами int та string, тобто числовим та рядковим типами матиме наступний вигляд:

```
type Main struct {
  Age int
  Name string
}
```

Потрібно зауважити що на відміну від певних інших мов, наприклад похідних від C, в мові програмування Golang немає модифікаторів доступів до методів та властивостей. Зараз їх реалізація буде опущена та розглянута в розділі 3.5, пункті 1.

Тепер в функціях і методах де доступна змінна з типом цієї структури будуть доступні поля Age та Name, які можна викликати через крапку. Таким самим чином можна отримати доступ і до методів структури.

Методи у структур існують до запуску програм, тобто ми не можемо під час роботи програми створити у опису структури, або існуючого об'єкту новий метод. Методи описуються в тому ж пакеті що і визначена структура. Це означає що не обов'язково описувати всі методи в одному файлі, проте це буде гарним тоном.

Мова програмування Golang передає аргументи в функції та методи за значенням, якщо не вказано іншого. Тобто при кожному виклику функції будуть створені копії аргументів, які вже будуть передаватись у функцію. Таким чином якщо використовувати функції або методи які приймають аргументи по значенню – вони не зможуть змінювати значення

цих аргументів так, щоб це було видно ззовні. В цьому правилі є винятки, проте наразі розглядати їх не будемо.

Також метод від функції відрізняється тим, що у методу є «ресівер», тобто об'єкт який приймає виклик. Цей параметр може бути одним з двох для кожного окремого методу. Ресівер вказується перед сигнатурою функції і як і аргументи може бути як значенням, так і вказівником. Ресівер також має ім'я, яке в С-подібних мовах замінює *this*, а в мові програмування *Python* замінює загальноприйнятий аргумент *self*, який існує в сигнатурі функції, проте при виклику такої функції вказувати цей параметр непотрібно: він передається неявно.

Ресівер в мові Golang прийнято називати однією літерою (рідше декількома літерами і ще рідше словом), часто це може бути першою літерою назви типу.

Розглянемо визначення методу під назвою Hello, який буде повертати рядок для вищезазначеної структури Main:

```
func (m Main) Hello() string {
  return "Hello! My name is" + m.Name + " and I am " + m.Age + " years old!"
}
```

Метод, як і функція, мають починатись з ключового слова *func*, а далі опис методів та функцій різняться. Для методів наступним параметром обов'язково має бути ресівер, який пишеться в дужках. Він може мати ім'я та обов'язково тип. Функція відрізняється тим що не потребує ресіверу. В даному прикладі ім'я ресіверу, яке можна буде використовувати всередині методу є *m*. В даному випадку тип ресіверу є типу *Main*. Про певні особливості ресіверів буде розказано у розділі 3.5, пункті 3.

В цьому прикладі всередині методу ми використовуємо властивості об'єкту типу *Main* для того щоб повернути рядок в якому буде використані властивості ресіверу.

Іншою корисною можливістю є «вбудовування» (*embedding*) [18] інших структур. Ця можливість дозволяє розмежовувати структури та інтерфейси розділяючи їх можливості. В мові програмування Golang є вислів: «чим більший інтерфейс, тим слабша абстракція». Цей вислів

означає що краще мати декілька інтерфейсів які слугують різним цілям, ніж мати один великий інтерфейс, який робить всі потрібні речі. Вбудовування дозволяє притримуватись цього виразу і робити малі інтерфейси, і за потреби робити з них більші. Розглянемо на прикладі стандартних інтерфейсів Reader та Writer.

Інтерфейс Reader має всього один метод всередині який має назву Read. На вхід ця функція приймає масив байтів та повертає кількість прочитаних байтів та помилку, якщо така є. Виглядає описаний інтерфейс наступним чином:

```
type Reader interface {
  Read(p []byte) (n int, err error)
}
```

Цей інтерфейс показує що структура яка його задовольняє має мати метод Read з вказаною вище сигнатурою.

Інтерфейс Writer є схожим, і навіть має схожу сигнатуру:

```
type Writer interface {
  Write(p []byte) (n int, err error)
}
```

Тут так само структура яка хоче задовольняти цей інтерфейс має мати метод Write з вказаною вище сигнатурою.

Але якщо потрібно щоб структура мала обидва методи і таким чином задовольняла обидва інтерфейси можна зробити ще один інтерфейс, який буде складатись з інших двух вищезазначених методів. Щоб це зробити маючи описані вище інтерфейси потрібно зробити ще один яким матиме наступний вигляд:

```
type ReadWriter interface {
  Reader
  Writer
}
```

Описаний інтерфейс немає всередині явних методів які мають бути описані в структурі, проте є вбудовані два інших інтерфейси. Тепер коли потрібно щоб структура описувала обидва інтерфейси Reader та Writer

одночасно вона має описати методи Read та Write. Тоді така структура буде задовольняти одразу трьом інтермейсам: Reader, Writer та ReadWriter.

Утиліти. Через неймовірну широкість можливостей утиліт які входять до складу мови Golang розглядатись зараз будуть лише найбільш часто вживані утиліти та їх параметри.

До найбільш часто використовуваних утиліт можна віднести наступні:

- get
- fmt
- build
- run
- vet
- test
- env

Кожна з цих утиліт є частиною команди go, яка стає доступною після встановлення середовища Golang.

Go get дозволяє завантажити необхідні пакети з мережі інтернет якщо вони недоступні локально. Це відбувається за наявності певних систем контролю версій, таких як git. Також може бути використаний Mercurial, svn та інші, проте вони не так часто використовуються як git.

Go fmt почергово переходить по файлам у вказаних пакетах та виправляє помилки форматування, а точніше надає коду єдиного формату, який запропонований з мовою програмування Golang. Наприклад до таких виправлень належить заміна пробілів на знак табуляції для відступів. Завжди рекомендується використовувати цю утиліту, адже це значно спрощує майбутню підтримку та надає єдину стилістику, незалежно від автору коду.

Go build дозволяє компілювати пакет або окремі файли у виконуваний файл, який надалі можна використовувати без встановленого середовища Golang. При компіляції можна задати ім’я результуючого файлу та вказати чи потрібно рекомпілювати всі складові, чи можна використати кеш.

Go run дозволяє запускати файл або пакет з необхідними опціями без попередньої явної компіляції. Також можна передати аргументи які використовуються для компіляції.

Go vet шукає відомі помилки або неправильні частини коду і показує це. Також може приймати пакет, окремий файл або список файлів.

Go test виконує тести які існують для пакету за певними назвами або всі існуючі в цьому пакеті або файлі. Дає також можливість отримати кількість коду у відсотковому відношенні який охоплений тестами. Також використовується для того щоб провести тести на швидкодію.

Go env виводить всі або конкретні значення змінних оточення середовища Golang в якому запускається ця утиліта.

2.2 Особливості мови програмування Golang

2.2.1 Модифікатори доступу

Доступність методів та властивостей визначається за допомогою їх назв. Якщо назва починається з великої літери – такий метод або властивість є експортованими, тобто їх можуть використовувати інші методи та функції, якщо ж назва починається з маленької літери то тоді цей метод або властивість будуть доступні лише всередині структури та її методів.

Такий самий підхід і з глобальними змінними, проте тоді якщо змінна починається з маленької літери – вона буде доступна лише в пакеті, в якому вона була визначена.

2.2.2 Функція init

Досить часто потрібно перед основним виконанням надати значення якимось змінним, або отримати значення флагів які передані програмі. Або є пакет, в якому потрібно реалізувати побічний ефект.

В такому разі потрібно використовувати функцію init, яка викликається перед функцією main. Ця функція викликається відповідно до імпорту пакетів і вона може міститись в одному або кількох файлах всередині пакету.

Використання цієї функції може бути досить різним в залежності від ситуації і потреби, проте в більшості випадків вона використовується для трьох основних речей:

- Ініціалізація глобальних змінних
- Отримання значень флагів переданих в програму
- Створення побічних ефектів

Ця функція є досить корисною, проте не потрібно покладати на неї всі операції щодо ініціалізації програми. Інші речі що не описані вище краще щоб використовувались у функції main, як основний блок програми.

2.2.3 Ресівер, аргументи та значення які повертаються

Ресівери – спеціальні змінні, які визначають об’єкт для якого будуть виконуватись методи визначені для певного типу даних. Ресівери мають імена і в більшості випадків вони складаються з однієї літери яка описує тип даних. У випадках коли всередині методу не використовується ресівер його можна не називати.

Методи можуть виконуватись на двох типах ресіверу: за значенням та за посиланням. Коли використовується метод з ресівером за посиланням тоді створюється копія змінної для якої викликається метод і для цієї копії вже викликається відповідний метод. Саме тому методи які змінюють внутрішній стан об'єкту мають мати ресівер за посиланням. Гарною практикою вважається обрання одного типу ресіверу для всіх методів для певного типу даних. Навіть якщо існує сукупність методів серед яких є такі що змінюють внутрішній стан та таких що не змінюють то потрібно використовувати для всіх методів ресівер за посиланням.

Аргументи можуть розділятись за методом передачі на три групи: ті що передаються за значенням, за посиланням та варіадивні. Аргументи які передаються за значенням та посиланням мають такі самі властивості як і ресівери які вказані відповідно за значенням та за посиланням. Варіадивні аргументи є більш унікальними: насамперед таких аргументів може бути всього один на функцію, і такий аргумент має бути останнім в списку аргументів. Варіадивний аргумент вказується однією назвою, проте в середині функції таки аргумент може мати від нуля до безлічі значень так

як всередині функції такий аргумент перетворюється на «слайс», тобто масив без попередньо заданої довжини. Варіадивний аргумент може приймати безліч значень, проте всі вони мають бути одного типу даних.

Наприклад вигляд варіадивної функції Println зі стандартного пакету fmt, яка друкує передані значення в консоль, має наступну сигнатуру:

func Println(a ...interface{}) (n int, err error)

Тут аргумент a є варіадивним, і відповідно функція також є варіадивною.

2.2.4 Інтерфейси

Golang дозволяє створювати інтерфейси та легко їх задовольняти в структурах, проте інтерфейси мають більше можливостей ніж просто описувати необхідну поведінку.

Для початку для того щоб тип даних задовольняв певний інтерфейс йому потрібно лише створити методи з необхідними сигнатурами, які описані в інтерфейсі. Це є особливістю мови і таким чином навіть тип даних який не мав на меті задовольнити певний інтерфейс може неявно це робити.

Гарним тоном є приймати в методи та функції не конкретний тип даних а інтерфейс, який повинен містити всі необхідні методи, проте нажаль це не завжди можливо, а деколи цього можуть не робити не знаючи цього, або просто вважаючи що це не потрібно. Насправді це не є правилом і якщо архітектура програми з самого початку не була розроблена згідно з такою концепцією – перевести методи та функції може бути непросто.

В загальному перевірка на те чи впроваджує тип даних потрібні методи виконується на етапі компіляції, проте є ситуації коли є перетворення одного інтерфейсу на інший і в таких ситуаціях на момент компіляції помилок видано не буде, якщо навіть конкретний тип який використовується в такому шматку коду не задовольняє необхідний інтерфейс.

Особливою ситуацією є використання порожнього інтерфейсу. Такий інтерфейс не описує жодної функції і відповідно всі типи даних неявно

задовольняють його. Програмісти часто використовують це коли тип даних може бути невідомим завчасно, або коли потрібно передати дані різного типу в одному об'єкті. Це є поганою практикою, та через широкість застосування такого підходу та неможливості використовувати узагальненні [19] типи даних програмісти не мають іншого виходу.

2.2.5 Горутини

Горутини є суттєвим фактором при виборі мови програмування для інтернет-ресурсів та інших програм які мають багато роботи яку можна виконувати паралельно. Горутини замінюють потоки у контексті операційної системи, та мають маленькі потреби по пам'яті. Програмісту не потрібно ними керувати, проте зловживання ними є ознакою поганого програмування, коли в одній функції є декілька створень горутин без необхідної потреби в цьому. Запуск горутин є дешевшим ніж створення нових потоків або тим паче процесів.

При всіх плюсах горутин неправильне їх використання може призвести до сповільнення програми, або навіть до критичних помилок. Також потрібно зауважити що горутини не повертають значеть. Тобто якщо функція, яка запускається в горутині, має значення які повинні повертатись то потрібно вибрати спосіб в який вони будуть це робити. Часто для цього можуть використовуватись канали.

Запуск нової горутини відбувається дуже просто: перед функцією яка має виконуватись у новому потоці потрібно поставити ключове слово go:

```
go DoSomething()
```

Після доходження програми до такого рядку функція DoSomething буде запущена у фоні, а виконання одразу перейде на наступний рядок.

Такий підхід ще використовується коли потрібно періодично оновлювати якісь дані у фоні.

2.2.6 Канали

Канали в більшості випадків слугують для передачі даних між горутинами, синхронізації між горутинами та як буфер.

Канал як метод передачі даних між горутинами часто використовується через те що це дуже просто, покращує читабельність коду, може без додаткових синхронізацій використовуватись з різних горутин та можна досить просто керувати передачею. Є ситуації коли потрібно повідомити назовні що більше даних вже не буде, і в таких випадках добре підходять канали, так як їх можна закрити щоб повідомити що більше даних вже не буде.

Канали можуть бути двох типів: буферизовані та небуферизовані.

Канали в яких немає буферу є прямим шляхом передачі даних між горутинами або змінними які потребують синхронізації. Такі канали можуть також забезпечувати очікування даних або якогось процесу, як для отримання, так і для передачі. Тобто такий канал може слугувати синхронізацією між двома процесами які мають закінчитись одночасно, проте один з них виконується швидше.

Тобто такі канали блокують горутину в якій виконується передача, якщо з іншої сторони немає приймачу, так само якщо є читання з каналу, проте не відбувається відправка даних в канал – горутина в якій читається канал буде заблокована до того часу поки не будуть відправлені дані в канал.

Буферизовані канали, на відміну від небуферих, мають певну ємність. Ця ємність, допоки вона повністю не заповнена, не блокує надсилання даних в канал. Якщо відбувається читання з буферизованого каналу то допоти в каналі є непрочитані дані то канал який читає не блокується, проте коли в каналі більше немає даних – горутина блокується до надходження нових даних.

3 АЛГОРИТМИ ТА МОДЕЛІ ДАНИХ СИСТЕМИ, ЩО РОЗРОБЛЮЄТЬСЯ

Зважаючи на доступні дані про збір зворотного зв'язку можна отримати певні алгоритми при взаємодії користувача з інтернет-ресурсом. Ці алгоритми можна розділити на категорії по сутностям, наприклад:

- Користувач
- Проект
- Відгук
- Коментар

При даному розподілі можна розділити межі відповідальності та встановити потрібні алгоритми при кожній дії сутності. Для користувача є властивим входити в персональний акаунт, виходити з нього, створювати проекти, відгуки, коментарі. Дані моделі є найважливішою частиною програмного забезпечення, адже саме засновуючись на їх параметрах можна створювати алгоритми та виконувати будь які дії. Тому саме зміни моделей дозволяють оновлювати алгоритми та додавати нові дії.

Розглянемо конкретні приклади для отриманих вище категорій.

3.1 Користувачі

Найпростішими потребами в будь якому інтернет-ресурсі які містять користувачів є можливість їх входу та виходу. Також у кожного користувача має бути унікальний ідентифікатор та можливість увійти на ресурс за допомогою секретної фрази(паролю) або зовнішніх інтерфейсів, таких як вхід за допомогою соціальний мереж. Список не є вичерпним.

Перший метод є досить простим і найрозповсюдженішим методом авторизації користувачів через свою простоту і не потрібність використовувати стороні залежності. Автор ресурсу може контролювати всі етапи авторизації та змінювати їх за потребою. Це дозволяє наприклад

використовувати для авторизації користувачів різні комбінації параметрів, такі як логін та пароль, поштову адресу та пароль, їх комбаніції та інше.

Захищеність даних є важливою складовою ресурсу, який зберігає цю інформацію у себе. Ресурс має робити все щоб сторонні люди не змогли отримати доступ до даних користувачів. Для авторизації по паролю автор ресурсу не має зберігати його у відкритому вигляді, а лише після виконання хеш-функцій над ними. Це дозволить безпечно зберігати паролі, при цьому так, щоб була можливість перевірити наданий користувачем пароль на правильність, але не простим порівнянням.

Розглянемо деякі варіанти таких хеш-функцій.

Розрізняють криптографічно стійкі алгоритми, та нестійкі. Нестійкі функції досить рідко використовуються для хешування паролів через свої вразливості до підбору. Частіше за все такі функції використовуються для перевірки правильності отриманих даних, наприклад при скачуванні офісного пакету OpenOffice можна побачити посилання на результати хешування для функції SHA-1 різної довжини. Перейшовши за посиланням буде отриманий попередньо обрахований результат виконання функції для відповідного файлу який можна побачити на рисунку 3.1.

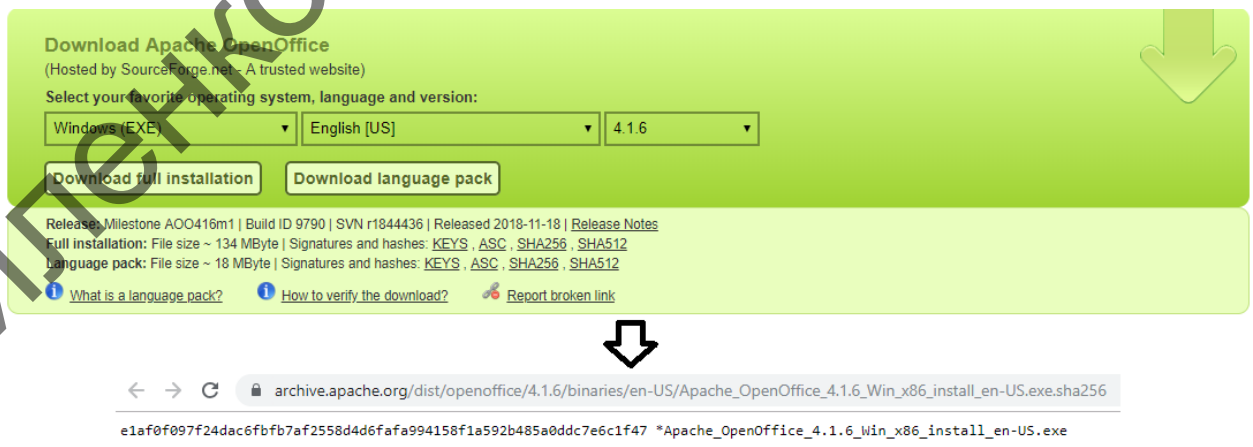

Рисунок 3.1 — Значення SHA-1 для скачуваного файлу

Таким чином виконавши хешування функцією SHA потрібної довжини можна порівняти результат і відповідно отримати висновок чи є завантажений файл оригінальним і чи отриманий він правильно.

Певні алгоритми мають також колізії, тобто отримавши на вхід два різних значення на виході буде отриманий один і той самий результат. Саме тому такі хешувальні функції ні в якому разі не можна використовувати для зберігання паролів користувачів.

Розглянемо можливі варіанти таких функцій.

MD5 [1] – алгоритм хешування розроблений у 1991 році, який був розроблений професором Рональдом Рівестом, що прийшов на заміну алгоритму MD4. Цей алгоритм часто використовується для підтвердження цілосності даних, адже алгоритм є дуже швидким. Проте через те що знайти колізію для цього алгоритму займає лічені секунди – цей алгоритм більше не використовується для важливих речей, таких як, наприклад, зберігання паролів.

SHA-1 [2] – криптографічний алгоритм отримання «дайджесту» повідомлення, тобто значення за яким можна перевірити оригінальність повідомлення (або файлу). Алгоритм розроблений Агентством національної безпеки США і з 2005 року не вважається безпечним для використання проти «гарно профінансованих» опонентів.

SHA-3 [3] – найновіший член сімейства алгоритмів хешування *Secure Hashing Algorithm*. Ця функція заснована на «ефекті губки», коли вона «всмоктує» вхідні дані, які при отриманні змінюються, і потім «вижимаються» після її виконання. Підхід значно відрізняється від попередніх версій алгоритму і через те що він показав найкращі характеристики на конкурсі криптографічних алгоритмів, на якому вибирали наступника SHA-2.

