Міністерство освіти і науки України

Одеський національний політехнічний університет

Інститут комп’ютерних систем  
Кафедра інформаційних систем

**КУРСОВА РОБОТА**

з дисципліни «Технології створення програмних продуктів»

за темою

«Переналаштуй своє життя - Rewired»

Частина Друга

Виконав:

студент 3-го курсу

групи ЛАІ-185

Глаголев Р.

Перевірив:

Блажко О. А.

Одеса 2020

АНОТАЦІЯ

В курсовій роботі розглядається процес створення програмного продукту «Rewired». В пояснювальній записці у розділах «Проектування» та «Конструювання» детально описано особливості конструювання:

* структур даних в системі керування базами даних MySQL;
* програмних модулів в інструментальному середовищі WordPress з використанням додаткових плагінів та мови програмування PHP.

Результати роботи розміщено на *github*-репозиторіях за адресою:

https://github.com/RomaNNtic/Rewired

ПЕРЕЛІК СКОРОЧЕНЬ

ОС – операційна система

ІС – інформаційна система

БД – база даних

СКБД – система керування базами даних

ПЗ – програмне забезпечення

ПП– програмний продукт

UML – уніфікована мова моделювання

ЗМІСТ

1 Вимоги до програмного продукту…………………………...............................7

1.1 Визначення потреб споживача……………………....................................7

* + 1. Ієрархія потреб споживача……………………..................................7
    2. Деталізація матеріальної потреби……………………......................8

1.2 Бізнес вимоги до ПП……………………....................................................8

1.2.1 Опис проблем користувача…………...................................................8

1.2.1.1 Концептуальний опис проблеми споживача..............................8

1.2.1.2 Метричний опис проблеми користувача....................................9

1.2.2 Мета створення ПП................................................................................9

1.2.2.1 Проблемний аналіз існуючих ПП................................................9

1.2.2.2 Мета створення ПП......................................................................10

1.2.3 Назва ПП……………………………………….....................................10

* + - 1. Гасло ПП…………………………............................................10

1.2.3.2Логотип ПП…………………………………………....................11

1.3 Вимоги користувача до ПП………………………………………..............11

1.3.1 Історія користувача ПП…………………………………...................11

1.3.2 Діаграма прецедентів ПП……………………………………………12

1.3.3 Сценарії використання прецедентів ПП………………………...…..12

1.4 Функціональні вимоги до ПП………………………………………..……19

1.4.1 Багаторівнева класифікація функціональних вимог…………..……19

* + 1. Функціональний аналіз існуючих ПП………………………………20

1.5 Нефункціональні вимоги до ПП…………………………………...……...22

1.5.1 Опис зовнішніх інтерфейсів………………………………………...22

1.5.1.1 Опис інтерфейса користувача…………………………..….…22

1.5.1.1.1 Опис INPUT-інтерфейса користувача……………….…22

1.5.1.1.2 Опис OUTPUT-потоків……………………………...…..25

1.5.1.2 Опис інтерфейсу з зовнішніми пристроями……………...….27

1.5.1.3 Опис програмних інтерфейсів………………………..………27

1.5.1.4 Опис інтерфейсів передачі інформації…………………..…..28

1.5.1.5 Опис атрибутів продуктивності…………………………..….28

2 Планування процесу розробки програмного продукту………………………29

2.1 Планування ітерацій розробки програмного продукту…………….......29

2.2 Концептуальний опис архітектури програмного продукту……………30

* 1. План розробки ПП………………………………………………………..30

2.3.1 Оцінка трудомісткості розробки ПП………………………………30

* + 1. Визначення дерева робіт з розробки ПП…………………………31
    2. Графік робіт з розробки ПП…………………………………...…..34

2.3.3.1 Таблиця з графіком робіт………………………………..……34

2.3.3.2 Діаграма Ганта…………………………………………..……..35

3 Проектування ПП……………………………………………………………..…36

3.1 Концептуальне та логічне проектування структур даних ПП……….....36

3.1.1 Концептуальне проектування на основі UML-діаграми концептуальних класів……………………………………………………….……36

3.1.2 Логічне проектування структур даних…………………..…………37

* 1. Проектування програмних класів………………………………........…...38
  2. Проектування алгоритмів роботи методів програмних класів………....38
  3. Проектування тестових наборів методів програмних класів………...….43