Дані функції добре підходять для отримання «дайджесту» повідомлення для його підтвердження, проте вони слабко підходять для

зберігання паролів через їх швидкість виконання та невибагливість до ресурсів що призводить до великої кількості паролів які можна перевірити за короткий проміжок часу або велику ймовірність отримати колізії значень. Тому для хешування паролів використовуються спеціальні функції які називаються «функції формування ключа» які мають певні вхідні параметри, окрім самого повідомлення, корегуючи які можна змінити складність виконання функції по часу та / або оперативній пам'яті. Через це складність таких функцій може варіюватись в залежності від потреб користувача і перебирати такі паролі вкрай складно.

Розглянемо декілька прикладів:

PBKDF2 [4] – розшифровується як Password-Based Key Derivation Function 2, прийшов на заміну стандарту PBKDF1, тому що перша версія приймала довжину ключів до 160 біт. Функція має змінну складність і навіть після того як була затверджена версія 2.1 та інші схожі функції все одно є рекомендованою до використання.

Bcrypt [5] – вперше запропонована у 1999 році і дотепер є стандартом у отримані криптостійких хешів паролів. Кількість ітерацій у функції можна змінювати за бажанням, тим самим збільшуючи потрібний час на її виконання, тим самим витримуючи підбори паролів навіть зі збільшеною потужністю комп'ютерів. Існує функція scrypt, яка збільшує використання оперативної пам'яті, що ще більше ускладнює процес підбирання паролів.

Argon2 [6] – алгоритм який отримав перемогу на конкурсі «Password hashing competition» в 2015 році. Його новизна є його проблемою, адже впевнитись в його надійності досить важко, зважаючи на недостатність проведених аналізів цієї функції. Наразі є три модифікації функції які відрізняються методом роботи.

У проекті буде використаний алгоритм bcrypt, який рекомендується використовувати якщо немає суттєвих причин цього не робити. Окрім того що він надасть достатню складність паролям його можна досить швидко

зробити складнішим у виконанні без суттєвих часових або людських затрат. Також ця функція виконується константний час, тобто на будь які вхідні дані для перевірки результату завжди потрібен один і той самий час.

Для мови програмування Golang існує декілька реалізацій цього алгоритму, але в проекті буде використовуватись версія з нестандартного репозиторію, але за розробки авторів Golang – crypto/bcrypt [7]. Пакет має простий інтерфейс, який складається з двох функцій які нас цікавлять: перша GenerateFromPassword у якої параметрами є пароль користувача у відкритому вигляді і складність функції. Друга функція – порівняння паролів яка називається CompareHashAndPassword, яка на вхід приймає два аргументи: перший пароль який був раніше захешований, і пароль у відкритому вигляді який надав користувач.

У даному пакеті є також допоміжна функція Cost, яка надає складність паролю. Її можна використовувати для того щоб оновити паролі у випадку коли потрібно підвищити складність функції. Це можна зробити дуже просто при авторизації користувача: якщо пароль вірний – перевірити складність паролю і якщо вона нижче певної межі – виконати хешування з потрібною складністю і зберегти результат.

При авторизації користувача з функціями які мають константний час виконання важливо щоб користувач, при невдалій авторизації, не зміг зрозуміти що саме стало причиною цього – чи пароль, чи ім’я користувача. Це потрібно для того щоб атакувальник не міг дізнатись правильність ім’я користувача, коли при неправильному паролі відмова у авторизації відбувається одразу. Тоді атакувальник напевне знає що користувач з таким ім’ям існує в системі і може далі продовжити або збирати імена користувачів, або ж підбирати пароль до існуючих. Тому в таких ситуаціях потрібно завжди робити перевірку паролю з будь чим, щоб атакувальник не зміг отримати інформацію про існування користувача і час авторизації був однаковим з будь яким набором даних.

Для входу на ресурс було обрано можливість входити через комбінацію логіну або електронної пошти та паролю. Це дозволить користувачам швидше авторизуватись на сайті завдяки тому що не потрібно вводити повну адресу, або згадувати ім’я користувача яке було спеціально використано для цього ресурсу.

Потрібно розуміти що через те що на ресурсі не має бути повторень електронних адрес, або нікнеймів користувачів, тому ці дані мають бути унікальними в базі даних. Це дозволить реалізувати авторизацію по логіну та електронній пошті, тому що не буде існувати двох користувачів з однаковими даними для авторизації.

Для швидкого доступу до певних рядків в базі даних використовуються унікальні ідентифікатори. Такими ідентифікаторами можуть бути наприклад слова, набір літер чи число. В більшій частині випадків використовується числове значення, адже сучасні бази даних дозволяють їх не вказувати, але при цьому вони автоматично збільшують номер запису для нових вставок в таблицю. Таким чином кожен новий запис в таблиці має своє унікальне значення, при цьому зі сторони користувача це відбувається прозоро. Для цього сервісу також будуть використовуватись числові ідентифікатори для таблиць.

Були описані мінімально необхідний набір полів для того щоб мати можливість авторизувати користувачів та мати унікальний ідентифікатор кожного з них. Проте, для того щоб мати більше можливостей на сервісі для користувача ми можемо отримати від нього додаткові дані, такі як його прізвище та ім’я, дату народження країну проживання та інші дані. Дата народження може знадобитись наприклад для того щоб відправити лист з поздоровленнями в потрібний день, а прізвище та ім’я для того щоб показувати на сайті саме їх, а не ім’я користувача, якщо користувач так захоче, або, наприклад, для того щоб підтвердити особистість, хоч це має

бути крайнім випадком, адже така інформація може бути легкодоступна через велику кількість професійних та непрофесійних соціальних мереж.

Так як ресурс має можливість відображатись певним набором мов для користувача було б корисно обирати його мову за замовчуванням коли він авторизується на сайті. Тоді навіть якщо він заходить з США то після авторизації сайт буде відображатись на його мові яку він вибрав за замовчуванням.

Зауважимо що користувач в базі даних має мати додаткову інформацію за якою програмне забезпечення та / або адміністратор зможе змінити запис користувача. Це необхідно, наприклад, коли було зареєстровано спробу авторизуватись з пристрою який раніше не був авторизованим. За цими додатковими даними можна заблокувати акаунт, або відіслати користувачу лист на пошту для того щоб перевірити чи справді вхід здійснює потрібна людина. До такої інформації також можна віднести якісь специфічні дані для користування ресурсом, наприклад кількість під'єднані акаунти соціальних мереж.

Отже приблизний список полів для моделі користувача можна назвати:

- Логін (нікнейм)
- Електронна пошта
- Пароль
- Прізвище та ім'я
- Дата народження
- Стать
- IP адреса з якої був зареєстрований користувач

Список полів не є закінченим, адже при розширені ресурсу кількість полів може змінюватись в будь якому напрямі. Потрібно зауважити що

через те що ресурс вже працює віддавати повний список полів є неправильним з точки зору безпеки даних користувачів і сервісу.

Щодо дій користувачів для них найбільш ймовірні дії це реєстрація, авторизація, скидання паролю та заміна персональних даних які вони надали сервісу.

Проста схема авторизації включає всього декілька кроків:

1. Отримання логіну та паролю від користувача
2. Пошук потрібного користувача по логіну в базі даних
3. Порівняння паролю спеціальними функціями
4. Вхід користувача

Ця і подальші схеми описуються за умови що кожен крок виконується успішно і з позитивним (очікуваним) результатом якщо не вказано інакшого.

Схема з більш безпечною авторизацією може виглядати складніше, але вона і потребує більше даних про користувача. Ця схема буде включати в себе додаткові кроки які потрібні для ситуацій коли користувач виконує авторизацію з невідомого місця (або навіть країни) чи пристрою:

1. Отримання логіну та паролю від користувача
2. Перевірка чи IP-адреса реєстрації користувача збігається з адресою з якої відбувається авторизація
3. Якщо не збігається перевірити чи є відповідні цифрові відбитки (fingerprints) користувача. Наприклад чи входив користувач раніше з такою комбінацією IP-адреса, агент користувача в браузері та, наприклад, час входу.
4. Якщо таких відбитків немає – відправити на пошту повідомлення з проханням перейти за посиланням або ввести тимчасовий код щоб підтвердити вхід

5. При переході за посиланням, якщо параметри які використовувались для авторизації та параметри при переході за посиланням збігаються – зберегти цифровий відбиток та виконати вхід користувачу

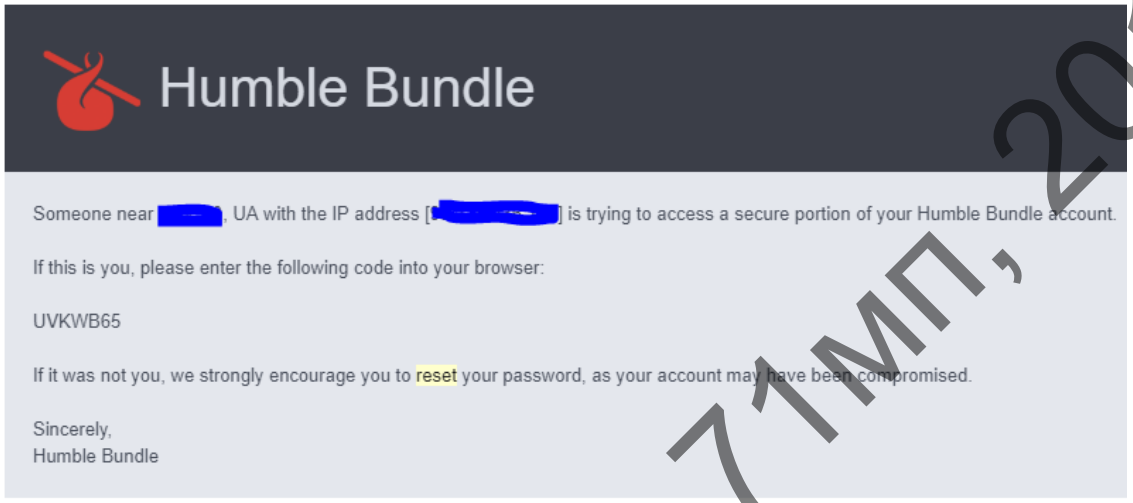

Рисунок 3.2 — Прохання ввести тимчасовий код

Таким чином забезпечується більш безпечна авторизація для користувача, яка не дозволить простим чином при заволодінні паролю третіми особами увійти до акаунту якщо, звичайно, користувач використовує різні паролі для даного та поштового сервісу.

Потрібно нагадати що в будь якому випадку, якщо навіть потрібний користувач не знайдений в базі даних потрібно робити перевірку паролю для того щоб по часу авторизації не можна було зрозуміти що саме неправильне: чи логін, чи пароль який був наданий користувачем.

Для реєстрації користувача алгоритм однаковий і малозалежний від вхідних параметрів користувача:

1. Отримати базовий набір даних таких як ім’я користувача (логін), пароль, електрону пошту та інші додаткові параметри
2. Перевірити вхідні дані на правильність
3. Переконатись що потрібного користувача за логіном чи поштую не зареєстровано

4. Додати до бази даних запис про користувача, при цьому виконавши хешування паролю

У відкритому вигляді пароль користувача в базі даних зберігатись не має.

Для більшого захисту модна використовувати шифрування або хешування даних, наприклад, для електронної пошти користувача та інших персональних даних. Це дозволить при отримані доступу до бази даних стороннім особам забезпечити безпеку персональних даних користувачів. Проте така практика немає широкого вжитку через свою складність та, в загальному випадку, непотрібність.


Welcome to Acapella!

Hello, [redacted]!

You have recieved this email because someone requested password reset on Acapella
If it wasn't you - please ignore or delete this message

To reset your account password - please tap on this link

Or paste it in your address bar manually: https://acapella.pp.ua/reset?rid=9&code=fc0f1c76-a26c-4742-8557-[redacted]

Рисунок 3.3 — Лист з проханням змінити пароль

Зміна пароля в більшості випадків може виконуватись трьома шляхами.

1. При успішній авторизації користувач в особистому кабінеті може вказати старий і новий пароль. Старий пароль потрібен для того щоб переконатись що зміну паролю запросив саме потрібний користувач, а не треті особи.
2. Користувачу на електрону пошту приходить посилання з тимчасовим кодом після переходу за яким користувачу буде одразу надана можливість встановити новий пароль.

3. Користувачу на пошту приходить тимчасовий пароль. Такий пароль замінює старий і після авторизації за тимчасовим паролем користувач може змінити пароль на новий.

3.2 Проекти

Проекти є основною частиною інтернет-ресурсу, адже саме завдяки ним він набуває потрібного значення та ваги для людей.

Кожен проект має мати певний набір основних даних, для того щоб він міг функціонувати. З необхідних даних у проекта це ім'я, внутрішня адреса та автор. Додаткові параметри можуть бути обширними через постійні доповнення функціоналу сервісу, проте серед загальних параметрів це можуть бути посилання на зовнішній ресурс, опис, іконка проекту та дата його створення.

Додавши декілька внутрішніх параметрів можна отримати такий неповний список:

- Назва
- Автор
- Дата створення
- Опис
- Іконка проекту
- Посилання на зовнішній ресурс
- Статус блокування
- Статус модерування

Через недобросовісність користувачів не варто довіряти всім даним які вони надають на проект, а тому модерація є важливим фактором для дотримання інтернет-ресурсу в гарному стані, при якому немає порушень щодо правил користування сервісу та інших норм та правил держави в якій працює цей сервіс.

Модерація може бути декількох виглядів: перевірка вхідних даних, наприклад коли користувач проводить реєстрацію потрібно перевірити чи доступний логін який обрав користувач, чи задовольняє певним параметрам наданий пароль, чи збігається він з повторним його введенням і так далі. Другий варіант це автоматична модерація коли наприклад

користувач створює коментар то сервіс самостійно визначає що коментар не відповідає нормам спілкування за допомогою даних та коментарів які були зареєстровані в системі раніше. Тобто такий варіант спершу розраховує на дані які першочергово існують в системі, наприклад в базі даних або файлі конфігурації і потім відбувається автоматичне або ручне «навчання», коли до існуючих даних постійно додаються нові для покращення результату модерації тексту. Третій варіант полягає в ручній модерації коментарів, відгуків, проектів та користувачів. В такому випадку людина має самостійно мати доступ до потрібних даних та вже вручну має виправляти, видаляти чи змінювати інформацію яку вона вважає за необхідне. Тоді доцільно ще може мати якусь історію змін які зробили люди, щоб можна було корегувати дії людей.

Сервіс робить перевірку вхідних даних та має можливість ручної модерації, проте не виключено що також буде застосоване автоматична модерація. Якщо буде працювати і такий варіант то це також скоротить потрібні людські ресурси на внесення змін вручну.

3.3 Відгуки

Відгуки є основною частиною сервісу отримання, накопичення та обробки відгуків користувачів інтернет-ресурсів. Через це система має мати якнайбільше можливостей для комфортної роботи, мати простий графічний інтерфейс та надавати як користувачам, так і власнику проекту якомога більше корисних даних.

Щодо можливостей над відгуками то щонайменше користувачі та модератори очікують наступні можливості:

- Створення проектів
- Їх модерація
- Видалення проектів

Базовою можливістю для користувачів має бути можливість створення відгуку до проекту. Проте для створення відгуку має бути виконано декілька умов:

1. Користувач авторизований
2. Проект існує

3. Проект не заблокований
4. Дані відгука пройшли першочергову перевірку

Тільки якщо всі умови вище задоволені то тоді сервіс має створити відгук до відповідного проекту.

Функція яка буде виконувати відповідні перевірки може мати наступний вигляд:

```
usr := session.GetUser(c)
if usr == nil {
  c.JSON(http.StatusForbidden, gin.H{"status": "failed", "error": "should be logged in"})
  return
}

prjSplit := strings.Split(c.Request.URL.Path, "/") // { "", "p", projName, "additem" }
var prj *models.Project
var err error
if prj, err = a.Projects.ByURL(prjSplit[2]); err != nil {
  c.JSON(http.StatusBadRequest, gin.H{"status": "fail", "error": "project not found"})
  return
}
if prj.Locked {
  c.JSON(http.StatusBadRequest, gin.H{"status": "fail", "error": "project is locked"})
  return
}

bProjItem := &models.ProjectItem{
  // Removed. Filling needed project and item params
}
bProjItem.PopulateFromForm(c.Request)
```

```
// Check if data is "valid"
if !bProjItem.ValidRegData() {
  c.JSON(http.StatusBadRequest, gin.H{"status": "fail", "error": "invalid form data"})
  return
}

_, err = a.Projects.ItemAdd(bProjItem)
if err != nil {
  c.JSON(http.StatusBadRequest, gin.H{"status": "fail", "error": err.Error()})
  return
}

c.JSON(http.StatusOK, gin.H{"status": "ok"})
```

В цьому випадку відбувається спочатку перевірка чи авторизований користувач, потім відбувається перевірка по шляху запиту чи існує потрібний проект, якщо існує – перевіряється чи він не заблокований.

Після цього створюється внутрішній вигляд відгуку у вигляді змінної. У неї заповнюються параметри з проекту та запиту користувача. Після цього перевіряється чи правильні дані надані користувачам у функції bProjItem.ValidRegData яка може мати наступний вигляд:

```
func (p *ProjectItem) ValidRegData() bool {
  p.TrimData()
  isNotValid := len(p.Name) < 5
  isNotValid = (p.ProjectID <= 0) || isNotValid
  isNotValid = (p.UserID <= 0) || isNotValid
  isNotValid = !isValidNameWithSpace(p.Name) || isNotValid
  return !isNotValid
}
```

В цій функції перевіряється чи достатня довжина назви відгуку, чи дійсний проект та користувач і якщо параметрі дійсні – тоді система реєструє відповідний відгук.

Метод ItemAdd який і додає відгук в базу даних виглядає наступним чином:

```
func (p *Projects) ItemAdd(item *models.ProjectItem) (*models.ProjectItem, error) {
	var regProjItem models.ProjectItem
	p.DB.Where("name = ?", item.Name).First(&regProjItem)

	if regProjItem.Name != "" {
		return nil, errors.New("Project item with such name already registered")
	}

	item.AddedTime = time.Now().UTC()
	item.State = StateNew
	err := p.DB.Create(item).Error
	if err != nil {
		return nil, err
	}
	lk := &models.Like{
		UserID:       item.UserID,
		ProjectItemID: uint(item.ID),
		IsLike:       true,
		IsDislike:    false,
	}
	if err = p.DB.Create(lk).Error; err != nil {
		return nil, err
	}
	p.Counters.Hit(counter.COUNTER_ITEMS)
	return item, err
}
```

В даному методі перевіряється лише чи не існує вже відгуку з такою самою назвою. Після цього відбувається реєстрація відгуку та автоматичне

створення голосу користувача за відповідний відгук. Також додається одиниця до кількості всіх відгуків.

Після проходження всіх цих дій користувачу надається позитивна відповідь що відгук створений. Всі інші дії для відгуків мають бути доступні лише для автору проекту, довірених обличь заданих для проекту або модераторів. Тому для модерації або видалення потрібно обов'язково перевіряти чи є користувач який намагається це зробити одним з вищезазначених людей.

З вищезазначеної інформації проект має містити таку базову інформацію:

- Унікальний ідентифікатор
- Назву
- Опис
- Проект до якого направлений відгук
- Автор відгуку
- Чи заблокований відгук
- Список коментарів

У відгуку ще можуть бути додаткові поля, проте через їх недостатню необхідність – вони не були додані до списку.

В плані модерації автору проекту має бути доступно реагувати на відгук, тобто змінювати внутрішній його стан на один з декількох які можуть бути. З типових станів відгуку можна назвати наступні:

- Новий
- На модерації
- Дублікат
- Спам
- Порушує правила
- На розгляді
- Відмовлений
- В процесі
- Завершений

Всі ці статуси явно вказують на актуальний стан відгуку та те що наразі відбувається з ним. Проте стрибки між станами не є нормальною поведінкою, тому потрібно розуміти з яких статусів автор чи модератор може робити зміну статусу і на які. Наприклад автор проекту не може мати можливості перевести відгук зі стану дублікат в будь який інший стан, адже тоді втратися певний опис цього відгуку та його достовірність. Тим паче що дублікати це саме те, від чого цей сервіс має оберігати за допомогою відкритості даних.

3.4 Коментарі

Коментарі наразі є додатковою складовою проекту, тому реалізується тільки базовий функціонал, тобто тільки їх додавання та відображення. Коментарі можуть мати безліч можливостей, починаючи з важливих – наприклад модерації, і закінчуючи голосуванням та візуальною зміною вигляду [13] найбільш поганих або хороших коментарів.

Рисунок 3.4 — Візуальна зміна коментаря при низькому рейтингу

Незважаючи на те що вони не є основною частиною сервісу – коментарі можуть слугувати причиною для правильної відповіді на відгук автору проекту, що також збільшує їх потребу. Також коментарі дозволяють людям підтримати точку зору автору відгуку, або ж навпаки, повідомити автору проекту про те що цей відгук може бути неправильним, недійсним або за інших причин немає цінності для проекту.

Найпростіший коментар має містити наступні поля:

- Автор
- Дата створення коментаря

- Текст коментаря
- До чого був створений коментар(в даному випадку до якого відгуку)

У більш складних випадках коментарі також можуть містити також відповідь, тобто до якого коментаря був зроблений цей коментар. В такому випадку коментар може бути розміщеним одразу під оригінальним коментарем і також він може бути візуально виділеним, наприклад мати відступ. Також коментарі можуть голоси, що також може мати візуальне відображення на сторінці користувача.

Інакший вигляд можуть також мати коментарі які залишені автором проекту, для того щоб можна було візуально відрізнити людей які голосують та автору проекту, який веде дискусію, або дає відповіді на певні коментарі.

Можливість користувача залишити коментар залежить від певних факторів. Користувач має бути авторизованим на сервісі та мати активований акаунт. Окрім цього відгук, до якого користувач хоче залишити відгук та проект має також бути незаблокованим з будь яких причин. В такому випадку в коді шаблону такі перевірки можуть виглядати наступним чином:

```
{{if and (.usr.IsActivated) (not .item.Project.Locked) (not
.item.Project.LockedByStuff) (not .item.Locked) (not
.item.LockedByStuff)}}<form class="ui reply form">
    <div class="field">
      <textarea name="form_text" id="form_text"></textarea>
    </div>
    <div class="ui blue labeled submit icon button" id="submit">
      <i class="icon edit"></i> {{call .T "item_comments_add"}}
    </div>
  </form>{{end}}
```

Тут відповідно перевіряється чи активований користувач, чи заблокований проект або відгук автором чи модераторами сервісу. Якщо всі перевірки пройшли тоді створюється форма і відправляється користувачу разом з іншим кодом сторінки.

В ситуаціях коли користувач може щось змінити і відправити неправильні або шкідливі дані важливо окрім того що робити перевірки на стороні клієнту, тобто в браузері, так і на стороні сервісу при отримані цих даних. Це має бути правилом для будь яких форм та можливих змін зі сторони користувача.

Для теперішніх потреб сервісу коментарям достатньо мати наступні поля:

- Унікальний ідентифікатор
- Дату створення
- Автору коментаря
- Унікальний ідентифікатор відгуку до якого зроблений коментар
- Текст коментарю

Для такого роду сервісу буде також мати відповіді на певні коментарі, тому до полів вище можна ще також додати унікальний ідентифікатор коментаря на який надається відповідь. Таким чином можна буде просто знайти коментарі верхнього рівня, тобто такі які не відповідають інші коментарі, та відповідно для кожного коментаря можна буде знайти повний список відповідей.

4 ГРАФІЧНИЙ ІНТЕРФЕЙС. ТЕСТУВАННЯ ІНТЕРНЕТ-СЕРВІСУ

4.1 Графічний інтерфейс

Графічний інтерфейс інтернет-ресурсу є чи не найважливішою складовою всього ресурсу при контакті з користувачем. Якщо графічний інтерфейс не дозволяє виконувати потрібні користувачеві операції за мінімальну кількість кроків або є неінтуїтивним – користувач не буде користуватись таким ресурсом. Виключення можуть бути лише для сервісів-монополістів, і таких зараз можна знайти одиниці. Причиною на те є висока ступінь конкуренції яку неможливо контролювати. Винятки можуть бути промислові речі, або речі які недоступні публічно. Наприклад програмне забезпечення для супер-комп'ютерів через те що люди фактично не можуть мати до нього доступу.

4.1.1 Пошук бібліотеки елементів

За для скорочення часу потрібного на розробку ресурсу буде використовуватись готова бібліотека з елементами керування, сітками розбиття та загальними покращеннями візуального вигляду сторінки.

Доступність таких бібліотек значно спрощує розробку і скорочує потрібний час. Вони дозволяють навіть людям які мають незначні пізнання у веб-розробці розробити привабливий і функціональний сайт при цьому надавши користувачу широкий вибір елементів для використання. Наразі кількість таких бібліотек досить висока і всі вони надають базовий функціонал який включає в себе:

- Сітку розбиття для елементів на сторінці
- Базові макети для швидкої розробки
- Базові елементи користувача такі як кнопки та поля вводу
- Типографічні елементи

Певні бібліотеки надають більш широкий набір можливостей, проте це не завжди є необхідним для невеликих проектів, та і деколи якість таких проектів може бути низькою, через велику кількість потрібних ресурсів та їх нестачу у розробника.

І саме через те що багато бібліотек надають схожі можливості вибір може бути зроблений на їх оцінках, кількості реалізованих користувацьких елементів та суб'єктивному рішенню про вигляд кожної з них. Також при перегляді кожної такої бібліотеки розробник може будувати макети вигляду сайту на їх основі і також за цим примати рішення по достатньості реалізованих елементів та дизайну. Саме так і було обрано бібліотеку для цього сервісу: по кількості якісно реалізованих елементів та суб'єктивна оцінка зовнішнього вигляду.

4.1.2 Огляд бібліотек з компонентами

Проведемо огляд декількох бібліотек які розповсюджуються безкоштовно. Однією з найбільш часто використовуваних готових бібліотек є Bootstrap [8], розроблена компанією Twitter, яка недавно отримала оновлення до четвертої версії.

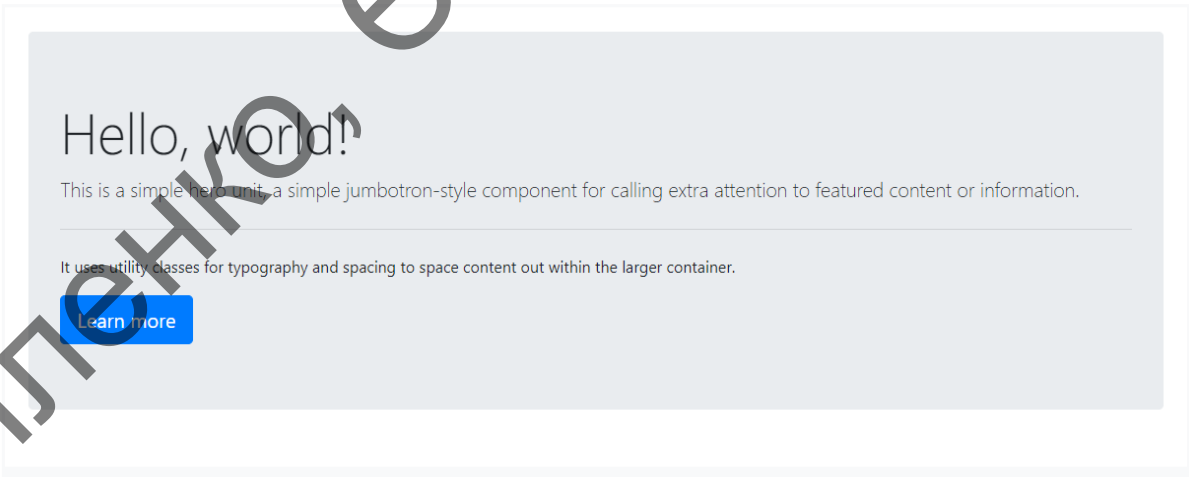

Рисунок 4.1 — Компонент Jumbotron у Bootstrap 4

Вона має велике поширення через свою простоту у використанні, можливості використання без скачування на сайт розробника та достатню

кількість користувацьких елементів доступних для використання. Велика кількість прикладів, доступна документація, можливість вибрати готові теми та великі можливості для персоналізації також позитивно впливають на широту використання цієї бібліотеки.

Ця бібліотека має свій унікальний вигляд, який явно відрізняє її від інших. Кольори якими пропонують виділяти елементи є яскравими і насиченими. Таке вираження явно дає розподіл загальній інформації на сторінці та елементам які виконують якісь дії. Цей підхід є більш універсальним, що дозволяє використовувати його на багатьох ресурсах де зовнішній вигляд не є першочерговою причиною ним користуватись. Не рідкістю є окремі компоненти для цього фреймворку на просторах інтернету, що ще більше дає йому поширення.

Ця бібліотека дозволяє швидко реалізувати потрібний функціонал ще завдяки достатній кількості доступних одразу компонентів. Тому вона досить часто використовується при розробці внутрішніх інтернет-ресурсів компаній.

Для особливо вибагливих людей окрім безкоштовних тем існують навіть платні, як видно з рисунок 4.2.

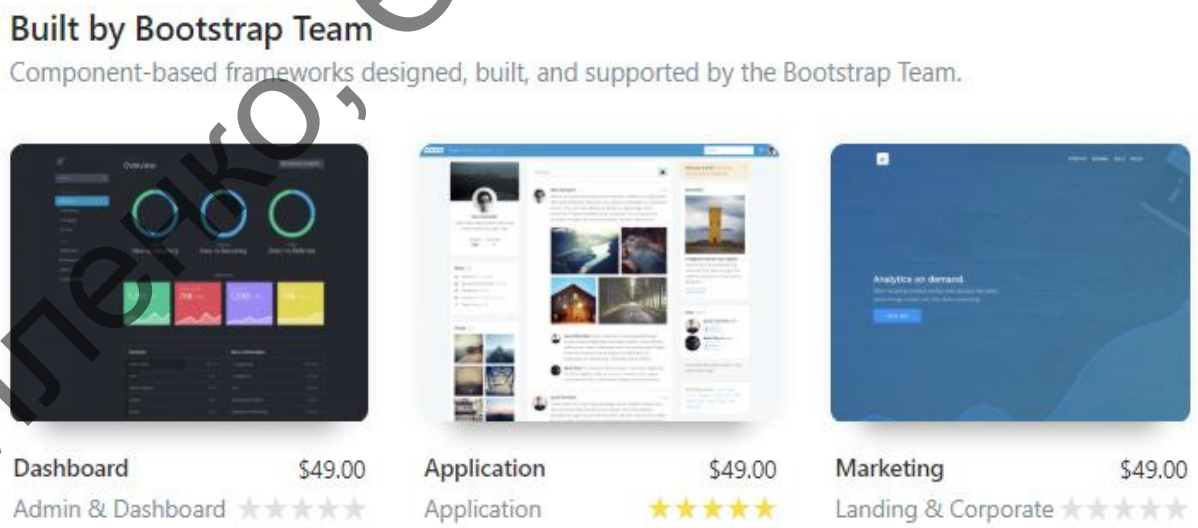

Рисунок 4.2 — Платні теми для Bootstrap

Semantic UI [9] безкоштовна бібліотека яка має широкий вибір готових елементів та макетів для їх розташування. Вона також має приклади макетів які можна використовувати на власному сайті та обширні можливості для видозмінення зовнішнього вигляду.

Цю бібліотеку також можна використовувати без завантаження на свій сервер, проте тоді втрачається можливість використовувати інші теми, окрім тієї що надається одразу з бібліотекою. Semantic UI надає щонайменше 11 готових тем для своїх елементів, не потребуючи для їх зміни та використання ніяких складних дій від користувача.

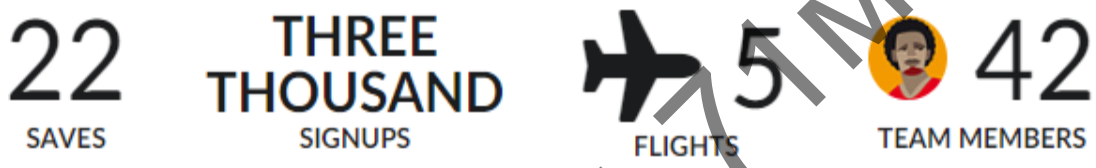

Рисунок 4.3 — Елемент статистики

Серед стандартних елементів, таких як кнопки, таблиці, форми та групи елементів вона надає такі елементи як статистика, покрокові дії, список коментарів, стрічку подій, перевірку форм та навіть окремі форми для рекламних банерів.

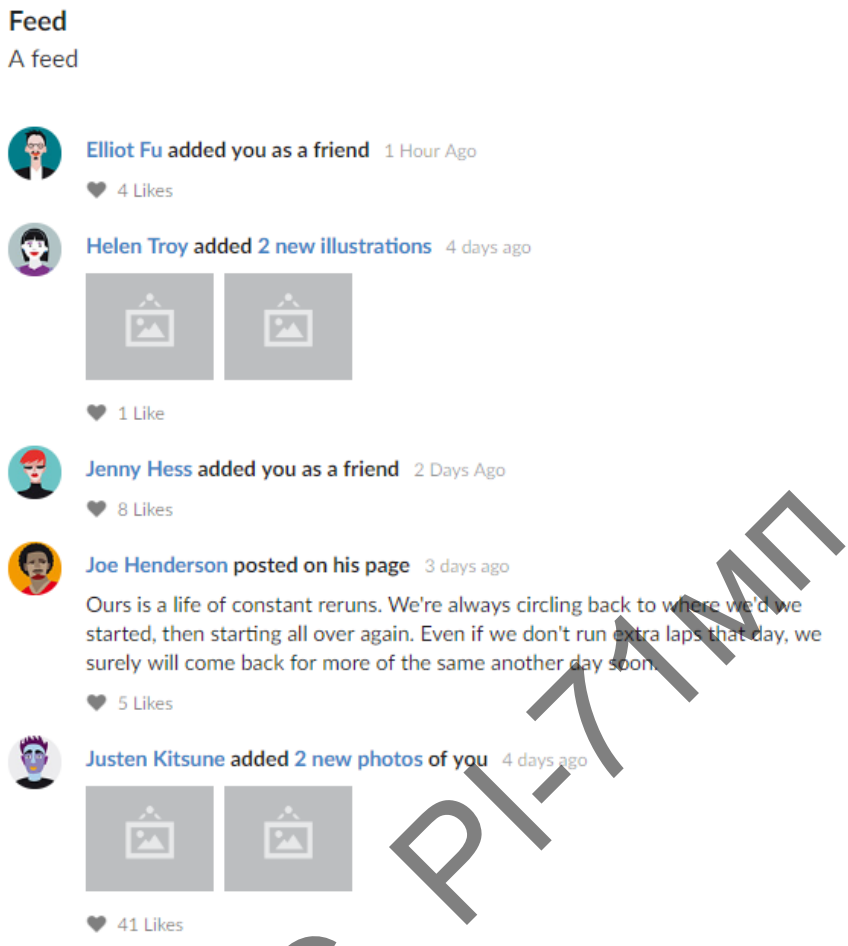

Рисунок 4.4 — Елемент стрічки подій

Окрім цього, особливістю цієї бібліотеки компонентів є його семантичність, тобто назви класів які використовуються для модифікації елементів сторінки, на чому і базується назва. Наприклад, для отримання стрічки подій достатньо до компоненту div додати класи ui feed, і всередині використовувати div’и з класом event, в яких і потрібно розміщувати необхідні події. Така семантичність дозволяє вивчити назви і використання класів досить швидко, що пришвидшує розробку та повторне використання компонентів бібліотеки.

В цій бібліотеці тут також пропонуються нестандартні елементи, такі як панель навігації яка розміщується поза сторінкую і по спрацюванню події виїздить з вказаної сторони екрану. Є «рейки», які дозволяють винести певні елементи керування або текст на одну або обидві сторони

сторінки, які будуть фіксовані по висоті і при прокрутці не будуть залишати межі видимої частини сторінки, тим самим забезпечуючи швидку доступність винесених туди елементів. Окремо можна сказати про покрокові дії [10], які дозволяють людям виконуючи послідовність дій розуміти на якому етапі вони знаходяться та зрозуміти статус дії, а для розробника це можливість розділити одну велику дію на декілька менших, що полегшить користування сайтом та зменшить негативність від однієї великої форми.

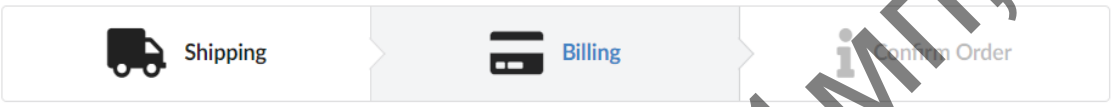

Рисунок 4.5 — Покрокові дії

Також тут є достатня кількість невеликих елементів, які дозволяють акуентувати увагу на конкретних місцях або елементах. До таких можна віднести іконки – набір малих картинок які дозволяють графічно описати дію або стан, рейтинг – одна або група іконок які дозволяють вказати своє вподобання, флаги – іконки флагів різних країн які можна використовувати, наприклад, для вибору країни чи мови на сайті, «ярлк» або «позначка» - слово або текст невеликого розміру в окремому виконанні не схожому на інші частини сайту для того щоб виразити окремі параметри або характеристики об’єкту.

UIkit [11] також досить відомий фреймворк, який містить базовий набір компонентів, розміщення їх на сторінці та певний набір нестандартних компонентів.

Тут пропонуються всі потрібні елементи для типового сайту, такі як кнопки, розмітка сторінки за сіткою, панель навігації та групування у форми. Їх вибір є достатнім і окрім цього є ще нестандартні елементи які можна розділити на функціональні та нефункціональні.

До нефункціональних можна віднести ті що дозволяють розміщувати текст або елементи на сторінці нестандартним чином, або значно

спрощують їх розміщення. Сюди можна віднести акордіон – елемент що дозволяє розбити текст та елементи згруповані за певною тематикою і відкривати їх по назві, або іншому характерному опису, елемент стаття, яка дозволяє швидко отримати готовий вигляд статті з вказанням всіх необхідних її атрибутів, маркери, які дозволяють поверх зображень наносити кнопки з певною взаємодією, що корисно для опису предмету на картинці, та інше.

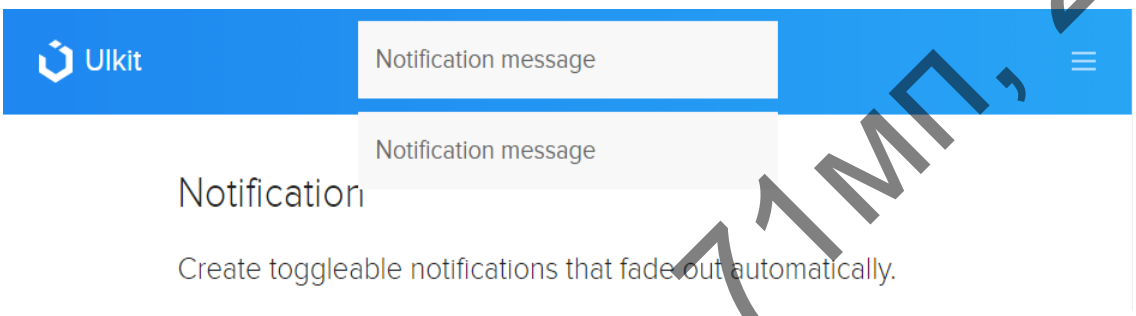

Рисунок 4.6 — Нотифікації на сторінці

Функціональні елементи включають в себе нотифікації – можливість показати тимчасове повідомлення користувачу з певної сторони екрану поверх сторінки, лічільник зворотнього часу, який дозволяє вести відлік для, наприклад, акційних пропозицій на сайті, елементи які можна сортувати за допомогою переносу.