1. Конструювання ПП…………………………………………………………...45

4.1 Особливості конструювання структур даних…………………….…….46

4.1.1 Особливості інсталяції та роботи з СУБД…………………...……46

4.1.2 Особливості створення структур даних……………………...……46

4.2 Особливості конструювання програмних модулів…………………..…48

4.2.1 Особливості роботи з інтегрованим середовищем розробки……48

4.2.2 Особливості створення програмної структури з урахуванням спеціалізованого фреймворку…………………………………………….….49

4.2.3 Особливості створення програмних модулів………....…………..51

4.2.4 Особливості розробки алгоритмів методів програмних класів або процедур/функцій………………………………………………………...…...52

4.3 Тестування програмних модулів…………………………………….…...55

5 Розгортання та валідація ПП………………………………………………..….56

5.1 Інструкція з встановлення ПП……………………………………..…….56

5.2 Інструкція з використання ПП…………………….………………..……56

5.3 Результати валідації ПП………………………………………….……...60

Висновки………………………………………………………………….……….61

Джерела……………………………………………………………………………61

1 ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

* 1. Визначення потреб споживача

1.1.1 Ієрархія потреб споживача

Відомо, що в теорії маркетингу потреби людини можуть бути представлені у вигляді ієрархії потреб ідей американського психолога Абрахама Маслоу включають рівні:

* фізіологія (вода, їжа, житло, сон);
* безпека (особиста, здоров’я, стабільність),
* приналежність (спілкування, дружба, любов),
* визнання (повага оточуючих, самооцінка),
* самовираження (вдосконалення, персональний розвиток).

На рисунку 1.1 представлено одну ієрархію потреби споживача, яку

хотілося б задовольнити, використовуючи майбутній програмний продукт.

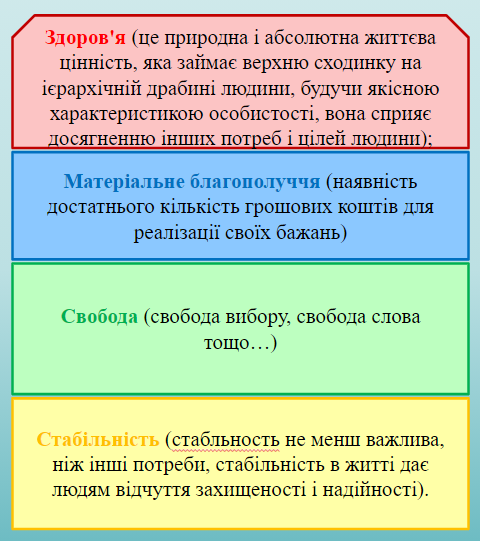


Рис. 1 – Ієрархія потреб споживача наш

1.1.2 Деталізація матеріальної потреби

Матеріальні потреби – це сукупність матеріальних благ, які мають матеріальну форму і виступають як головний спонукальний мотив трудової діяльності людини.

Споживач – це людина, яка споживає продукти виробництва і сфери послуг для відтворення робочої сили.

Тому була винайдена деталізація матеріальної потреби для нашого ПП.

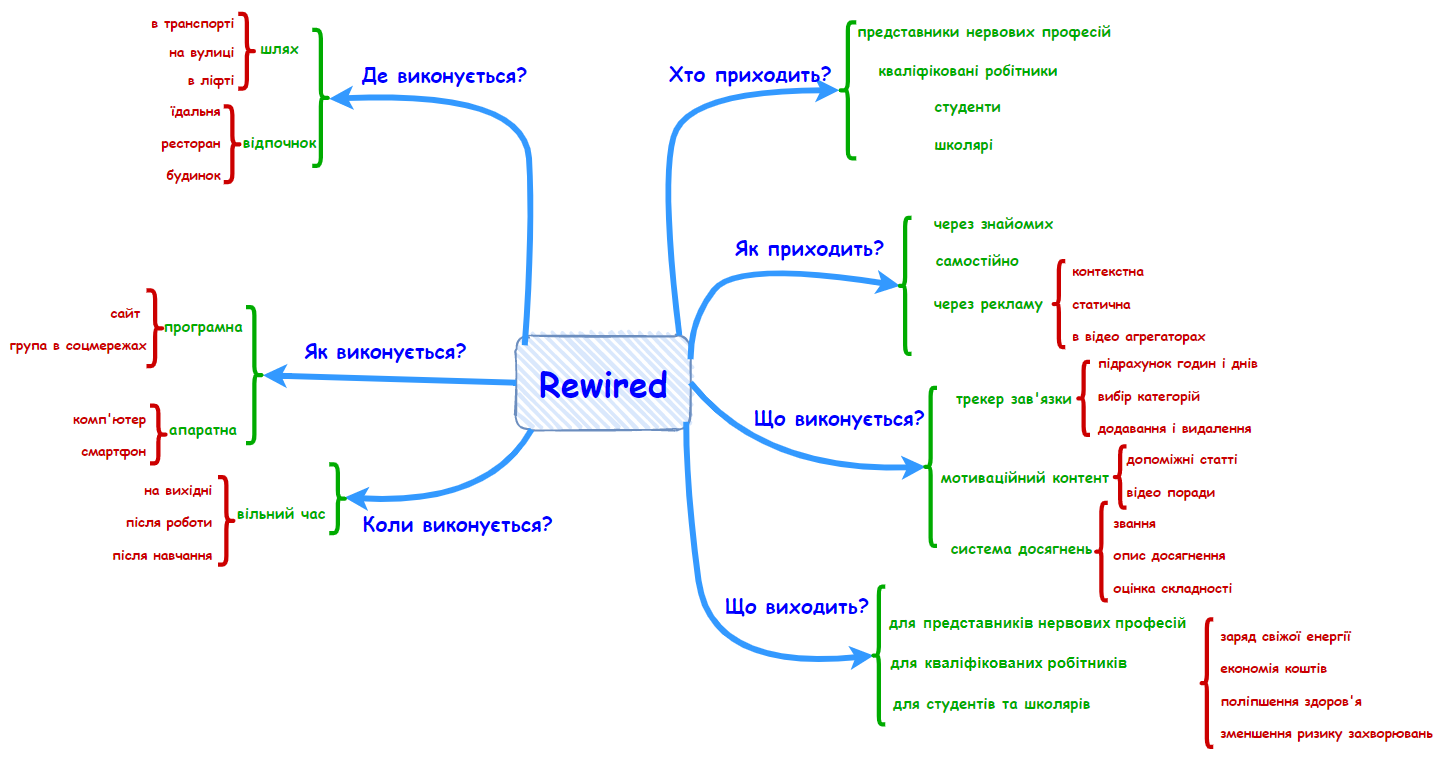


Рис. 2 - матеріальнi потреби майбутнього ПП

* 1. Бізнес вимоги до ПП

1.2.1 Опис проблем користувача

1.2.1.1 Концептуальний опис проблеми споживача

Умова задоволення потреб переведення даних в інформацію:

Доступність, Представленість мовою споживача, Інтерес, Актуальність.

Загальний опис проблеми: важко стежити за прогресом і контролювати себе під час позбуття від шкідливих звичок.

1.2.1.2 Метричний опис проблеми користувача

Метричні показники незадоволенності споживача: мала кількість корисних рішень для подолання залежності та високий рівень агітації шкідливих звичок в повсякденному житті.

Формула розрахунку доступності така:

Availability = (AST - DT) / AST × 100 = Servise or Component Availability (%) де

AST (agreed service time) - узгоджений час надання послуги;

DT (actual downtime during agreed service time) - фактичний час, коли послуга була недоступна протягом узгодженого часу її надання.

1.2.2 Мета створення ПП

1.2.2.1 Проблемний аналіз існуючих ПП

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Назва пробукту | Вартість | Ступінь готовності | Примітка |
| 1 | Sobriety Counter – Bad Habits | Безкоштвно | 1 | Недоступність на стаціонарних ПК |
| 2 | JustSayNo | Безкоштовно | 1 | Недоступна для більшості користувачів |
| 3 | HabitShare | Безкоштовно | 1 | Немає системи досягнень для користувачів |

Таблиця 1 – Аналіз існуючих програмних продуктiв

1.2.2.2 Мета створення ПП

Підвищення рівня доступності  програмного продукту для всіх соціальних рівнів. Допомога користувачам в налагодженні життєвого ритму, саморозвитку і особистісному зростанні, шляхом позбавлення від залежностей і шкідливих звичок.