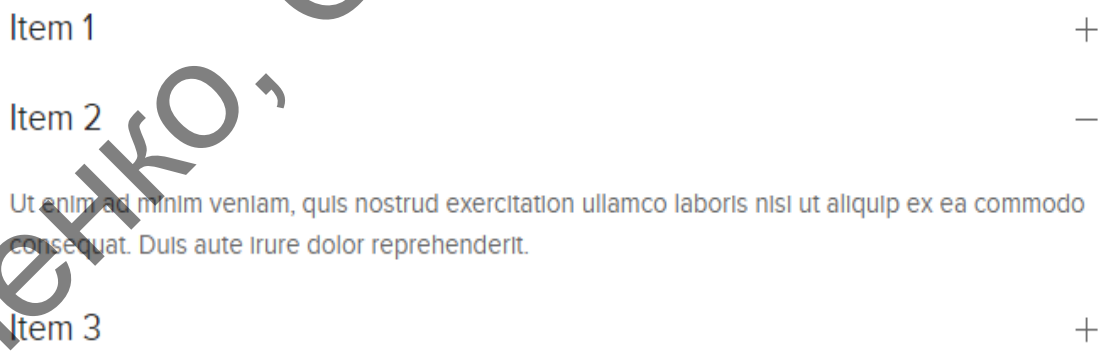

Рисунок 4.7 — Елемент акордіон

Цей фреймворк пропонує мінімалістичний дизайн, проте розміри елементів та їх вигляд може сподобатись не кожному. Головним кольором який виділяє необхідні елементи є синій, і гсотрі форми елементів також

певним чином звужують аудиторію користувачів. Використовуючи UIkit потрібно чітко розуміти стиль сторінки і мати готовий макет сайту бо може бути важко робити виправлення або зміну вигляду сайту пізніше, якщо не продумати це заздалегідь.

Рисунок 4.8 — Форма з різних елементів

Кількість доступних безкоштовних бібліотек є дуже великою, тому розглядати всі немає можливості. Для різних потреб є різні рішення, включно і з такими бібліотеками. Вони є окремо для сайтів які будуть переглядатись з комп'ютерів, окремо які спеціально розроблялись для перегляду з мобільних пристроїв і ті, що називаються responsive [12], тобто які автоматично підлаштовують зовнішній вигляд до пристрою користувача. Більшість бібліотек намагаються мати адаптивний дизайн, тому що все більше користувачів користуються інтернетом з мобільних пристроїв. Версія сайту для настільних комп'ютерів надзвичайно незручна для мобільних користувачів, так як постійно потрібно наближувати щоб побачити потрібний текст чи графічну складову. Певним чином це

нівелюється великими розмірами нових мобільних пристроїв, проте ніщо не замінить «нативний» користувацький інтерфейс для мобільних пристроїв.

Бібліотеками які спеціально розроблялись для мобільних пристроїв є наприклад Bootstrap, а бібліотека Semantic UI є адаптивною, що означає що вона автоматично буде підлаштовувати свій вигляд до розміру екрану користувача.

Для використання в реалізації графічного інтерфейсу було обрано бібліотеку Semantic UI через свою простоту, велику кількість доступних елементів та достатню простоту зміни під свої потреби.

4.1.3 Розробка графічного інтерфейсу

Перед розробкою графічного інтерфейсу як мінімум потрібно мати уяву про розділення компонентів сервісу та хоча б наявність початкових елементів керування для користувача.

Структуру сайту можна розділити на декілька частин. Розділ може бути не кінцевим через те що в залежності від статусу користувача, чи пройшов він авторизацію, чи активований його акаунт будуть також залежати зовнішній вигляд та можливість виконувати певні дії.

Ресурс для накопичення та обробки відгуків має мати наступні базові розділи:

- Головна сторінка
- Сторінка всіх проектів
- Сторінка проекту
- Сторінка відгуку

Ці розділи мають існувати в будь якому випадку і будуть відвідуватись користувачами найчастіше.

До розділів які також мають бути, проте за певних умов можуть не бути необхідними є:

- Сторінка авторизації користувача
- Сторінка профілю користувача
- Сторінка модерації проектів, відгуків так коментарів

Всі доступні для користувача дії мають бути йому зрозумілими, простими та доступними. Такий підхід забезпечить максимальну ефективність сервісу та доступність як для користувачів, так і авторів сервісів.

Головний екран сервісу має надавати достатню інформацію для користувачів про те що надає сервіс та певні посилання на сторінки які можуть зацікавити його. Для запропонованого сервісу на головній сторінці є короткий опис, переваги щодо інших сервісів та посилання на корисну інформацію, таку як сторінка всіх проектів та умови користування сервісом.

На рисунку 4.9 показаний приблизний вигляд головної сторінки яка буде надана користувачеві коли він буде переходити на адресу сервісу. Візульно можна просто розділити її на три частини: головне меню згори, текстова інформація поцентру сторінки та список основних посилань після текстової інформації. Такий розподіл показує важливість інформації та неперенавантажує користувача зайвою графікою чи текстом.

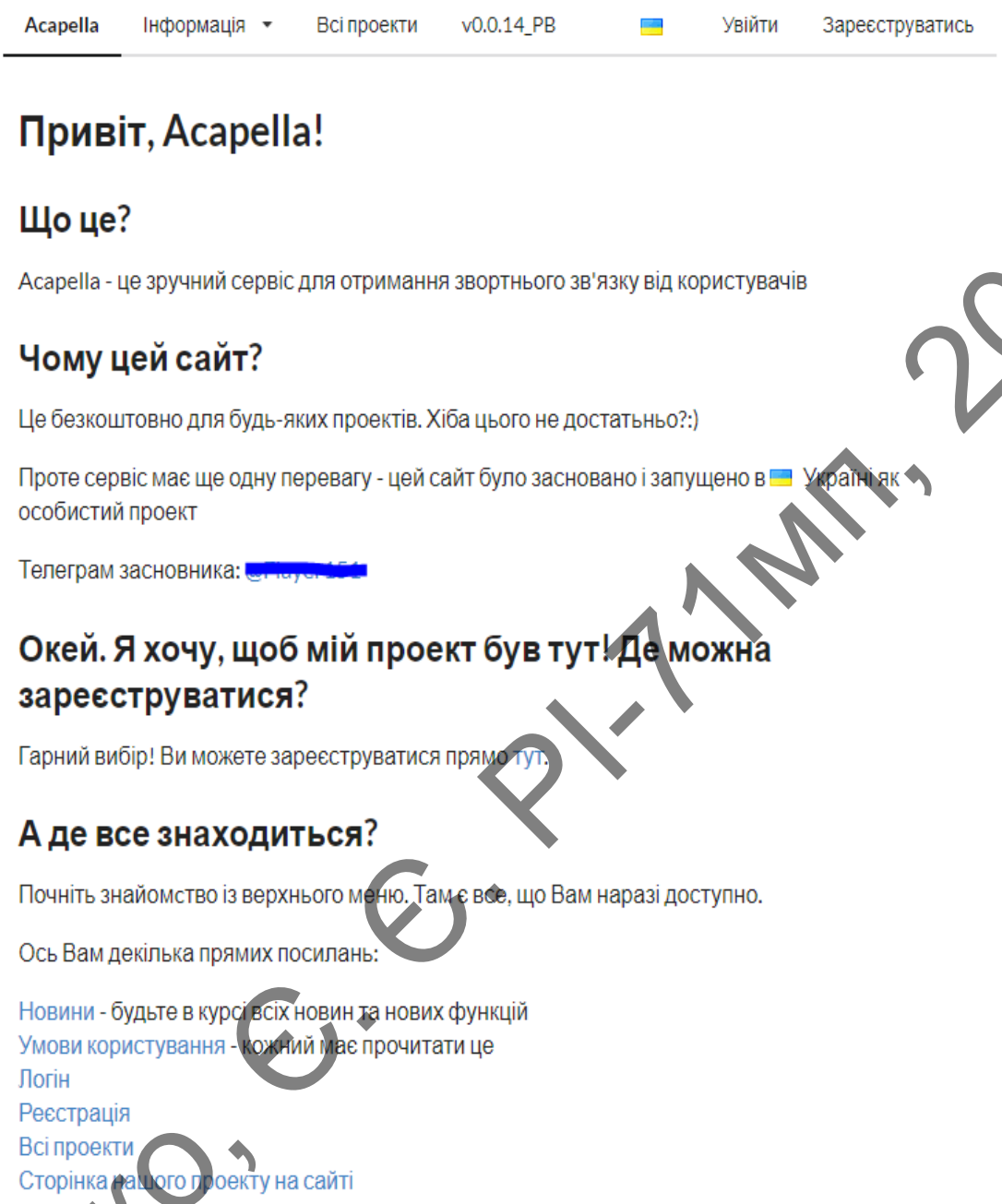

Рисунок 4.9 — Головна сторінка

Сторінка авторизації користувача має містити також мінімальну кількість елементів для достатнього вираження на що потрібно зосереджувати увагу при переході на цю сторінку. Приклад такого блоку авторизації показаний на рисунку 4.10

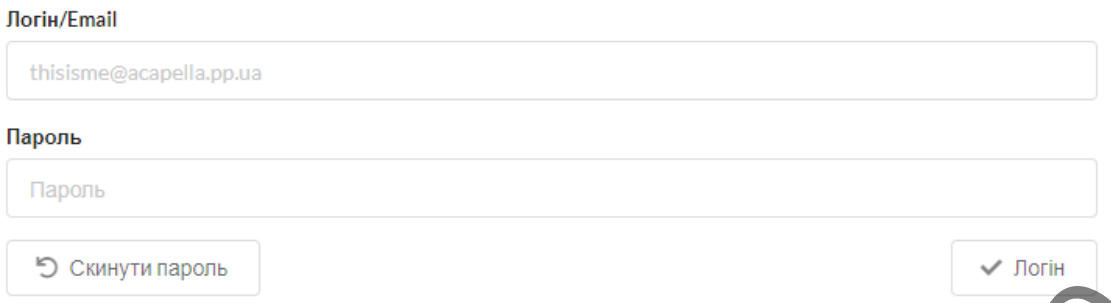

Рисунок 4.10 — Блок авторизації користувача

Вигляд сторінки користувача дозволяє мінімально необхіну кількість операцій, а точніше скинути пароль та провести авторизацію, для того щоб явно показати користувачу які можливості надаються на цій сторінці та не мати на ній нічого зайвого.

Вигляд проекту також не має надавати всю необхідну інформацію яка потрібна користувачу, проте так щоб вона була доступна та не потребувала додаткових кроків на її отримання або сприймання. Така сторінка може мати вигляд, що запропонований на рисунку 4.11

На цій сторінці користувач має нагорі сторінки обраний проект, коментар до нього та автора проекту. Під цією інформацією доданий рядок з кнопкою для можливості додати новий відгук. Після цього до кінця сторінки надаються відгуки з їх назвами, частиною коментаря до них, автором відгуку та кількістю користувачів які надали свої голоси за відгук.

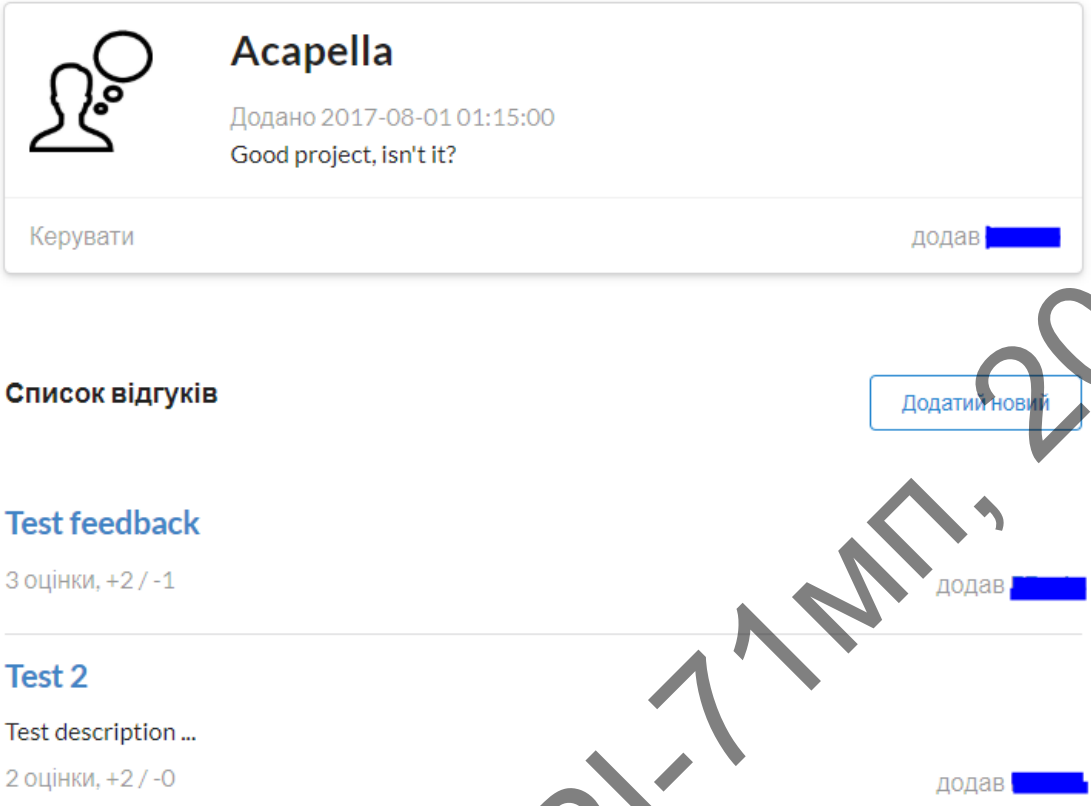

Рисунок 4.11 — Сторінка проекту

Сторінка конкретного відгуку для проекту має містити наступну інформацію: проект до якого створений відгук, назва відгуку, опис відгуку, голоси за відгук та коментарі. Це достатній набір інформації якій дозволить користувачам зробити висновки щодо роботи автору проекту та відношення користувачів щодо конкретного відгуку. Запропонована інформація може мати наступний вигляд, який показаний на рисунку 4.12.

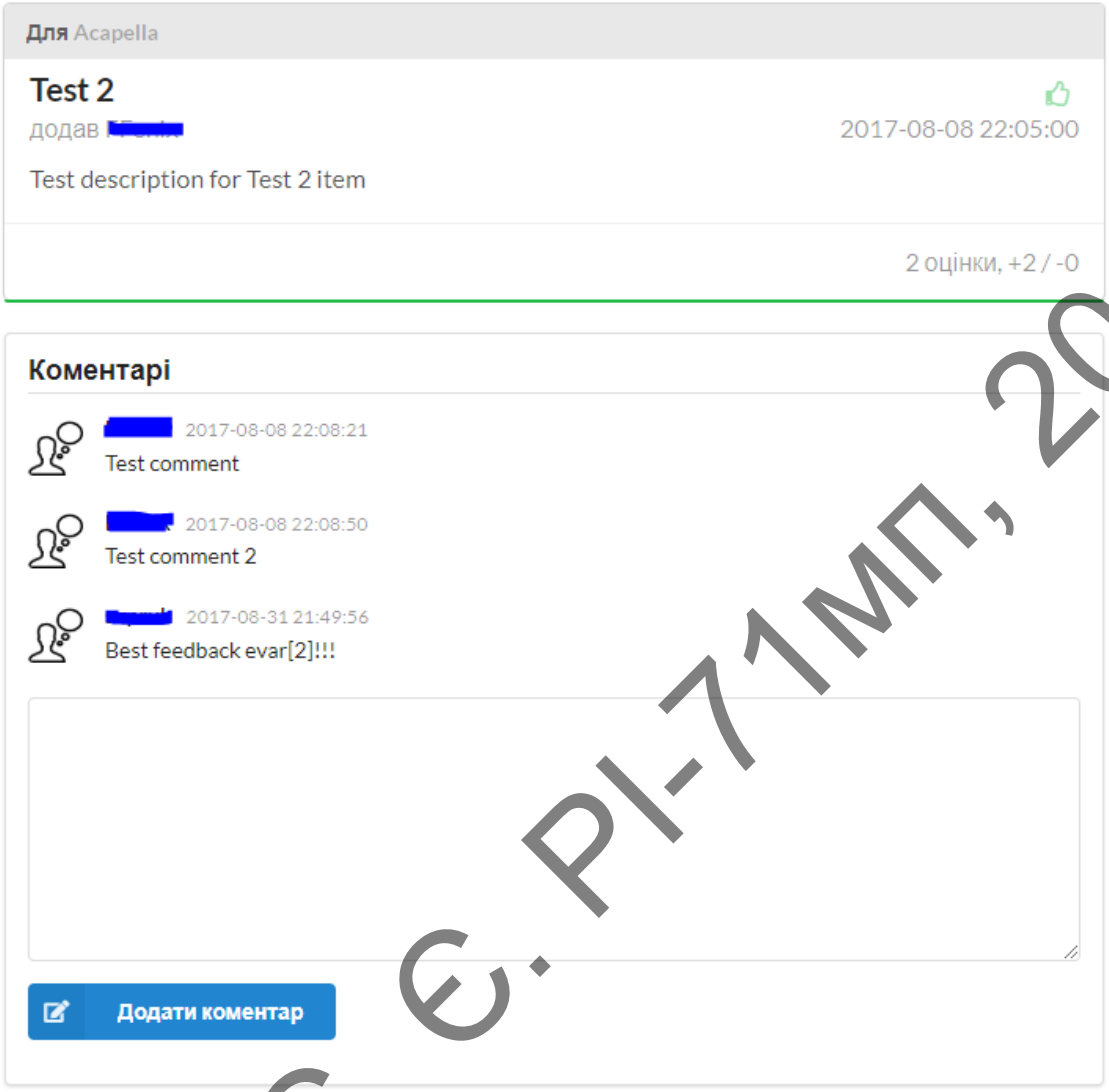

Рисунок 4.12 — Сторінка відгуку

Описані сторінки дозволять швидко і доступно користувачам користуватись сервісом при цьому отримуючи всю необхідну інформацію та витрачаючи мінімальну кількість часу для сприйняття інформації. Вигляд сайту також розроблений в мінімалістичному дизайні, що є більш практичним підходом до графічного інтерфейсу ніж більш графічно-навантажені інтерфейси через те що такий вигляд може сподобатись більшій аудиторії та бути для них простішим в користуванні.

4.1.4 Інтеграція системи

Запропонована система відрізняється від конкурентів можливістю простої інтеграції з існуючими інтернет-ресурсами. Така інтеграція

дозволяє власникам інтернет-ресурсів швидше отримувати зворотній зв'язок і з більшою ймовірністю через те що користувачу не потрібно явно робити перехід на сторонні ресурси та виконувати там ніяких дій.

Існує декілька варіантів інтеграції, що дозволить автору ресурсу самостійно вибрати необхідний йому варіант який в більшій ступені впишеться в існуючий інтернет-ресурс.

Перший варіант це інтеграція за допомогою прикладного програмного інтерфейсу системи. Такий варіант є найбільш придатним до конфугірації бо автор рерсу буде виконувати лише необхідні йому запити для того щоб отримати, додавати або змінювати необхідну інформацію. Проте такий варіант не дуже підходить коли немає необхідних знань або ресурсів для того щоб забезпечити правильну роботу такої інтеграції. З усіх можливих інтеграцій такий варіант є найбільш складним в реалізації, проте через свою безмежну можливість в конфігурації на інтернет-ресурсі та кращу інтеграцію в кінцевий продукт є хорошим варіантом для використання якщо є відповідна можливість та ресурси.

У випадку з використанням прикладного програмного інтерфейсу розробник не отримує сторінку з елементами керування, а чисті дані які задані в програмному інтерфейсі, що можна побачити на рисунку 4.17

Request URL: https://www.acapella.pp.ua/v2/server/projects/
Request Method: GET
Response Time: 0.138 seconds
Response Status: 200 - OK

200

Response Body | Response Body (RAW) | Response Headers | Request Details


```
[
  {
    "Description": "",
    "ID": 2,
    "Icon": "",
    "InternalURL": "test",
    "IsPublic": false,
    "Locked": false,
    "LockedByStuff": false,
    "Moderated": false,
    "Name": "test_internal",
    "RegisterTime": "",
    "URL": "",
    "User": "FFenix"
  },
  {
    "Description": "Good project, isn't it?",
    "ID": 1,
    "Icon": "",
    "InternalURL": "acapella",
    "IsPublic": true,
    "Locked": false,
    "LockedByStuff": false,
    "Moderated": false,
    "Name": "Acapella",
    "RegisterTime": "2017-08-01 01:15",
    "URL": "",
    "User": "FFenix"
  }
]
```


Рисунок 4.13 — Відповідь прикладного програмного інтерфейсу

Наступна можливість інтеграції це використання субдомену сервісу зворотного зв'язку що дозволяє використовувати такий сервіс на власному ресурсі з власним доменом. Таким чином якщо адреса сторінки проекту має вигляд https://acapella.pp.ua/p/acapella/ то в разі використання субдомену цей проект матиме наступну адресу: https://acapella.acapella.pp.ua/ що забезпечує використання такого ресурсу на власному домені за допомогою простих змін в DNS налаштуваннях.

Простота цієї інтеграції має недолік в плані налаштувань. Через те що використовується сторонній ресурс то можливість налаштування вигляду

сторінки є досить малою і в більшості випадків може бути взагалі відсутньою. Зміна вигляду сторінки має бути реалізована на стороні сервісу зворотного зв'язку, і якщо такої можливості немає – користувач стороннього інтернет-ресурсу буде візуально розуміти що використовується стороній сервіс, що показано на рисунку 4.14.

Рисунок 4.14 — Використання субдомену

Таким чином розробник сервісу зворотного зв'язку має самостійно забезпечити хоча б найменшу можливість для видозміни сторінки для нормального її використання на інших інтернет-сервісах.

Ще однією можливістю інтеграції ресурсу є так звані «віджети», тоюто невеликі елементи керування на сторінці які при натисканні відкривають модальне вікно в якому користувачу пропонується переглянути або додати новий відгук. Цей варіант є також досить простим у використанні через те що власнику інтернет-ресурсу потрібно лише додати певний код на сторінки свого ресурсу де він хоче мати відповідні можливості.

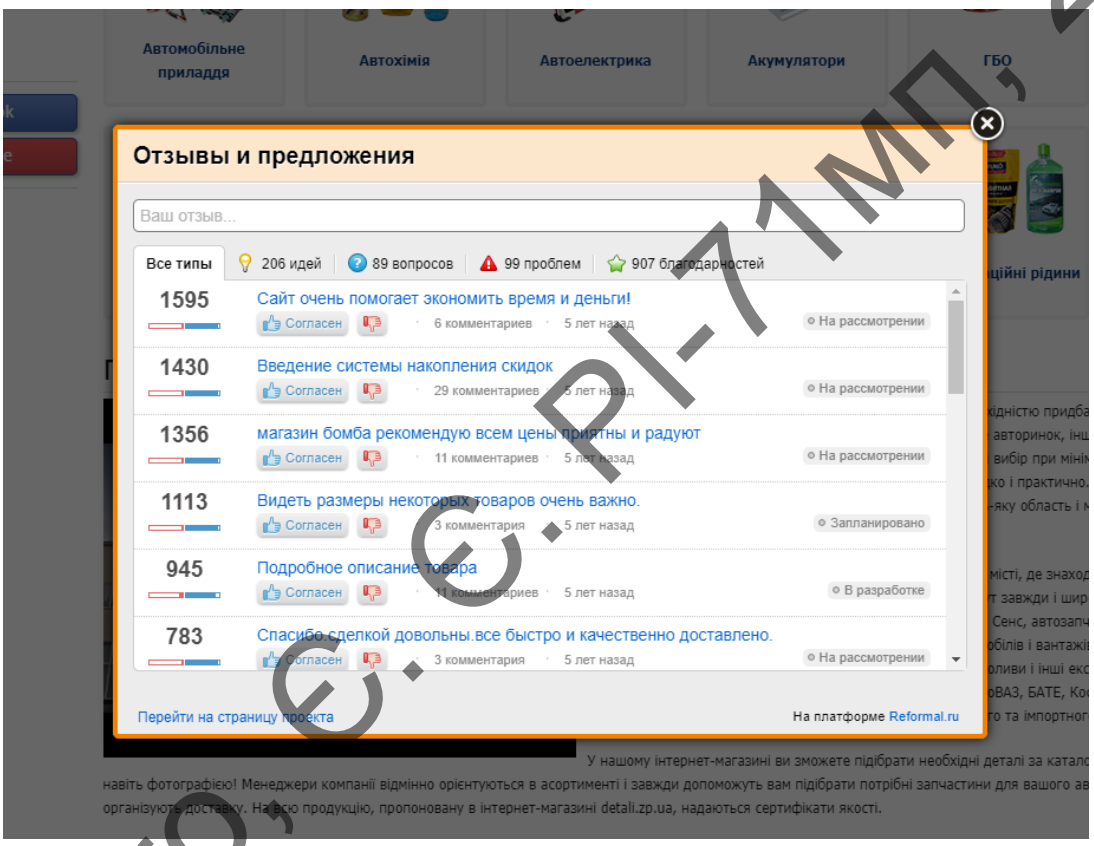

Рисунок 4.15 — Віджет ресурсу Reformal.ru

Такі віджети є зручними у користуванні ще через те, що користувачу не потрібно переходити на інші сторінки, відкривати інших вкладок або нових вікон браузеру, що показано на рисунку 4.15. Проте можливість видозміни такого модального вікна також залежить від розробника сервісу зворотного зв’язку, що також ускладнює її використання коли потрібно звести стиль сторінки до одного. Проте в деяких ситуаціях видозміна такого вікна взагалі може бути неможливою через те що такі віджити

поширюють відомість про сервіс, яким користується певний інтернет-ресурс.

Це не всі можливі методи інтеграції, проте вони є найрозповсюдженішими. Вони дозволяють швидко та ефективно виконувати необхідний функціонал і також вони не навантажують користувачів. Тобто використовуючи один або декілька таких методів можна отримати максимальну кількість відгуків від користувачів які користуються інтернет-ресурсом за допомогою простої інтеграції такого сервісу.

4.2 Тестування інтернет-сервісу

4.2.1 Працездатність сервісу

Тестування відбувалось за допомогою як автоматизованих засобів, так і вручну. Після проведення низки тестів було отримано результат що сервіс працює коректно та відповідає заданим параметрам в технічному завданні.

Автоматичні тести мають інтеграційний характер, тобто перевіряється працездатність уже кінцевих сторінок ресурсу. Таким чином окрім того що перевіряється вірність роботи окремих функцій та пакетів також гарунтається працездатність ресурсу після того як він буде розгорнутий на сервері. Такі тести запускаються перед тим як завантажити зміни в систему контролю версій.

Наприклад до складу автоматизованих перевірок входять перевірки авторизації користувача як за правильних даних для авторизації, так і без них. Так виглядає тест для перевірки авторизації користувача без правлильних вхідних даних:

```
func TestUserLoginPostEmpty(t *testing.T) {

    response, _ := doTestRequest(«POST», «/login», nil, getPostHeader())
```

```
 if response.Code != http.StatusForbidden {
        t.Errorf(«Code != %d, actual is %d», http.StatusForbidden, response.Code)
 }
}
```

Тут виконується запит на URL /login з методом POST, який використовується для того щоб передати дані авторизації на сервер. Для такого запиту потрібно вставити відповідні заголовки, такі як тип контенту який відправляється на сервер. Для таких запитів тип контенту має значення application/x-www-form-urlencoded та опціонально ще вказується кодування даних які передаються, що має значення UTF-8. Ці дані автоматично надаються при виклику функції getPostHeader().

Так як перевіряється що авторизація не може бути пройденою то у запит не передається ніяких даних, щоб симулювати невірну авторизацію. Для всіх невірних авторизацій сервер має повертати статус 403, що означає що доступ заборонений. В тесті числове значення статусу заміненне на зміну http.StatusForbidden зі стандартного пакету net/http.

Якщо сервер повертає код статусу що відрізняється від 403 то тест вважається непройденим та пише про це повідомлення в консоль з інформацією про отриманий код та той що має бути.

4.2.2 Швидкодія

Для перевірки швидкодії були використані додатки Apache Benchmark Apache JMeter

Ці утиліти використовуються для отримання результатів швидкої сайтів, часу отримання відповіді та супутньої статистики викликів за різних параметрів. Apache Benchmark дозволяє перед проведенням тесту задавати параметри кількості запитів які можуть виконуватись паралельно,

загальну кількість запитів яка має бути відправлена під час тесту та адресу на яку має бути відправлений запит. Також ця програма дозволяє задавати метод який має бути виконаний на сервері, за замовчуванням він становить GET, і дані які мають бути відправлені на сервер разом з запитом.

За допомогою цієї утиліти було проведено декілька тестів з різними параметрами які мають показати зміну в швидкодії сервісу в залежності від кількості паралельних користувачів.

```
ffenix@ffenix-GE62-6QF:~/GoglandProjects/Acapella$ ab -q -c 10 -n 2000 https://acapella.pp.ua/
This is ApacheBench, Version 2.3 <$Revision: 1807734 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking acapella.pp.ua (be patient).....done


Server Software:        cloudflare
Server Hostname:        acapella.pp.ua
Server Port:            443
SSL/TLS Protocol:       TLSv1.2,ECDHE-ECDSA-CHACHA20-POLY1305,256,256
TLS Server Name:        acapella.pp.ua

Document Path:          /
Document Length:        4958 bytes

Concurrency Level:      10
Time taken for tests:   72.215 seconds
Complete requests:      2000
Failed requests:        4
   (Connect: 0, Receive: 0, Length: 4, Exceptions: 0)
Non-2xx responses:      4
Total transferred:      11236192 bytes
HTML transferred:       9912276 bytes
Requests per second:    27.69 [#/sec] (mean)
Time per request:       361.076 [ms] (mean)
Time per request:       36.108 [ms] (mean, across all concurrent requests)
Transfer rate:          151.95 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        6   10   3.6      9      61
Processing:    39  325 2131.4     53   30955
Waiting:       38  324 2131.4     52   30954
Total:         49  335 2131.7     63   30972

Percentage of the requests served within a certain time (ms)
  50%     63
  66%     70
  75%     75
  80%     79
  90%     92
  95%    110
  98%   3200
  99%  15371
 100%  30972 (longest request)
ffenix@ffenix-GE62-6QF:~/GoglandProjects/Acapella$
```

Рисснуок 4.16 — Швидкодія головної сторінки

Перший результат на рисунку 4.16 виконання показує час виконання запитів за певними часовими показниками. З результатів видно що сервіс наразі може забезпечити відповідь на 27,69 запитів в секунду, за умови що

постійно та одночасно на сайт виконується запит на головну сторінку десятьма клієнтами. Проте це значення не є достовірним, так як воно отримано з часу який був витрачений на тест, проте час сам не є достовірним через повільну роботу стороніх ресурсів. З цього результату також видно що на 95% запитів час відповіді сервера не перевищив 110 мс, що майже в десять разів менше за встановлений в технічному завданні, а 90% запитів отримали повний результат менше ніж за 92мс. Максимальний час запиту склав 30 секунд, що означає проблеми з мережею або недоступністю стороніх бібліотек при запиті головної сторінки сервісу.

Потрібно також провести тест з отриманням часу відповіді при запиті сторінки проекту, через те що така сторінка потребує більше запитів в базу даних та загалом передає більше даних з серверу.

Для виконання цього тесту буде запитуватись сторінка з власним проектом, на якій вже існує два вдігуки. Такий запит має зробити декілька запитів в базу даних, що збільшує навантаження на диск та оперативну пам’ять.

```
ffenix@ffenix-GE62-6QF:~/GoglandProjects/Acapella$ ab -q -c 10 -n 2000 https://acapella.pp.ua/p/acapella
/
This is ApacheBench, Version 2.3 <$Revision: 1807734 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking acapella.pp.ua (be patient).....done


Server Software:        cloudflare
Server Hostname:        acapella.pp.ua
Server Port:            443
SSL/TLS Protocol:       TLSv1.2,ECDHE-ECDSA-CHACHA20-POLY1305,256,256
TLS Server Name:        acapella.pp.ua

Document Path:          /p/acapella/
Document Length:        5132 bytes

Concurrency Level:      10
Time taken for tests:   30.107 seconds
Complete requests:      2000
Failed requests:        0
Total transferred:      11588000 bytes
HTML transferred:       10264000 bytes
Requests per second:    66.43 [#/sec] (mean)
Time per request:       150.534 [ms] (mean)
Time per request:       15.053 [ms] (mean, across all concurrent requests)
Transfer rate:          375.88 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        7   11   4.3     10      49
Processing:    64  139  28.7    138     396
Waiting:       63  138  28.7    137     395
Total:         72  150  29.1    149     404

Percentage of the requests served within a certain time (ms)
  50%    149
  66%    158
  75%    164
  80%    168
  90%    180
  95%    194
  98%    217
  99%    237
 100%    404 (longest request)
ffenix@ffenix-GE62-6QF:~/GoglandProjects/Acapella$
```

Рисунок 4.16 — Швидкодія сторінки з проектом

З отриманих значень на рисунку 4.16 видно що найдовший час отримання сторінки був 404мс, що є в два з половиною рази менше ніж потрібно за технічним завданням. 90% з усіх запитів отримали відповідь менше ніж за 180мс, і час розходження між максимальним та середнім часом не такий великий, що вказує на правильність виконання тесту та стабільного часу виконання запитів на сервері. При десяти постійних клієнтах сервер може обробити понад 60 запитів на секунду, а саме 66,43 запити на секунду, що є достатнім показником зважаючи на те що ще не проводились роботи по оптимізації як програмної складової сервісу, так і складової інтернет ресурсу.

Також результати є більш ніж задовільними через те що даний сервіс працює на Raspberry PI 3 B та запущений в домашніх умовах з провайдеру

мережі який не надає потрібних ресурсів для забезпечення такого роду інтернет-сервісів, особливо для більших масштабів.

Щодо Apache JMeter то результати відрізняються, проте не сильно. Параметри тестування відрізнялись, а саме кількість паралельних запитів дорівнювала 20, що вдвічі більше ніж використовується для Apache Benchmark.

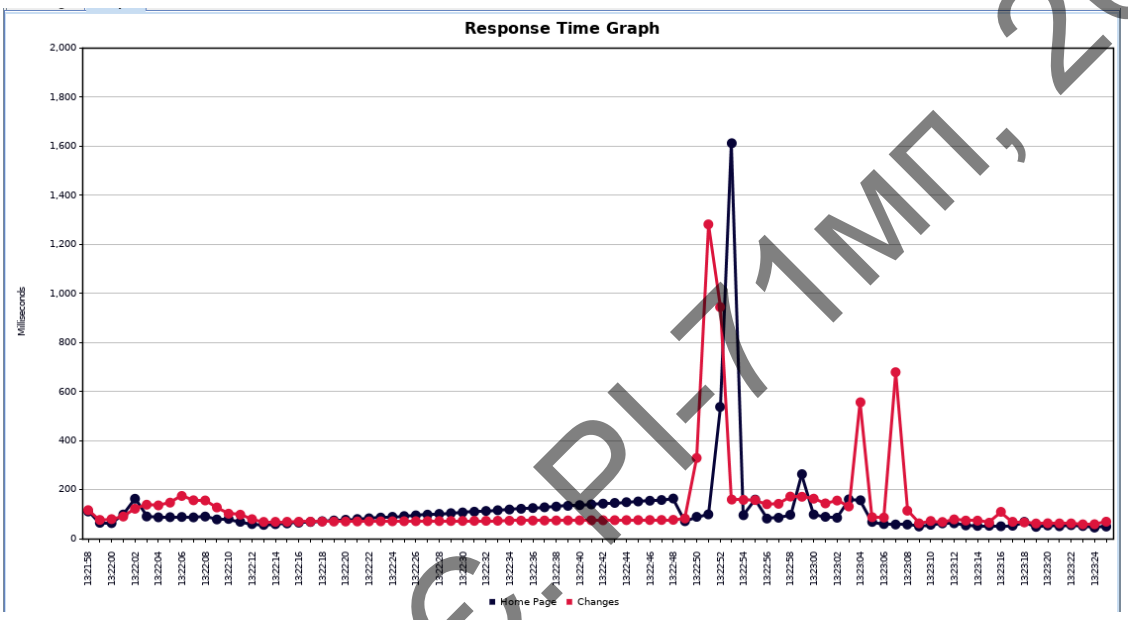

Рисунок 4.17 — Графік швидкодії обох сторінок

На рисунку 4.17 чорна лінія – запит домашньої сторінки, червона – запит сторінки з проектом.

Так як значення часу запиту сторінки проекту сильно неспівпадають з тими які були отриманні в Apache Benchmark не можна казати що вони є достовірними. Проте графік на запит домашньої сторінки більш схожий на результати отримані раніше, тому можна сказати що домашня сторінка отримується в деяких випадках швидше ніж за 100 мс.

При виконанні цих тестів також стало відомо що при цих тестах програмне забезпечення не використовує більше ніж 30МБ оперативної пам’яті, що не є чудовим показником, але якщо зважати на те що код сервісу на теперішний час не містить будь яких оптимзацій – результат більш ніж прийнятний.

Після проведення тестів була проведена невелика оптимізація проекту, що зважаючи на нескладність зміни принесла видимі результати покращення щвидкодії. Оптимізація полягає в тому що раніше статистика отримувалась з бази даних трьома запитами кожного разу як користувач робив зарит на сторінку, а після оптимізації ця інформація береться без звернень до бази даних за допомогою існуючих змінних у програмі. При запуску сервісу одноразово відбуваються запити в базу даних, які повертають кількість проектів, звернень та коментарів та записуються в внутрішні змінні, після цього для отримання статистики не потрібно більше звернень до бази даних. Коли користувач стоврює проект, пише відгук або коментар то до відповідної зміної додаться одиниця і вже цей результат буде показуватись користувачу при наступному запиті будь якої сторінки. Також при видалені проектів, відгуків та коментарів значення відповідних зміних.

За допомогою цього зменшується потреба в ресурсах диску та процесорі і збільшується швидкодія сервісу. За попередніми розрахунками час на отримання головної сторінки проекту та сторінки авторизації користувача зменшився більш ніж на 40 відсотків, сторінки які містять інші запити до бази даних не отримали такого пришвидшення, проте час отримання сторінки також зменшився до 25 відсотків. Ці результати отримані локально, тобто завантаження стороніх ресурсів при цьому не відбувалось.

Після оптимізації були проведені повторні виміри. Починаючи з тесту на отримання головної сторінки ресурсу.

```
ffenix@ffenix-GE62-6QF:~/GoglandProjects/Acapella$ ab -q -c 10 -n 2000 https://acapella.pp.ua/
This is ApacheBench, Version 2.3 <$Revision: 1807734 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking acapella.pp.ua (be patient).....done


Server Software:        cloudflare
Server Hostname:        acapella.pp.ua
Server Port:            443
SSL/TLS Protocol:       TLSv1.2,ECDHE-ECDSA-CHACHA20-POLY1305,256,256
TLS Server Name:        acapella.pp.ua

Document Path:          /
Document Length:        4958 bytes

Concurrency Level:      10
Time taken for tests:   20.620 seconds
Complete requests:      2000
Failed requests:        0
Total transferred:      11240000 bytes
HTML transferred:       9916000 bytes
Requests per second:    96.99 [#/sec] (mean)
Time per request:       103.099 [ms] (mean)
Time per request:       10.310 [ms] (mean, across all concurrent requests)
Transfer rate:          532.33 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        6   15  56.9     10    1038
Processing:    39   88  27.9     86     335
Waiting:       39   87  27.9     85     335
Total:         49  103  63.6     97    1169

Percentage of the requests served within a certain time (ms)
  50%     97
  66%    109
  75%    115
  80%    120
  90%    132
  95%    144
  98%    164
  99%    215
 100%   1169 (longest request)
ffenix@ffenix-GE62-6QF:~/GoglandProjects/Acapella$
```

Рисунок 4.18 — Швидкодія головної сторінки після оптимізації

З цього результату на рисунку 4.18 можна побачити що мережа працювала повільніше ніж під час першого тесту, проте мала більшу стабільність, через що кількість запитів в секунду має більш реальний вигляд: 97 запитів на секунду, з середнім часом відповіді 103 мс. Такий час повністю задовільняє умови технічного завдання та ще після низки покращень можна отримати ще кращі результати.