1.2.3 Назва ПП

Rewired

1.2.3.1 Гасло ПП

Переналаштуй своє життя - Rewired

1.2.3.2 Логотип ПП



Рис. 3 – Логотип програмного продукту

* 1. Вимоги користувача до ПП
     1. Історія користувача ПП
* Як споживач, я маю можливість зберігати особисті дані;
* Як споживач, я маю можливість змінювати особисті дані;
* Як споживач, я маю можливість переглядати пункти меню і переміщатися між сторінками;
* Як споживач, я маю можливість створювати змінювати і видаляти звички;
* Як споживач, я маю можливість додавати таймер на кожну звичку;
* Як споживач, я маю можливість додавати опис негативної та позитивної сторони кожної звички;
* Як споживач, я маю можливість отримувати досягнення та отримувати і втрачати звання в залежності від часу утримання;
* Як споживач, я маю можливість переглядати мотиваційний контент;
* Як споживач, я маю можливість переглядати цитату дня;
* Як споживач, я маю можливість лайкати або ділитися цим контентом з друзями;
* Як споживач, я маю можливість вести свою статистику і додавати свої спостереження в замітки;
* Як споживач, я маю можливість переглядати контактну інформацію і зв'язуватися з підтримкою.

1.3.2 Діаграма прецедентів ПП

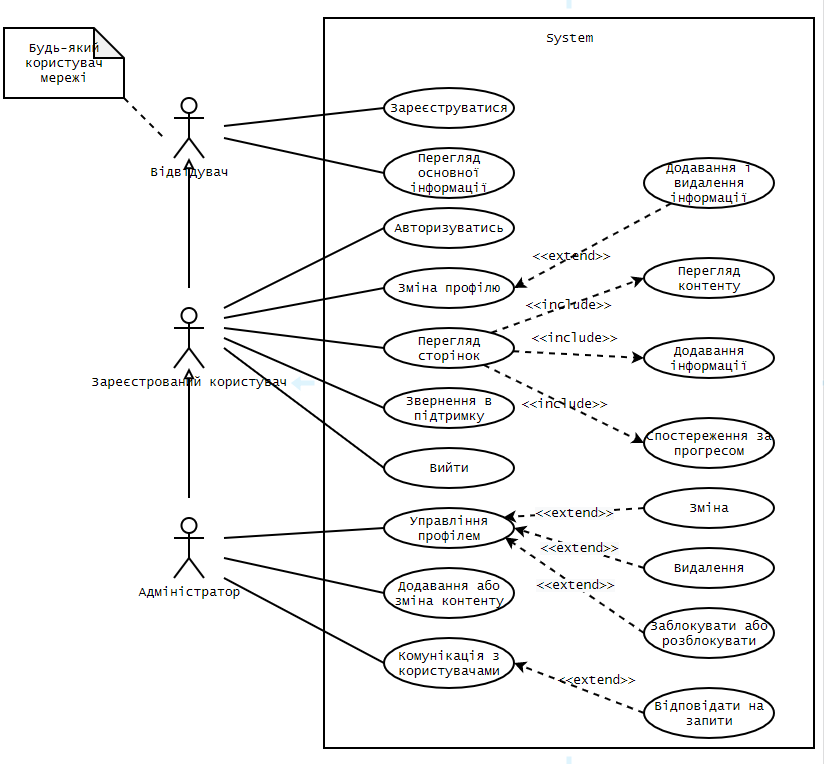


Рис. 4 – Діаграма прецедентів ПП

1.3.3 Сценарії використання прецедентів ПП

Прецедент «Зареєструвати користувача»:

Передумови початку: вiдвiдувача зацікавив ПП і він хоче створити обліковий запис

Актори: Відвідувач

Актор-основна зацікавлена особа: відвідувач

Гарантії успіху: відвідувач отримає власний обліковий запис та доступ до додаткової інформації.

Приклад основного успішного сценарію прецеденту «Зареєструвати користувача»:

* 1. ПП надає відвідувачеві можливість реєстрації
  2. Відвідувач переходить на сторінку реєстрації
  3. ПП відображає форму реєстрації у вигляды поля для введення ідентифікації та аутентифікації
  4. Відвідувач заповнює поля форми ідентифікації та аутентифікації і підтверджує реєстрацію
  5. ПП реєструє відвідувача.

Приклад альтернативного сценарію для попереднього

прикладу основного успішного сценарію прецеденту «Зареєструвати користувача»:

5.1 ПП виявляє, що відвідувач неправильно заповнив форму, видає повідомлення про помилку і переходить до кроку 1.

Прецедент «Перегляд основної інформації»:

Передумови початку: відвідувач тільки зайшов і поки нічого не знає про ПП, він хоче ознайомитися з ПП

Актори: відвідувач та зареєстрований користувач

Актор-основна зацікавлена особа: відвідувач

Гарантії успіху: відвідувач зацікавиться ПП і стане новим користувачем

Приклад основного успішного сценарію прецеденту «Перегляд основної інформації»:

1. ПП надає ресурс з основною ознайомчою інформацією

2. Відвідувач реєструється

Приклад альтернативного сценарію для попереднього прикладу основного успішного сценарію прецеденту «Перегляд основної інформації»:

3.1 Відвідувач покидає ресурс.

Прецедент «Зміна профілю»:

Передумови початку: користувач хоче змiнити інформацію в профілі (наприклад: нікнейм, ім'я користувача, ел. адресу)

Актори: зареєстрований користувач

Актор-основна зацікавлена особа: зареєстрований користувач

Гарантії успіху: користувач може розповісти про себе, заявити про себе широкому загалу

Приклад основного успішного сценарію прецеденту «Зміна профілю»:

* 1. ПП надає можливість користувачам редагувати профіль
  2. Користувач переходить на сторінку і змінює особисті дані
  3. ПП отримує запит на зміну даних
  4. ПП перевіряє і зберігає нові дані.

Приклад альтернативного сценарію для попереднього прикладу основного успішного сценарію прецеденту «Зміна профілю»:

4.1 ПП виявляє, що користувач передав їй неправильні значення, видає повідомлення про помилку і переходить до кроку 2.

Прецедент «Перегляд сторінок»:

Передумови початку: користувач хоче більше дізнатися про ПП

Актори: зареєстрований користувач

Актор-основна зацікавлена особа: зареєстрований користувач

Гарантії успіху: користувач дізнається про існування інших потенційно цікавих йому розділів.

Приклад основного успішного сценарію прецеденту «Перегляд сторінок»:

* 1. ПП надає можливість навігації між сторінками у вигляді пунктів меню
  2. Користувач переглядає пункти меню і переходить на вподобану сторінку

Приклад альтернативного сценарію для попереднього прикладу основного успішного сценарію прецеденту «Перегляд сторінок»:

2.1 Користувач переходить на іншу сторінку.

Прецедент «Звернення в підтримку»:

Передумови початку: у клієнта (користувача), виникла проблема або він хоче задати питання

Актори: зареєстрований користувач та адміністратор

Актор-основна зацікавлена особа: зареєстрований користувач

Гарантії успіху: користувач вирішить проблему, репутація ПП не постраждає.

Приклад основного успішного сценарію прецеденту «Звернення в підтримку»:

* 1. ПП надає можливість зв'язатися з підтримкою у вигляді поля для введення запиту
  2. Користувач заповнює поле і відправляє запитів в технічну підтримку
  3. ПП отримує запит від клієнта і фіксує його
  4. Адміністратор бачить запит від клієнта і зв'язується з ним

Приклад альтернативного сценарію для попереднього прикладу основного успішного сценарію прецеденту «Звернення в підтримку»:

3.1 Користувач неправильно заповнив поле або запит не пройшов

3.2 ПП повідомляє користувача про помилку.

Прецедент «Вийти»:

Передумови початку: користувач хоче вийти зі свого облікового запису

Актори: зареєстрований користувач

Актор-основна зацікавлена особа: зареєстрований користувач

Гарантії успіху: користувачів може зайти в інший обліковий запис або перезайти в свій.

Приклад основного успішного сценарію прецеденту «Вийти»:

* 1. ПП надає можливість виходу з облікового запису
  2. Користувач використовує цю можливість і виходить
  3. ПП завершує сесію користувача
  4. ПП видає користувачеві повідомлення з приводу успішного виходу і переводить його на головну сторінку.

Приклад альтернативного сценарію для попереднього прикладу основного успішного сценарію прецеденту «Вийти»:

3.1 ПП не вдається завершити сесію і він видає повідомлення про помилку при выході.