Результат отримання сторінки проекту також зазнав покращень, що можна побачити на рисунку:

```
ffenix@ffenix-GE62-6QF:~/GoglandProjects/Acapella$ ab -q -c 10 -n 2000 https://acapella.pp.ua/p/acapella
/
This is ApacheBench, Version 2.3 <$Revision: 1807734 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking acapella.pp.ua (be patient).....done


Server Software:        cloudflare
Server Hostname:        acapella.pp.ua
Server Port:            443
SSL/TLS Protocol:       TLSv1.2,ECDHE-ECDSA-CHACHA20-POLY1305,256,256
TLS Server Name:        acapella.pp.ua

Document Path:          /p/acapella/
Document Length:        5132 bytes

Concurrency Level:      10
Time taken for tests:   28.621 seconds
Complete requests:      2000
Failed requests:        0
Total transferred:      11588000 bytes
HTML transferred:       10264000 bytes
Requests per second:    69.88 [#/sec] (mean)
Time per request:       143.105 [ms] (mean)
Time per request:       14.310 [ms] (mean, across all concurrent requests)
Transfer rate:          395.39 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        7   12  11.0     10     249
Processing:    62  131  28.4    130     368
Waiting:       61  130  28.0    129     367
Total:         72  143  30.4    141     404

Percentage of the requests served within a certain time (ms)
  50%    141
  66%    150
  75%    156
  80%    160
  90%    171
  95%    186
  98%    203
  99%    246
 100%    404 (longest request)
ffenix@ffenix-GE62-6QF:~/GoglandProjects/Acapella$
```

Рисунок 4.19 — Швидкодія сторінки проекту після оптимізації

Оптимізація покращила результат і тепер сервіс може обробити 70 запитів на секунду, що видно з рисунку 4.19, з середнім часом відповіді в 143 мс. Зважаючи на те що кожна секунда витрачена на очікування сторінки може значно погіршити сервіс в очах користувача та в певних ситуаціях винудити його перестати взагалі ним користуватись – будь які покращення в доступності ресурсу не залишаться непоміченими з боку користувачів, а також і адміністраторам ресурсу, через меншу кількість яка потрібна сервісу для роботи можна значно скоротити рахунок на обладнані та кількості переданих даних.

Щодо отриманих значень з Apache JMeter то виконання тестування після оптимізації вони стали точнішими, що видно з рисунку.

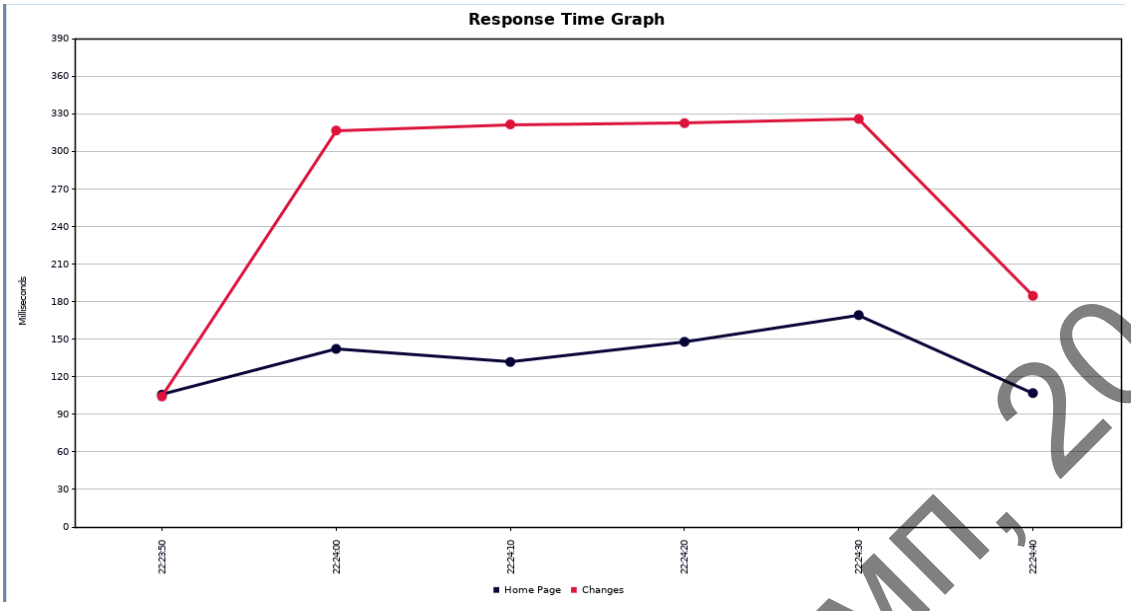

Рисунок 4.20 — Графік швидкодії після оптимізації

На рисунку 4.20 видно що при 20 паралельних з’єднаннях час отримання відповіді з домашньої сторінки склав не більше 170 мс і в середньому тримався на рівні менше 150 мс, а сторінка з проектом отримується швидше ніж за 330 мс.

Зважаючи на всі отримані значення після тестування можна прийти до висновку що умови технічного завдання виконані і сервіс має достатню швидкодію для повсякденного користування. Окрім цього є великий простір для оптимізації проекту як за допомогою змін в коді, так і за допомогою використання сторонніх сервісів які прискорюють завантаження файлів з сторонніх сервісів.

5 РОЗРОБЛЕННЯ СТАРТАП ПРОЕКТУ

5.1 Опис ідеї проекту

5.1.1 Зміст ідеї

Останнім часом важливим фактором у роботі будь якого сервісу є задоволеність користувача наданими послугами. Причиною тому є велика кількість необмеженої конкуренції на ринку. Досить просто наразі створити компанію яка буде мати будь який профіль роботи і при цьому така компанія може нічим не поступатись іншим гравцям ринку. Саме для того щоб зберегти клієнтів власники сервісів мають дослуховуватись до думок користувачів, або в інакшому випадку відкритість ринку та велика кількість конкуренції зробить свою справу і користувач отримає послугу в іншому сервісі. Саме тому доцільно мати можливість отримувати зворотній зв’язок користувачів як важливий метод покращення працюючого сервісу який і надалі хоче продовжувати покращувати свої показники.

Сутність ідеї магістерської дисертації є розробка сервісу отримання, накопичення та обробки відгуків користувачів інтернет-ресурсів, що дозволить уникнути проблем для інтернет-ресурсів при розробці власного сервісу, зменшить навантаження на персонал інтернет-ресурсу через непотрібність опрацьовувати велику кількість вхідних повідомлень з однаковими зверненнями та підвищить лояльність клієнтів за допомогою відкритості даних які вони надають інтернет-ресурсу.

Впровадження такого ресурсу дозволить отримати багато переваг як для власників ресурсу, так і для користувачів. Включаючи посилання на зворотній зв’язок на інтернет-ресурсі користувач має в будь який момент повідомити власникам ресурсу про проблему чи побажання з приводу роботи ресурсу чи супутніх речей. Також для власника ресурсу це здатність напряму звертатись до своїх клієнтів за допомогою відповіді на

відгуки користувачів та реакції на них. Якщо власник ресурсу виконує побажання користувачів то це значно підвищує якість інтернет-ресурсу в їх очах і в такій ситуації користувачі, задоволені реакцією сервісу будуть рекомендувати його і іншим.

Більш детальна та структурована щодо опису ідеї стартап-проекту, цільові групи споживання наведена в табл. 4.1.

Таблиця 4.1. Опис ідеї стартап-проекту

Зміст ідеї	Цільові групи споживання	Вигоди для користувачів
Розробка системи отримання, накопичення та обробки відгуків користувачів інтернет-ресурсів	Інетренет-ресурси на почактовій стадії розробки та роботи	− швидке отримання пропозицій користувачів щодо подальшого розвитку
	Невеликі компанії	− реальні відгуки користувачів щодо роботи інтернет-ресурсу − можливість отримання як пропозицій щодо покращення так і відгуків щодо виконаних транзакцій

	Промислові компанії та розвинені інтернет-ресурси	− відгуки щодо різних продуктів та якості обслуговування − отримання вектору для змін та нового функціоналу
	Торгові точки	− відгуки щодо конкретних точок продажу та цілих міст − можливість дізнатись потреби користувачів в залежності від місця розташування

Напрям використання проекту, що описується, розповсюджується на інтернет-ресурси будь якої складності та статусу виконання, що застосовуються для проектів які розвиваються, сталих ресурсів та торгових представництв. До таких ресурсів відносяться персональні проекти, форуми та соціальні мережі, онлайн-магазини будь якого розміру та торгові заклади які мають інтернет-ресурси.

Продукт за задумом. Реалізація ідеї включає в себе реалізацію самого сервісу отримання, накопичення та обробки звернень користувачів інтернет-ресурсів.

Сервіс у вигляді інтернет-ресурсу дозволить користуватись нею з різних пристроїв та додатків, що підвищує універсальність та виключає потребу для використання конкретної платформи користувачами сервісу.

5.1.2 Потреби, що задовольняються

Головна мета розроблюваного проекту — підвищити рівень захисту

ав- торського ПЗ від несанкціонованого використання за рахунок підвищення рівня безпеки при використанні ліцензії. Це буде досягнуто шляхом викорис- тання поєднання апаратного методу захисту й ліцензування ПЗ, при якому ліцензія ПЗ може використовуватися лише при наявності апаратного ключа (апаратної частини захисту). Потреби, що задовольняються, описані в табл. 4.2.

Таблиця 4.2 – Потреби, що задовольняються

№	Сутність
1	Отримання відгуків користувачів
2	Можливість встановлення статусу відгуку
3	Можливість отримання пропозицій, скарг та відгуків
4	Модерація за допомогою функціоналу сервісу
5	Створення коментарів до відгуків
6	Блокування та/або видалення поганих відгуків
7	Інтеграція до будь якого інтернет-ресурсу

5.2 Аналіз ринкових можливостей

5.2.1 Конкуренти

На українському ринку прямих конкурентів немає через те що запропонований сервіс не потребує оплати для роботи.

До непрямих конкурентів які засновані або мають офіси в Україні можна віднести такі сервіси як Yaware.Service, SendSMS та Revizion. Серед компаній які більше спрямовані на іншомовних користувачів можна визначити Reformal.ru, UserEcho та UserVoice

Більшість компаній пропонують відкриті відгуки, проте отримати такий сервіс можливо тільки після оплати послуг. Сервіс Reformal.ru дозволяє отримувати відкриті відгуки безкоштовно і тому він вважається основним конкурентом. Детальна інформація про конкурентів наведена в табл.4.3.

Конкурент	Відмінності	Переваги/недоліки розроблюваної ідеї
Yaware.Service	Більше спрямований на фізичні об’єкти господарювання	Використовує сервіс Aha! для роботи з відгуками Працює на платній основі

SendSMS	Працює на основі відправки СМС повідомлень	Немає відкритих відгуків. Працює на платній основі через повідомлення
Revizion	Має можливість надавати подяку за відгуки	Є відкриті відгуки Використовується на платній основі
UserVoice	Нерозповсюджений в Україні та близьких країнах	Мало інформації на офіційному ресурсі про цінову політику Має відкриті відгуки
Reformal.ru	Можливість більшої інтеграції з інтернет-ресурсами	Графічний інтерфейс користувача застарів Безкоштовний у використанні

Велика кількість ресурсів створюється кожного дня і кожен ресурс прагне бути кращим за своїх конкурентів. Саме цю задачу мають виконувати сервіси зворотного зв’язку, які надають інформацію авторам проекту по задоволеності користувачів їх сервісом, і саме тому ринок

такого програмного забезпечення розширюється з кожним днем.

5.2.2 Товари - замінники

Основний товар-замінник — це сервіс Reformal.ru. Серед інших сервісів-замінників є ресурси українського та англійського походження такі як Revizion та UserEcho, але вони потребують помісячної оплати для їх роботи.

Напрямок захисту ПЗ від несанкціонованого використання був започат- кований зі стрімким розвитком дорогого ПЗ, до якого відносяться різномані- тні промислові системи, конструкторські системи та інше. За останніх 2 роки на території України відбувся скачок за рахунок збільшення кількості інозе- мних замовників. За рахунок цього кількість товарів — замінник різко зрос- тає, але вони всі відрізняються своєю функціональністю.

5.2.3 Постачальники та партнери

Постачальниками є виробники готових USB модулів, виробники друко- ваних плат та пластикових/алюмінієвих корпусів для електроніки. Основна ідея— створювати модулі на основі вже готових компонентів. Можливість підбору партнерів базується лише на вартості послуг, що партнер пропонує.

Існує два можливих варіанти підбору постачальників. Перший варіант базується на підвищенні локалізації виробництва на території України з ме- тою підвищити промисловий потенціал країни. Виробництво друкованих плат налагоджено в Україні, тому локалізація виробництва складає 100%. Виробництво корпусів також має локалізацію виробництва 100%. USB моду- лі не виготовляються на даний момент на території України. Тому це головна ланка, яка має бути впроваджена у виробництво.

Другий полягає на виборі іноземних постачальників, коли за основу бе- реться вартість комплектуючих. Основним іноземним постачальником

— є китайські виробники електроніки.

Розробка програмної частини проекту повністю базується в Україні, що знижує собівартість та підвищує якість виконуваних функцій.

Партнери — в основному китайські виробники електроніки, так як вони є найбільш гнучкими в прийнятті нових рішень.

За останні 2 роки в Україні зросло виробництво електроніки за рахунок розвитку ВПК, що вплине на локалізацію виробництва друкованих плат та корпусів у кращу сторону.

5.3 Реалізація проекту

5.3.1 Фінансування проекту

Головним питанням, що постає перед розробниками стартап — проекту є його фінансування. Для реалізації даного проекту початкове фінансування полягає лише в тому, щоби реалізувати мінімально потрібний функціонал. Але основне фінансування повинно бути витрачене на розроблення програмного забезпечення з метою реалізації прототипу та подальше його покращення. Також для просунення сервісу потрібно мати фінансування для реклами.

Першочергове фінансування повинно бути сформоване засновниками проекту для розробки прототипу сервісу. Після створення найменшого прототипу потрібно провести дослідження щодо іншого потрібного функціоналу клієнтам та заохотити їх користуванням запропонованим сервісом. У разі успішності, потрібно залучати інвесторів для залучення більшої кількості співробітників.

5.3.2 Персонал і команда

Важливу роль у формуванні стартапу посідає команда. Внутрішня організація команди повинна бути оформлена таким чином, щоби не потрібно було мотивувати кожного члена команди окремо. Тобто, кожен

член команди має бути зацікавлений у подальшому розвитку сервісу.

Персонал команди підібраний таким чином, що він повністю складається з розробників програмного забезпечення. Команда налічує 1-2 особи. Цього достатньо на першому етапі для формування прототипу, подальшого вдосконалення та покращення проекту.

Початкова команда (1-2 людини) — це люди, які в перспективі створять свою компанію й наберуть більший штаб персоналу. Тому, необхідна злагоджена робота між членами команди.

Сучасні інструменти просування. До сучасних інструментів просування потрібно віднести інтернет-маркетинг, який включає в себе контекстну рекламу. Але на перших етапах у цьому не буде необхідності, адже основна ідея — запропонувати використовувати сервіс існуючим компаніям що розвиваються. Тобто, реклама проекту буде відбуватися на першому етапі як презентація серед інших стартапів, що мають свій програмний продукт або фізичні об’єкти з сайтом.

На другому етапі просування, потрібно підготувати сервіс для поширення серед більш масштабних компаній таких як невеликі магазини, інтернет-сервіси, мережі швидкого харчування та інших компаній, які можуть бути (потенційно) зацікавлені в отриманні відгуків користувачів для покращення існуючого бізнесу.

Інші сучасні інструменти просування можуть бути використані лише з метою презентації перед розробниками ПЗ на певних виставках та конференціях. Це сприятиме подальшому впровадженні цього проекту в реалізацію.

5.3.3 Перспективи реєстрації

Реєстрація / патентування розробки відіграє важливу роль у сучасних ринкових відносинах. Зважаючи на тенденції використовувати чужі напрацювання, усі розробки повинні бути запатентовані щонайменше в

трьох країнах світу: Україні, Китаї і США.

Компанія повинна бути зареєстрована в США, як найбільш сприятливій країні для розвитку бізнесу. В Україні, Китаї, Німеччині, Ізраїлі, Чехії повинні бути відкриті філіали компанії з метою впровадження розробок в існуючу промисловість.

На початкових етапах реєстрація компанії не є першочерговою метою. Більшість молодих стартапів розвиваються в країнах, де вони були винайдені. Подальший ріст відбувається в більш економічно сприятливих країнах, таких як США, Естонія.

На даному етапі Естонія є дуже сприятливою для реєстрації компанії, що пов'язане з дуже розвиненим сегментом підприємницької діяльності.

5.4 Висновки по розділу

З погляду конкурентоспроможності запропонована ідея може бути монетизована. Також як приклад отримання грошей на утримання сервісу можна зазначити пожертви, які користувачі будуть надавати самостійно Рентабельність роботи може бути оцінена лише після реалізації прототипу через безкоштовну модель роботи.

За рахунок великої кількості інтернет-сервісів, даний сервіс може бути впроваджений в дуже широкому спектрі ресурсів та компаній які прагнуть покращити свої продукти.

Як альтернатива, можна зупинитися на розробці ще простішого сервісу, за допомогою більшого використання сторонніх ресурсів. У подальшому, це полегшить розробку та покращення даної ідеї.

Подальша імплементація проекту є доцільною через низьку конкуренцію та широкі можливості для покращення сервісу та отримання прибутків.

6 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

Враховуючи специфіку даної науково-дослідницької роботи, яка має суто теоретичний характер, і виконана із застосуванням засобів обчислювальної техніки, у даному розділі в урахуванням вимог ДСТУ 9241:6-2004,ДСанПіН 3.3.2.007-98 «Державні санітарні правила й норми роботи з ВДТ ПЕОМ» та ДНАОП 0.00-1.31-99визначені основні потенційно шкідливі й небезпечні виробничі фактори, що виникають при роботі з електронними та обчислювальними пристроями в науково-дослідній лабораторії радіотехнічного факультету НТУУ «КПІ».

У даному розділі і запропоновані відповідні технічні рішення та організаційні заходи щодо створення комфортних та безпечних умов праці користувачів ВДТ ПЕОМ, а також розглянуті заходи з безпеки в надзвичайних ситуаціях.

6.1 Визначення основних потенційно шкідливих виробничих факторів при виконанні науково – дослідницької роботи

При роботі на ПК людина піддається впливу іонізуючого, інфрачервоного й ультрафіолетового випромінювань екрана монітора, рівні яких повинні відповідати вимогам ДСанПіН 3.3.2.007-98 «Державні санітарні правила й норми роботи з ВДТ ПЕОМ» та ДНАОП 0.00-1.31-99.

При проведенні повного циклу робіт із проектування потенційно – небезпечними та шкідливими факторами можуть бути:

- наявність електромагнітного випромінювання;
- можливість ураження електричним струмом;
- невідповідність освітлення робочої поверхні санітарним нормам;
- несприятливі мікрокліматичні умови;

- високе розумове навантаження;
- можливість виникнення пожежі та надзвичайних ситуацій.

6.2 Технічні рішення та організаційні заходи щодо створення комфортних та безпечних умов праці користувачів ВДТ ПЕОМ

ДСТУ 9241:6-2004 та ДСанПіН 3.3.2.007-98 «Державні санітарні норми і правила роботи з візуальними дисплейними терміналами (ВДТ) електронно- обчислювальних машин» встановлюють норми щодо забезпечення охорони праці користувачів ПК. Дотримання вимог цих правил може значно знизити наслідки несприятливої дії на працівників шкідливих та небезпечних факторів, які супроводжують роботу з відео-дисплейними матеріалами, зокрема можливість зорових, нервово-емоційних переживань, серцево-судинних захворювань.

Для того щоби забезпечити точне та швидке зчитування інформації в зоні найкращого бачення, площина екрана монітора виставлена перпендикулярно нормальній лінії зору. При цьому передбачена можливість переміщення монітора навколо вертикальної осі в межах ±30° (справа наліво) та нахилу вперед до 85° і назад до 105° з фіксацією в цьому положенні. Клавіатура розміщена на поверхні столу на відстані 100-300 мм від краю. Кут нахилу клавіатури до столу обрано в межах від 5 до 15° так, що зап'ястя на долонях рук розташовуються горизонтально до площини столу. Таке положення клавіатури зручне для праці обома руками.

Робочі місця з ПК розташовано відносно від стіни з вікнами на відстані не менше 1,5м, від інших стін — на відстані 1 м, відстань між собою - не менше ніж 1,5 м. Причому так, щоби природне світло падало збоку, переважно зліва. Для захисту від прямих сонячних променів, які створюють прямі та відбиті відблиски з поверхні екранів ПК передбачені сонцезахисні жалюзі.

Штучне освітлення робочого місця, обладнаного ПК, здійснюється системою загального рівномірного освітлення. Як джерело штучного освітлення мають застосовуватись люмінесцентні лампи ЛБ.

Вимоги до освітлення приміщень та робочих місць під час роботи з ПК:

- освітленість на робочому місці повинна відповідати характеру зорової роботи, який визначається трьома параметрами: об'єктом розрізнення - найменшим розміром об'єкта, що розглядається на моніторі ПК; фоном, який характеризується коефіцієнтом відбиття; контрастом об'єкта і фону;
- необхідно забезпечити достатньо рівномірне розподілення яскравості на робочій поверхні монітора, а також у межах навколишнього простору;
- на робочій поверхні повинні бути відсутні різкі тіні;
- в полі зору не повинно бути відблисків (підвищеної яскравості поверхонь, які світяться та викликають осліплення);
- величина освітленості повинна бути постійною під час роботи.

Тривалість регламентованих перерв під час роботи з ЕОМ становить 10 хвилин через кожну годину роботи .

Для зниження нервово-емоційного напруження, втомленості зорового аналізатора, для поліпшення мозкового кровообігу й запобігання втомі доцільно деякі перерви використовувати для виконання комплексу вправ, які передбачені ДСанПіН 3.3.2.007-98.

6.2.1 Електробезпека

Згідно ОНТП24-86 та ПУЕ науково-дослідницька лабораторія відноситься до приміщень без підвищеного ризику. Електроустаткування належить до приладів до 1000 В. Устаткування, що використовується, відповідно до ГОСТ 12.2.007.0-75 належить до устаткування класів 0I, I

таII за електрозахистом.

У процесі експлуатації електронно-обчислювального обладнання людина може доторкнутися до частин електроустаткування, які перебувають під напругою. Оцінка небезпеки дотику до струмоведучих частин відноситься до визначення сили струму, що протікає через тіло людини, і порівняння його із допустимим значенням відповідно до ГОСТ 12.1.038-88. У загальному випадку допустима величина струму, що протікає через тіло людини, залежить від схеми підключення електроустаткування до електромережі, роду й величини напруги живлення, схеми включення.

При виконанні розрахунків для дипломного проекту використовувався персональний комп'ютер - I (системний блок) і II (ВДТ ПЕОМ) клас за електрозахистом, що живиться напругою 220 В. Для правильного визначення необхідних засобів та заходів захисту від ураження електричним струмом необхідно знати допустимі значення напруг доторкання та струмів, що проходять через тіло людини.

Напруга доторкання - це напруга між двома точками електричного кола, до яких одночасно доторкається людина. Гранично допустимі значення напруги доторкання та сили струму для нормального (безаварійного) та аварійного режимів електроустановок при проходженні струму через тіло людини по шляху «рука – рука» чи «рука – ноги» регламентуються ГОСТ 12.1.038-88.

Таблиця 6.1. Гранично допустимі значення напруги доторкання $U_{доп}$ та сили струму $I_л$, що проходить через тіло людини при нормальному режимі електроустановки

Вид струму	$U_{доп}$, В(не більше)	$I_л$, мА (не більше)
Змінний, 50 Гц	2	0,3
Змінний, 400 Гц	3	0,4
Постійний	8	1,0

Гранично допустимі значення сили струму (змінного та постійного), що проходить через тіло людини при тривалості дії більше ніж 1 с нижчі за порогів невідпускаючого струму, тому при таких значеннях людина, доторкнувшись до струмопровідних частин установки, здатна самостійно звільнитися від дії електричного струму.

Таблиця 6.2. Гранично допустимі значення напруги доторкання $U_{доп}$ та $I_л$, що проходить через тіло людини при аварійному режимі електроустановки

Вид струму	Нормоване значення	Тривалість дії струму t, с					
		0,1	0,2	0,5	0,7	1,0	>1,0
Змінний, 50 *Гц*	$U_{доп}$, В	500	250	100	70	50	36
	$I_л$, мА	500	250	100	70	50	6
Постій-ний	$U_{доп}$, В	500	400	250	230	200	40
	$I_л$, мА	500	400	250	230	200	15

Основними технічними засобами, що забезпечують безпеку робіт (згідно ПУЕ-87, ГОСТ 12.1.009-76) є: надійна ізоляція, захисне заземлення, занулення, захисне відключення, засоби індивідуального

захисту. У системі трифазних мереж із глухо заземленою нейтраллю, яка використовується в науково- дослідницькій лабораторії, найкращими засобами захисту є: надійна ізоляція струмоведучих частин електроустаткування відповідно до ГОСТ 12.1.009-76 і занулення відповідно до ПУЕ (з'єднання елементів, що перебувають під напругою, із глухо заземленою нейтраллю). Крім того, для заземлення переносних частин обладнання застосовують спеціальне з'єднання.

Розрахуємо ланцюг захисного відключення фазного проводу при аварійному режимі роботи електрообладнання . Струм КЗ можна обчислити за формулою:

$$I_{КЗ} = \frac{U_Ф}{R_0 + R_Ф + Z_{ТР}}$$

де $U_Ф = 220\ В$ - напруга фазного проводу;

$R_0 = 3\ Ом$ - опір нульового проводу;

$R_Ф = 7\ Ом$ - опір фазного проводу;

$Z_{ТР} \cong 0{,}1\ Ом$ - еквівалентний опір трансформатора.

$$I_{КЗ} = \frac{220}{3 + 7 + 0.1} = 21.78\ A$$

Струм спрацьовування автоматів захисту з електромагнітним розпилювачем повинен бути в 1,4 рази менше струму короткого замикання (при $I_{КЗ} \leq 100$ А).

$$I_{СПР} = \frac{21.78}{1.4} = 15.6\ A$$

Таким чином, струм спрацьовування автомата повинен бути менше 15,6 А. Автомати максимально струмового захисту, встановлені в науково-дослідницькій лабораторії задовольняють цим умовам, оскільки мають

$I_{СПР} = 10$ А. Розрахуємо напругу дотику до корпусів електрообладнання при короткому замиканні:

$$U_{ДОТ=}I_{КЗ}R_0 = 21{,}78 \cdot 3 = 65{,}34\ В$$

Відповідно до ГОСТ 12.1.038-88, щоби ця напруга була безпечна для людини, необхідно використовувати автомати максимально струмового захисту у яких час спрацьовування менше 0,8с (автомати встановлені в лабораторії мають $t_{СПР}$ < 0.1 с).

Із проведених розрахунків видно, що в науково-дослідницькій лабораторії основним захистом від поразки електричним струмом є занулення та застосування пристроїв максимального струмового захисту.

6.2.2 Освітлення робочих місць користувачів ВДТ ПЕОМ

Штучне освітлення, а саме відсутність у спектрі ламп денного світла й ламп накалювання біологічно активної ультрафіолетової складової при тривалому впливі може призвести до ультрафіолетової недостатності, при якій знижуються бактерицидні властивості шкіри та імунітет.

Істотне значення для збереження тривалої працездатності, підвищення продуктивності праці має забезпечення норм освітленості на робочому місці. Величина освітленості регламентується нормами ДБН В.2.5-28-2006. Робоче приміщення належить до I групи — приміщення, у яких розрізнення об'єктів зорової роботи здійснюється при фіксованому напрямку лінії зору працюючих на робочу поверхню.

Нормування штучного освітлення також здійснюється згідно ДБН В.2.5- 28-2006. Для загального освітлення використовують головним чином люмінесцентні лампи, що обумовлено їхніми перевагами. Для розрахунку штучного освітлення застосовують метод коефіцієнта використання потоку:

$$\Phi = \frac{E * K * S * Z}{N * C}$$

де Φ – світловий потік; E– нормована мінімальна освітленість; K– коефіцієнт запасу; S– освітлювана площа; Z– коефіцієнт нерівномірності освітлення; C– коефіцієнт використання випромінюваного світильниками

світлового потоку на розрахунковій площі; N– число світильників.

Згідно ДБН В.2.5-28-2006 визначаємо норму освітленості:

$$E = 300 \text{ лк}; K = 1{,}5; S = 5 \cdot 4 = 20 \text{ м}^2; Z = 1{,}2.$$

Необхідна кількість люмінесцентних ламп визначається по формулі:

$$N = \frac{E * K * S * Z}{\Phi * C}$$

Найбільш прийнятними для приміщення є люмінесцентні лампи ЛД (денного світла) потужністю 40 Вт. Нормальний світловий потік лампи ЛД40 дорівнює $\Phi = 2340$ лм. Величиною і, індексом приміщення можна встановити залежність від площі приміщення й висоти підвісу:

$$i = \frac{A * B}{h * (A + B)}$$

де $A = 5$м – довжина приміщення; $B = 4$м – ширина приміщення; h – висота підвісу;

$$h = H - hp - hc,$$

де $H = 3{,}5$м – висота приміщення; $hp = 0{,}8$м – висота робочої поверхні;

$hc = 0{,}4$м – висота від стелі до нижньої частини лампи;

$$h = 3{,}5 - 0{,}8 - 0{,}4 = 2{,}3 \text{ м};$$

$$i = \frac{4 * 5}{2.3 * (4 + 5)} = 0.97$$

Коефіцієнт використання світлового потоку на розрахунковій площі $C = 0{,}3$. У підсумку число світильників вийде рівним:

$$N = \frac{300 * 1.5 * 20 * 1.2}{2340 * 0.3} = 1.5$$

Для штучного освітлення в робочому приміщенні достатньо використати 2 люмінесцентні лампи денного світла ЛД – 40, зі світловим потоком $\Phi = 2340$ лм кожна.

6.2.3 Мікроклімат робочої зони

Мікроклімат у виробничих умовах визначається наступними параметрами: температурою повітря, відносною вологістю повітря, швидкістю руху повітря й інтенсивністю теплового випромінювання на робочому місці, температурою поверхні.

Для забезпечення нормального мікроклімату в робочій зоні «Санітарні норми мікроклімату виробничих приміщень» ДСН 3.3.6.042-99 встановлюють оптимальне і припустиме значення параметрів мікроклімату залежно від періоду року й категорії робіт. У таблиці наведені оптимальні і припустимі значення параметрів мікроклімату для категорій тяжкості робіт «Іа» (роботи, виконувані сидячи й не потребуючі фізичної напруги при витраті енергії не більше 120 ккал/година).У приміщенні використовується 6-ти секційна чавунна батарея центрального опалення для підтримки нормальної температури повітря в холодну пору року. Також є кондиціонер для підтримки постійної температури в приміщенні.

Для підтримки необхідних параметрів повітря в приміщенні використовується природна вентиляція. У приміщенні є вікно, яке можна відкривати, якщо буде потреба в теплий період і кватирка відкривається в холодний період. Шкідливі речовини в приміщенні не зберігаються й не використовуються.

Фактичні параметри мікроклімату в робочій зоні відповідають приведеним вище нормам ДСН 3.3.6.042–99.

Таблиця 6.3. Оптимальні й допустимі параметри (для постійного робочого місця) мікроклімату в приміщенні

Параметри	Холодний період		Теплий період	
	оптимальні	допустимі	оптимальні	допустимі
Температура ^{0}С	22-24	21-25	23-25	22- 28
Відносна вологість, %	40-60	80	40-60	75
Швидкість руху повітря, м/с	0,1	0,1	0,1	0,1-0,2

6.2.4 Заходи щодо поліпшення умов праці в науково-дослідній лабораторії

На підставі проведеного аналізу рівня електроживлення у виробничому приміщенні можна зробити висновок про доцільність застосування повторного заземлення нульового проводу електромережі, що дозволяє зменшити напругу дотику, як при нормальному так і при аварійному режимах роботи електрообладнання.

Правила улаштування електроустановок (ПУЕ) обмежують найбільші опори заземлення, $R_Д$: при сумарній потужності генераторів або трансформаторів у мережі живлення не більше 100кВт або 100 кВА – 10 Ом;

в інших випадках – 4 Ом;

При використанні штучного заземлення повинна виконуватись умова $R < R_Д$.

Нехай тип заземлення – вертикальний електрод – стержень довжиною 4м і діаметром d = 0.01м. Питомий опір ґрунту ρ - 100$Ом \cdot м$ (для суглинку). З урахуванням кліматичного коефіцієнта Ф = 2, маємо:

$$\rho_{РОЗРАХ} = \rho \cdot Ф = 200\ Ом \cdot м$$

Визначимо опір розтікання струму заземлювачів:

$$R = \frac{\rho}{2\pi L} * [\ln\left(\frac{2L}{d}\right) + 0.5 * \ln(\frac{4t + L}{4t - L})]$$

де $\rho = 200\ Ом \cdot м$; $L = 4\ м$; $d = 0{,}01\ м$; $t = 4\ м$. Підставивши числові значення, маємо:

$$R = \frac{200}{2\pi 4} * \left[\ln\left(\frac{2 * 4}{0.01}\right) + 0.5 * \ln\left(\frac{4 * 4 + 4}{4 * 4 - 4}\right)\right] = 55.2\ Ом$$

Таким чином, опір штучного заземлювача більше нормованого значення, тому необхідно паралельно з'єднати декілька однотипних заземлювачів:

$$R_{зв_\Sigma} = \frac{R_1}{n * \eta E}$$

де $\eta_E \approx 0{,}5$– коефіцієнт, що враховує взаємне екранування заземлювача; $n = 20$; $L/a = 1$ (a– відстань між заземлювачами), заземлювачі розташовані по контуру.

$$R_{зв} = 5{,}52\ Ом$$

Довжина сполучної смуги:

$$L_{СПОЛ} = a \cdot n = 4 \cdot 20 = 80\ м$$

Ширина смуги 0.02м, а відстань від смуги до поверхні землі 1м. Тоді опір розтікання струму сполучної смуги:

$$R = \frac{\rho}{2\pi * L_{спол}} * \ln\left(\frac{2 * L_{спол}^2}{B_{спол} * H_{спол}}\right) = \frac{200}{2\pi * 80} * \ln\left(\frac{2 * 80^2}{0.01 * 1}\right) = 5.32\ Ом$$

З урахуванням коефіцієнта використання смуг $\eta_П = 0{,}8$:

$$R_П = \frac{R_П}{\eta_П} = 6.65\ Ом$$

Еквівалентний опір заземлювача складається з паралельно включених $R_П$ і $R_{зв}$:

$$R_{ЕКВ} = \frac{R_П * R_{зв}}{R_П + R_{зв}} = \frac{5.52 * 6.65}{5.52 + 6.65} = 3.1 \text{ Ом}$$

Отримане значення опору заземлювача менше гранично допустимого

6.3 Безпека в надзвичайних ситуаціях

Безпека в надзвичайних ситуаціях регламентується ПЛАС. Основними складовими ПЛАС є розробки технічних рішень та організаційних заходів щодо оповіщення, евакуації та дій персоналу в надзвичайних ситуаціях, а також визначення основних заходів із пожежної безпеки.

6.3.1 Вимоги щодо організацій ефективної роботи системи оповіщення виробничого персоналу в разі виникнення надзвичайної ситуації

Для підвищення безпеки в надзвичайних ситуаціях (НС) пропонується встановлення системи оповіщення (СО) виробничого персоналу.

Оповіщення виробничого персоналу в разі виникнення НС, наприклад при пожежі, здійснюється відповідно до вимог НАПБ А.01.003-2009.

Оповіщення про НС та управління евакуацією людей здійснюється одним із наступних способів або їх комбінацією:

- поданням звукових і (або) світлових сигналів в усі виробничі приміщення будівлі з постійним або тимчасовим перебуванням людей;
- трансляцією текстів про необхідність евакуації, шляхи евакуації, напрямок руху й інші дії, спрямовані на забезпечення безпеки людей;

– трансляцією спеціально розроблених текстів, спрямованих на запобігання паніці й іншим явищам, що ускладнюють евакуацію;

– ввімкненням евакуаційних знаків "Вихід";

– ввімкненням евакуаційного освітлення та світлових покажчиків напрямку евакуації;

– дистанційним відкриванням дверей евакуаційних виходів;

Як правило, СО вмикається автоматично від сигналу про пожежу, який формується системою пожежної сигналізації або системою пожежогасіння. Також із приміщення оперативного (чергового) персоналу СО (диспетчера пожежного поста) слід передбачати можливість запуску СО вручну, що забезпечує надійну роботу СО не тільки при пожежі, а й у разі виникнення будь-якої іншої НС.

Згідно з вимогами ДБН В.1.1-7-2002 необхідно забезпечити можливість прямої трансляції мовленнєвого оповіщення та керівних команд через мікрофон для оперативного реагування в разі зміни обставин або порушення нормальних умов евакуації виробничого персоналу.

Оповіщення виробничого персоналу про НС здійснюється за допомогою світлових та/або звукових оповіщувачів обладнуються всі виробничі приміщення.

СО повинна розпочати трансляцію сигналу оповіщення про НС, не пізніше трьох секунд із моменту отримання сигналу про НС.

Пульти управління СО необхідно розміщувати в приміщенні пожежного поста, диспетчерської або іншого спеціального приміщення (в разі його наявності). Ці приміщення повинні відповідати вимогам пунктів ДБН В.2.5-56- 2014. Пожежна автоматика будинків і споруд".

Кількість звукових та мовленнєвих оповіщувачів, їх розміщення та потужність повинні забезпечувати необхідний рівень звуку в усіх місцях постійного або тимчасового перебування виробничого персоналу.

Звукові оповіщувачі повинні комбінуватися зі світловими, які

працюють у режимі спалахування, у таких випадках:

- у приміщеннях, де люди перебувають у шумозахисному спорядженні;
- у приміщеннях із рівнем шуму понад 95 дБ.

Допускається використовувати евакуаційні світлові покажчики, що автоматично вмикаються при отриманні СО командного імпульсу про початок оповіщення про НС /пожежу/ та (або) аварійному припиненні живлення робочого освітлення.

Вимоги до світлових покажчиків "Вихід" приймаються відповідно до ДБН В.2.5-28-2006 "Інженерне обладнання будинків і споруд. Природне і штучне освітлення".

СО в режимі "Тривога" повинна функціонувати протягом часу, необхідного для евакуації людей із будинку, але не менше 15 хвилин.

Вихід із ладу одного з оповіщувачів не повинен призводити до виведення з ладу ланки оповіщувачів, до якої вони під'єднанні. Електропостачання СО здійснюється за I категорією надійності згідно з ПУЕ від двох незалежних джерел енергії: основного - від мережі змінного струму, резервного - від акумуляторних батарей тощо.

Перехід з основного джерела електропостачання на резервний та у зворотному напрямку в разі відновлення централізованого електропостачання повинен бути автоматичним.

Тривалість роботи СО від резервного джерела енергії в черговому режимі має бути не менш 24 годин.

Тривалість роботи СО від резервного джерела енергії в режимі "Тривога" має бути не менше 15 хвилин.

Звукові оповіщувачі повинні відповідати вимогам ДСТУ EN 54-3:2003 "Системи пожежної сигналізації. Частина 3. Оповіщувачі пожежні звукові". Світлові оповіщувачі, які працюють у режимі спалахування, повинні бути червоного кольору, мати частоту мигтіння в межах від 0,5 Гц

до 5 Гц та розташовуватись у межах прямої видимості з постійних робочих місць.