Прецедент «Управління профілем»:

Передумови початку: обліковий запис користувача неправильно заповнена або містить заборонений контент

Актори: адміністратор

Актор-основна зацікавлена особа: адміністратор

Гарантії успіху: успiшне редагування деталей облікового запису користувача

Приклад основного успішного сценарію прецеденту «Управління профілем»:

* 1. ПП надаэ можливість пошуку і редагування профілів користувачів адміністратору
  2. Адміністратор знаходить потрібного користувача
  3. Адміністратор змінити дані користувача і запитує у ПП дозвіл на збереження
  4. ПП перевіряє коректність змінюваних даних
  5. ПП зберігає нові дані і видає відповідне повідомлення.

Приклад альтернативного сценарію для попереднього прикладу основного успішного сценарію прецеденту «Управління профілем»:

4.1 ПП виявляє неправильні значення, видає повідомлення про помилку і переходить до кроку 1.

Прецедент «Додавання або зміна контенту»:

Передумови початку: матеріал розміщений на сторінках сайту втратив свою актуальність або потребує зміни

Актори: адміністратор

Актор-основна зацікавлена особа: адміністратор

Гарантії успіху: завжди якісна та актуальна інформація на сайті

Приклад основного успішного сценарію прецеденту «Додавання або зміна контенту»:

* 1. ПП надаэ можливість адміністратору редагувати контент
  2. Адміністратор обирає місце яке потребує редагування або поповнення інформацією
  3. Адміністратор запитує дозвіл на збереження змін
  4. ПП отримує запит на зміну даних
  5. ПП зберігає нові дані і видає відповідне повідомлення.

Приклад альтернативного сценарію для попереднього прикладу основного успішного сценарію прецеденту «Додавання або зміна контенту»:

4.1   ПП виявляє неправильно заповнені поля, видає повідомлення про помилку і переходить до кроку 1.

Прецедент «Комунікація з користувачами»:

Передумови початку: прийшов запит від користувача про допомогу

Актори: адміністратор, зареєстрований користувач

Актор-основна зацікавлена особа: адміністратор

Гарантії успіху: утримування потенційних клієнтів, підвищення рейтингу

Приклад основного успішного сценарію прецеденту «Комунікація з користувачами»:

* 1. ПП надає можливість зв'язку з користувачами
  2. ПП надсилає запит від користувача адміністратору
  3. Адміністратор бачить запит вiд клієнта та зв'язується з ним.
  4. Функціональні вимоги до ПП

1.4.1 Багаторівнева класифікація функціональних вимог

|  |  |
| --- | --- |
| Ідентифікатор  функції | Назва функції |
| FR1 | Реєстрація користувача |
| FR1.1 | ПП відображає форму реєстрації у вигляды поля для введення ідентифікації та аутентифікації |
| FR1.2 | Відвідувач заповнює поля форми ідентифікації та аутентифікації і підтверджує реєстрацію |
| FR1.3 | ПП реєструє користувача |
| FR1.4 | ПП виявляє, що відвідувач неправильно заповнив форму, видає повідомлення про помилку і переходить до кроку 1 |
| FR2 | Перегляд основної інформації |
| FR2.1 | ПП надає ресурс з основною ознайомчою інформацією |
| FR2.2 | Відвідувач реєструється |
| FR2.3 | Відвідувач покидає ресурс |
| FR3 | Зміна профілю |
| FR3.1 | ПП надає можливість користувачам редагувати профіль |
| FR3.2 | Користувач переходить на сторінку і змінює особисті дані |
| FR3.3 | ПП отримує запит на зміну даних |
| FR3.4 | ПП перевіряє і зберігає нові дані |
| FR3.5 | ПП виявляє, що користувач надав неправильні значення, видає повідомлення про помилку і переходить до кроку 2 |
| FR4. | Перегляд сторінок |
| FR4.1 | ПП надає можливість навігації між сторінками у вигляді пунктів меню |
| FR4.2 | Користувач переглядає пункти меню і переходить на сторінку |
| FR4.3 | Користувач переходить на іншу сторінку |
| FR5 | Звернення в підтримку |
| FR5.1 | ПП надає можливість зв'язатися з підтримкою у вигляді поля для введення запиту |
| FR5.2 | Користувач заповнює поле і відправляє запитів в технічну підтримку |
| FR5.3 | ПП отримує запит від клієнта і фіксує його |
| FR5.4 | Адміністратор бачить запит від клієнта і зв'язується з ним |
| FR5.5 | Користувач неправильно заповнив поле або запит не пройшов |
| FR5.6 | ПП повідомляє користувача про помилку |
| FR6 | Вийти |
| FR6.1 | ПП надає можливість виходу з облікового запису |
| FR6.2 | Користувач виходить |
| FR6.3 | ПП завершує сесію користувача і виводить відповідне повідомлення |
| FR6.4 | ПП не вдається завершити сесію і він видає повідомлення про помилку при выході |
| FR7 | Управління профілем |
| FR7.1 | ПП надаэ можливість пошуку і редагування профілів користувачів адміністратору |
| FR7.2 | Адміністратор знаходить потрібного користувача |
| FR7.3 | Адміністратор змінити дані користувача і запитує у ПП дозвіл на збереження |
| FR7.4 | ПП перевіряє коректність змінюваних даних |
| FR7.5 | ПП зберігає нові дані і видає відповідне повідомлення |
| FR7.6 | ПП виявляє неправильні значення, видає повідомлення про помилку і переходить до кроку 1 |
| FR8 | Додавання або зміна контенту |
| FR8.1 | ПП надаэ можливість адміністратору редагувати контент |
| FR8.2 | Адміністратор обирає місце яке потребує редагування або поповнення інформацією |
| FR8.3 | Адміністратор запитує дозвіл на збереження змін |
| FR8.4 | ПП отримує запит на зміну даних |
| FR8.5 | ПП зберігає нові дані і видає відповідне повідомлення |
| FR8.6 | ПП виявляє неправильно заповнені поля, видає повідомлення про помилку і переходить до кроку 1 |
| FR9 | Комунікація з користувачами |
| FR9.1 | ПП надає можливість зв'язку з користувачами |
| FR9.2 | ПП надсилає запит від користувача адміністратору |
| FR9.3 | Адміністратор бачить запит вiд клієнта та зв'язується з ним |