6.3.2 Обов'язки та дії персоналу в разі виникнення надзвичайної ситуації

У разі виявлення ознак НС працівник, який їх помітив повинен:

- негайно повідомити про це засобами зв'язку органиДержавної Служби з НС (ДСНС), вказати при цьому адресу, кількість поверхів, місце виникнення пожежі, наявність людей, а також своє прізвище;
- повідомити про НС керівника, адміністрацію, пожежну охорону підприємства;
- організувати оповіщення людей про НС;
- вжити заходів щодо евакуації людей та матеріальних цінностей;
- вжити заходів щодо ліквідацій наслідків НС із використанням наявних засобів.

Керівник та пожежна охорона установки, яким повідомлено про виникнення пожежі, повинні:

- перевірити, чи викликані підрозділи ДСНС;
- вимкнути в разі необхідності струмоприймачі та вентиляцію;
- у разі загрози життю людей негайно організувати їх евакуацію, та їх рятування, вивести за межі небезпечної зони всіх працівників, які не беруть участь у ліквідації НС;
- перевірити здійснення оповіщення людей про НС;
- забезпечити дотримання техніки безпеки працівниками, які беруть участь у ліквідації НС ;
- організувати зустріч підрозділів Державної пожежної охорони, надати їм допомогу в локалізації й ліквідації НС.

Після прибуття на НС підрозділів ДСНС повинен бути забезпечений безперешкодний доступ їх до місця, де виникла НС.

6.3.3 Пожежна безпека

Відповідно до НАПБ Б.03.002-2007 та ОНТП24-86 робоче приміщення лабораторії відноситься до категорії В по вибухопожежній і пожежній небезпеці. Відповідно до ПУЕ-87 та ДНАОП 0.00-1.32-01 клас робочих зон приміщення лабораторії по пожежонебезпеці П-IIa. Можливими причинами пожежі в приміщенні є несправність електроустаткування, коротке замикання проводки, і порушення протипожежного режиму (використання побутових нагрівальних приладів, паління).

У зв'язку з цим, відповідно до вимог ПБЕ та ПУЕ, необхідно передбачити такі заходи:

1. Ретельну ізоляцію всіх струмоведучих провідників до робочих місць, періодичний огляд та перевірку ізоляції.
2. Строге дотримання норм протипожежної безпеки на робочих місцях.
3. Відповідні організаційні заходи (заборона паління, інструктаж).

Для гасіння пожежі в робочому приміщені лабораторії (клас пожежі «Е» - наявність електрообладнання під напругою) використовуються вогнегасники ОП-1 — «Момент» (2 шт.). Додатково в коридорі розташовані вогнегасники ОХП-10. Також на сходовій клітці розташований пожежний кран. Така кількість первинних засобів пожежогасіння відповідає вимогам ДСТУ 3675-98

ISO3941-77, якими передбачене обов'язкова наявність двох вогнегасників до 100 м^{2}площі підлоги для приміщення типу конструкторське бюро.

Згідно вимог ДБН В.2.5-56- 2014 робоче приміщення лабораторії необхідно оснастити системою автоматичної пожежної сигналізації.

У науково-дослідницькій лабораторії є план евакуації у випадку виникнення пожежі. Максимальна віддаленість робочих місць від евакуаційних виходів і ширина евакуаційних проходів відповідають вимогам ДБН В.1.1-7- 2002.

Значення основних параметрів шляхів евакуації приведені в табл.5.4.

Таблиця 6.4. Характеристики й норми еваковиходів

Параметр	Фактичне значення	Норма
Висота дверних прорізів	2,0 м	Не менше 2 м
Ширина дверних прорізів	0,8 м	Не менше 0,8 м
Ширина проходу для евакуації	Більше 1,5 м	Не менше 1 м
Ширина коридору	2 м	Не менше 2 м
Число виходів із коридору	2	Не менше 2
Ширина сходового маршу	1,2 м	Не менше 1 м
Висота поруччя сходів	1 м	Не менше 0,9 м

У приміщенні є план евакуації. Мінімальний час евакуації в разі виникнення пожежі, максимальне видалення робочих місць від евакуаційних виходів відповідає вимогам ДБН В.1.1-7-2002.

У робочому приміщенні виконуються всі вимоги НАПБ А.01.001-2004

«Правил пожежної безпеки України».

ВИСНОВКИ

1. У результаті розгляду існуючих сервісів отримання, накопичення та обробки звернеь користувачів інтернет-ресурсів встановлено такі особливості: існуючі аналоги мають конкретно свої унікальні можливості, які і використовуються для надання переваги конкретному сервісу.
2. Запропонована у магістерській роботі система відрізняється відкритістю інформації, можливістю інтеграції до інших інтернет-сервісів та безкоштостовністю користування.
3. Розроблена система на основі пакету gin-gonic/gin, з допомогою пакетів jinzhu/gorm, qor/admin для адміністрування даних та автоматичного створення прикладного програмного інтерфейсу. Графічний інтерфейс розроблений за допомогою бібліотек SemanticUI, JQuery та Toaster.
4. Тестування розробленого сервісу отримання, накопичення та обробки звернень користувачів інтернет-сервісів показало, що всі реалізовані методи розробленої системи працюють коректно, надаючі очікувані результати.
5. Розроблена система може компілюватись для різних операційних систем, відповідно може виконуватись на операційних системах різного сімейства та на мобільних пристроях.
6. Розроблена система завдяки графічному інтерфейсу дозволяє використовувати всі доступні програмні інтерфейси. Графічний інтерфейс дозволяє створювати користувачів, авторизовуватись, додавати проекти та відгуки, голосувати за відгуки та надавати коментарі. З допоміжних речей графічний інтерфейс також дозволяє виконувати модерацію для адміністраторів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. RFC 1321 - The MD5 Message-Digest Algorithm [Електронний ресурс] – Режим доступу до ресурсу: https://tools.ietf.org/html/rfc1321
2. FIPS Publication 180-4, Secure Hash Standard (SHS) [Електронний ресурс] – Режим доступу до ресурсу: https://ws680.nist.gov/publication/get_pdf.cfm?pub_id=910977
3. Federal Information [Електронний ресурс] – Режим доступу до ресурсу: https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.202.pdf
4. RFC 2898 - PKCS #5: Password-Based Cryptography Specification Version 2.0 [Електронний ресурс] – Режим доступу до ресурсу: https://tools.ietf.org/html/rfc2898
5. Introduction [Електронний ресурс] – Режим доступу до ресурсу: https://www.usenix.org/legacy/events/usenix99/provos/provos_html/node1.html
6. phc-winner-argon2/argon2-specs.pdf at master · P-H-C/phc-winner-argon2 [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/P-H-C/phc-winner-argon2/blob/master/argon2-specs.pdf
7. bcrypt – GoDoc [Електронний ресурс] – Режим доступу до ресурсу: https://godoc.org/golang.org/x/crypto/bcrypt
8. Bootstrap · The most popular HTML, CSS, and JS library in the world [Електронний ресурс] – Режим доступу до ресурсу: https://getbootstrap.com/
9. Semantic UI [Електронний ресурс] – Режим доступу до ресурсу: https://semantic-ui.com/
10. Step | Semantic UI [Електронний ресурс] – Режим доступу до ресурсу: https://semantic-ui.com/elements/step.html

11. UIkit [Електронний ресурс] – Режим доступу до ресурсу: https://getuikit.com/
12. Responsive web design - Wikipedia [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Responsive_web_design
13. Где Agile ужасен, особенно Scrum / Хабр [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/post/430890/
14. The Go Programming Language Specification - The Go Programming Language [Електронний ресурс] – Режим доступу до ресурсу: https://golang.org/ref/spec#Semicolons
15. The Go Programming Language Specification - The Go Programming Language [Електронний ресурс] – Режим доступу до ресурсу: https://golang.org/ref/spec#Operators_and_punctuation
16. SQLDrivers · golang/go Wiki [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/golang/go/wiki/SQLDrivers
17. pprof - The Go Programming Language [Електронний ресурс] – Режим доступу до ресурсу: https://golang.org/pkg/net/http/pprof/
18. Effective Go - The Go Programming Language [Електронний ресурс] – Режим доступу до ресурсу: https://golang.org/doc/effective_go.html#embedding
19. Узагальнене програмування — Вікіпедія [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%A3%D0%B7%D0%B0%D0%B3%D0%B0%D0%BB%D1%8C%D0%BD%D0%B5%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F
20. ДСанПіН 3.3.2.007-98 [Електронний ресурс]. — Режим доступу : http://ksv.do.am/publ/dstu/dstu_3798_98/3-3-2-007-98

21. ДНАОП 0.00-1.31-99 [Електронний ресурс]. — Режим доступу : http://otipb.at.ua/_ld/32/3215_bez_vir_proc.pdf
22. ДНАОП 0.00-1.31-99 [Електронний ресурс]. — Режим доступу : http://www.gorsvet.kiev.ua/wpcontent/uploads/2016/08/ДБН-В.2.5-28-2006.pdf
23. ДСН 3.3.6.037-99 [Електронний ресурс]. — Режим доступу : https://ohranatruda.ru/ot_biblio/normativ/data_normativ/6/6472/
24. ГОСТ 12.1.003-2014 [Електронний ресурс]. — Режим доступу: http://docs.cntd.ru/document/1200118606
25. ДСН 3.3.6.042–99 [Електронний ресурс]. — Режим доступу : http://cals.ru/sites/default/files/downloads/3.3.6.042–99.pdf
26. ОНТП24-86 [Електронний ресурс]. — Режим доступу : http://dnaop.com/html/2590/doc-%D0%9E%D0%9D%D0%A2%D0%9F_24-86
27. НАПБ Б.03.002-2007 [Електронний ресурс]. — Режим доступу : https://ohranatruda.ru/ot_biblio/normativ/data_normativ/002/2007/
28. ГОСТ 12.1.038-82 [Електронний ресурс]. — Режим доступу : https://ohranatruda.ru/ot_biblio/normativ/data_normativ/002/2007/
29. ПУЕ-87 [Електронний ресурс]. — Режим доступу : http://www.gosthelp.ru/text/PUEPravilaustrojstvaelekt3.html
30. ГОСТ 12.1.009-76 2011 [Електронний ресурс]. — Режим доступу : http://cals.ru/sites/default/files/downloads/2.702-2011.pdf

ДОДАТОК А

ПОГОДЖЕНО	ЗАТВЕРДЖЕНО
Керівник магістерської дисертації:	Завідувач кафедри
к.т.н., доц. Дюжаєв Л. П.	радіоконструювання та виробництва
Підпис Дата	радіоапаратури
	д.т.н., проф. Нелін Є.А.
	Підпис Дата

ТЕХНІЧНЕ ЗАВДАННЯ
НА МАГІСТЕРСЬКУ ДИСЕРТАЦІЮ
«Система отримання, накопичення та обробки відгуків користувачів інтернет-ресурсів»

1 Підстава для виконання магістерської дисертації

Завдання, видане кафедрою радіоконструювання та виробництва радіоелектронної апаратури.

2 Мета і призначення магістерської дисертації

Тема роботи є актуальною через недостатність у вільному доступі інформації про відгуки користувачів відносно інтернет-ресурсів та сервісів. Система буде реалізована як окремий інтернет-ресурс.

Метою магістерської дисертації є збільшення у вільному доступі відгуків користувачів про інтернет-ресурси, зменшення необхідних ресурсів для обробки відгуків та пришвидшення процедури обробки.

Об'єктом дослідження є система отримання, накопичення та обробки відгуків користувачів (далі система).

Предметом дослідження є аналіз функціональних можливостей створеної системи.

3 Вихідні дані для проведення магістерської дисертації

Найближчий аналог – інтернет-ресурс Reformal.ru.

Основні джерела:

1. Доннован А. Язык программирования Go // Вильямс, 2016 — 432с.
2. Kennedy W. Go in Action // Manning Publications Co., 2015 — 264с.

4 Вимоги до виконання магістерської дисертації

1. Виконати аналіз існуючих рішень
2. Розробити алгоритми та моделі даних для функціонування системи отримання, накопичення та обробки відгуків користувачів
3. Розробити програму в середовищі розробки Goland
4. Розробити графічний інтерфейс
5. Розробити інтернет-ресурс та перевірити його працездатність за допомогою програм Apache Benchmark та Apache JMeter

Основні технічні вимоги:

Платформа: Linux

Мова програмування: Golang

Мінімальні вимоги до ПК: 1 CPU, 128MB RAM, Debian Stretch

Якісні та кількісні показники, що мають бути досягнені в процесі виконання дисертації:

1. Швидкодія: середній час отримання відповіді від серверу не перевищує 400мс, отримання готової сторінки – не більше 1 с.
2. Методика тестування: тести функцій програмного інтерфейсу.
3. Графічний інтерфейс має давати доступ до всіх розділів інтернет-ресурсу та бути простим у використанні. Також інтерфейс має підтримувати мобільні пристрої.
4. Можливість адаптації до інших платформ: забезпечується компілятором мови програмування.
5. Можливість використовувати графічний інтерфейс користувача для адміністрування даних інтернет ресурсу.

5 *Етапи виконання магістерської дисертації*

Таблиця 1 – Етапи роботи

№	*Назва етапів*	*Термін виконання*	*Результат*
1.	Підбір джерел інформації	22.09.17 – 28.11.17	Знаходження та аналіз дослідженої літератури
2.	Виконання аналітичного огляду	28.11.17 – 13.12.17	Структуризація виконання магістерської дисертації
3.	Аналіз аналогів, що використовуються для отримання відгуків користувачів інтернет ресурсів	13.12.17 – 10.01.18	Розділ 2
4.	Розробка алгоритмів та моделей даних на їх основі.	10.01.18 – 20.01.18	Розділ 3
5.	Розробка попередньої версії ресурсу, тестування та додавання функціоналу	20.01.18 – 26.03.18	Розділ 4
6.	Доопрацювання системи, перевірка працездатності, створення інтернет-ресурсу	26.03.18 – 26.07.18	Розділ 5
7.	Створення презентації	26.07.18 – 10.08.18	Додатки
8.	Оформлення магістерської дисертації	12.09.18 – 1.10.18	Магістерська дисертація, презентація

6 *Очікувані результати*

1. Огляд з принципів побудови систем отримання, накопичення та обробки відгуків користувачів інтернет-ресурсів.

2. Моделі даних та алгоритми на їх основі.
3. Програмний та графічний інтерфейс інтернет-ресурсу.
4. Результати тестування.
5. Висновки.

7 Матеріали, що подаються після закінчення магістерської дисертації

Завдання на магістерську дисертацію.

Технічне завдання.

Пояснювальна записка.

Електронна презентація.

8 Порядок приймання магістерської дисертації

1. Поетапне узгодження з керівником.
2. Представлення кафедрі.
3. Попередній захист.
4. Захист на засіданні екзаменаційної комісії.

9 Орієнтовний зміст магістерської дисертації

Вступ

1. Принципи побудови систем отримання, накопичення та обробки відгуків користувачів інтернет-ресурсів
2. Особливості мови програмування Golang
3. Алгоритми та моделі даних системи
4. Графічний інтерфейс. Тестування системи

Виконавець _____ Силенко Є. Є.

ДОДАТОК Б ПУБЛІКАЦІЯ

СИСТЕМА ОТРИМАННЯ, НАКОПИЧЕННЯ ТА ОБРОБКИ ВІДГУКІВ КОРИСТУВАЧІВ ІНТЕРНЕТ-РЕСУРСІВ

Автор Силенко Євген Євгенійович.

(науковий керівник — к.т.н., доц. Дюжаєв Л. П.)

На теперішній час фактичне отримання послуги чи товару від будь якого сервісу або магазину вже перестали бути найважливішим фактором при виборі останніх. Відкритість ринку створила велику кількість конкуренції серед будь яких типів сервісів та магазинів, тому користувач буде шукати де б отримати послугу чи товар за сукупності характеристик, таких як вартість, відношення до покупця(клієнта) та сервіс. Всі ці показники можна покращити, якщо знати що покупці думають про ваш сервіс.

Саме тому великою частиною прибутковості є задоволеність покупців, а дізнатись її можна декількома шляхами:

- За допомогою дзвінків клієнтам
- Прохання після покупки залишити відгук в спеціальній формі
- На сайті(в основному згори або знизу) поставити кнопку «Залишити відгук», яка переводить на спеціальну форму
- Розробка власних сервісів для отримання зворотного зв’язку
- Використання сторонніх сервісів

Розглянемо недоліки та переваги кожного з них.

Дзвінки клієнтам більше набридають і дуже мала частина людей відповість дійсно що потрібно на поставлені запитання через те що в основному такі дзвінки проводяться у незручний час. Також при цьому має витрачатись час персоналу на збір відгуків. Перевагою такого методу є прямий контакт з покупцем та можливість надалі його підтримувати.

Прохання залишити відгук є малоефективними через те що клієнт ще не отримав послугу, або не повністю закінчив процес отримання товару/послуги. До переваг можна віднести лише простоту реалізації.

Кнопка «Залишити відгук» є також малоефективною через те що її використання буде більше спрямоване на скарги, через те що клієнти

будуть шукати спосіб показати своє незадоволення з будь яких причин. Перевагою тут можна назвати лише крайню простоту реалізації.

На розробку власних сервісів компанії в більшості випадків не мають ресурсів, а якщо і мають – створення такого сервісу малоймовірно через малу сферу його застосування. Перевагами в цьому випадку є можливість гнучкого налаштування сервісу під свої потреби та повний його контроль.

Використання сторонніх сервісів мають декілька недоліків які не пов'язані з користувачами: неможливість повністю контролювати ресурс, і відповідно дані на ньому, гнучкість налаштування залежить від обраного сервісу і те що такий сервіс необхідно інтегрувати в свою інфраструктуру. Переваг у такої реалізації найбільше: не потрібно витрачати ресурси на створення або підтримку ресурсу, легкість у використанні та мала кількість людських ресурсів потрібних для обробки відгуків.

Саме тому був розроблений проект Acapella [1], який йде по останньому шляху: це окремий інтернет-ресурс який буде інтегруватись до інших сервісів. Таким чином не потрібно мати реалізації під конкретний сервіс, а відкритість даних значно скоротить потребу в людських ресурсах для найшвидшої реакції на відгуки користувачів, а безкоштовність сервісу дозволить його використовувати навіть невеликим або персональним проектам.

Як найближчий аналог можна розглядати ресурс Reformal.ru [2] який пропонує схожі можливості.

Незважаючи на те що існують вже певні рішення на українському ринку, проте вони не пропонують безкоштовне використання та відкритий доступ до самих відгуків. Такі сервіси ніяким чином не допомагають менше витрачати ресурси, чим суттєво звужують свою аудиторію.

Проект Acapella вже наразі має достатній набір можливостей для отримання зворотного зв'язку: реєстрація користувачів, створення проектів та відгуків в них. Також є голосування за відгуки та їх коментування. Ці можливості вже надають змогу пришвидшити обробку відгуків користувачів та знизити потребу у людському ресурсі для їх обробки.

Можливості для розвитку такого ресурсу є обширними і кожен з них значно спростить використання ресурсом як для користувачів які створюють та модерують проекти, так і для користувачів які голосують, коментують та залишають відгуки. Наприклад планується ввести особистий кабінет для модерації проектів, інтеграцію до сторонніх ресурсів, можливість отримання короткої адреси проекту та інше.

Перелік посилань

1. Acapella | Сервіс зворотнього зв'язку для людей — Режим доступу: https://www.acapella.pp.ua/ — Назва з екрану

2. Reformal » Обратная связь нового поколения, Feedback 2.0 — Режим доступу: http://reformal.ru/ — Назва з екрану

Анотація

Описані характеристики по яким визначається задоволеність клієнтів. Розглянуті шляхи отримання зворотного зв’язку від клієнтів, їх переваги та недоліки. Виведені потрібні параметру сервісу зворотного зв’язку. Описаний розроблений інтернет-ресурс та його функціональні параметри.

Ключові слова: відгуки, сервіс зворотного зв’язку, інтернет-ресурс

Abstract

Described features that determine customer satisfaction. The ways of obtaining feedback from clients, their advantages and disadvantages are considered. The necessary parameters of the service of feedback are obtained. Developed internet resource and its functional parameters were described.

Keywords: feedback, feedback service, internet resource.