Таблиця 2 – Функціональні вимоги до ПП

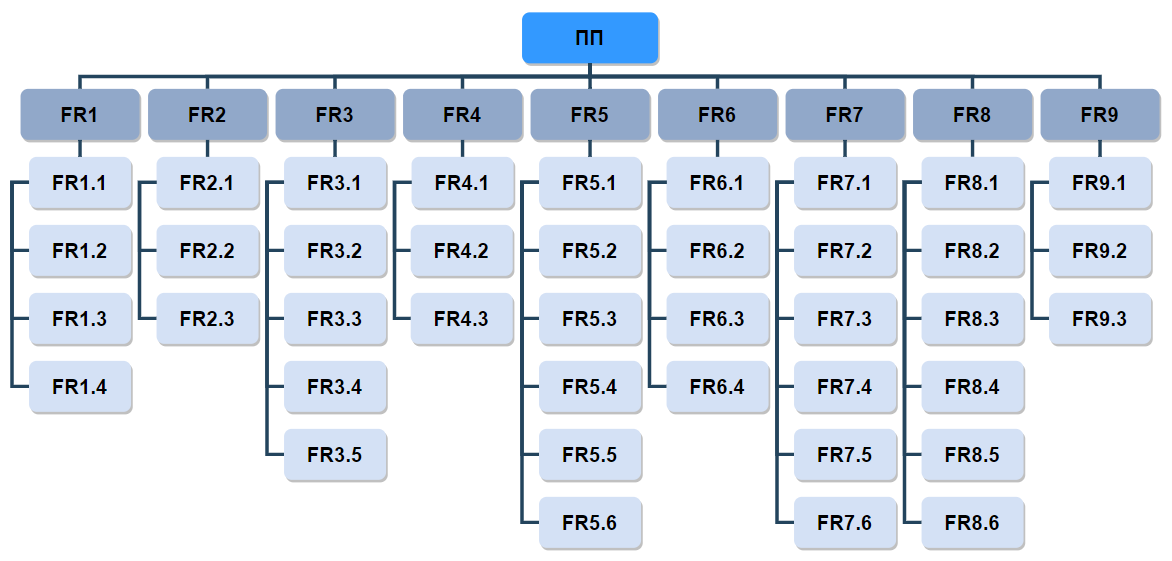


Рис. 5 – WBS структура вимог до ПП

1.4.2 Функціональний аналіз існуючих ПП

|  |  |  |  |
| --- | --- | --- | --- |
| Ідентифікатор  функції | Sobriety Counter –  Bad Habits | JustSayNo | HabitShare |
| FR1 | - | + | + |
| FR2 | + | + | + |
| FR3 | + | + | + |
| FR4 | - | - | - |
| FR5 | + | + | + |
| FR6 | - | + | + |
| FR7 | - | + | + |
| FR8 | - | - | - |
| FR9 | + | + | + |

Таблиця 3 – Функціональний аналіз існуючих ПП

* 1. Нефункціональні вимоги до ПП

1.5.1 Опис зовнішніх інтерфейсів

1.5.1.1 Опис інтерфейса користувача

1.5.1.1.1 Опис INPUT-інтерфейса користувача

|  |  |  |
| --- | --- | --- |
| Ідентифікатор  функції | Засіб INPUT-потоку | Особливості використання |
| FR1.2 | - 2/3-кнопочний маніпулятор  типу "миша“;  - сенсорний екран (Touchscreen, Touchpad, Multi-touch); | Використання миші і кнопки миші або сенсорного екрану для переходу на сторінку |
| FR1.3 | - 2/3-кнопочний маніпулятор  типу "миша“;  - стандартна комп`ютерна клавіатура;  - сенсорний екран (Touchscreen, Touchpad, Multi-touch); | Використання миши або сенсорного екрану для переміщення і підтвердження, а клавіатуру для введення даних |
| FR1.4 |
| FR2.1 |
| FR2.2 |
| FR2.3 |
| FR3.2 | - 2/3-кнопочний маніпулятор  типу "миша“;  - стандартна комп`ютерна клавіатура;  - сенсорний екран (Touchscreen, Touchpad, Multi-touch); | Використання миші і сенсорного екрану для переміщення, а клавіатури для введення змінюваних даних |
| FR4.1 | Використання миші або сенсорного екрану для переміщення по сторінці і між сторінками |
| FR4.2 |
| FR4.3 |
| FR5.2 | Використання миші і сенсорного екрану для переміщення, а клавіатури для введення повiдомлення |
| FR5.4 |
| FR6.2 | - 2/3-кнопочний маніпулятор  типу "миша“;  - сенсорний екран (Touchscreen, Touchpad, Multi-touch); | Використання миші або сенсорного екрану для навігації |
| FR6.3 |
| FR6.4 |
| FR7.2 | - 2/3-кнопочний маніпулятор  типу "миша“;  - стандартна комп`ютерна клавіатура;  - сенсорний екран (Touchscreen, Touchpad, Multi-touch); | Використання миші і сенсорного екрану для переміщення, а клавіатури для введення змінюваних даних |
| FR7.3 |
| FR7.5 | - 2/3-кнопочний маніпулятор  типу "миша“;  - сенсорний екран (Touchscreen, Touchpad, Multi-touch); | Використання миші або сенсорного екрану для навігації |
| FR7.6 |
| FR8.2 | - 2/3-кнопочний маніпулятор  типу "миша“;  - стандартна комп`ютерна клавіатура;  - сенсорний екран (Touchscreen, Touchpad, Multi-touch); | Використання миші і сенсорного екрану для переміщення, а клавіатури для введення змінюваних даних |
| FR8.3 |
| FR8.5 | - 2/3-кнопочний маніпулятор  типу "миша";  - сенсорний екран (Touchscreen, Touchpad, Multi-touch); | Використання миші або сенсорного екрану для навігації |
| FR8.6 |
| FR9.2 |
| FR9.3 |

Таблиця 4 – INPUT-інтерфейси

1.5.1.1.2 Опис OUTPUT-потоків

|  |  |
| --- | --- |
| Ідентифікатор  функції | Засіб OUTPUT -  потоку |
| FR1 | https://lh6.googleusercontent.com/x1Z1omu_t4Dgc4QVmxzcPyUvVofFyIZXJfAdwL59dV2jHAObH-Dk1uc_QYmPOe-2CDhoS6lhUq3ySxJeEJOfe4wOW7DvPBMzsgYht6jao8ZgXy9uzWpl5vkNShNBlvGtxZReHKg |
| FR2 | https://lh6.googleusercontent.com/ecOS85G1vdUx7kMCgb0sv8zgxydfQDhniE-G2cIKyt7dcAC7bWtu4zc5RmLO-Gsp1_D-yIDfjHigK3k8pyCvt6AngHJUc0QZx1zD0DiPFvtwciW7_NnUjy4HVOOHP1swpF9_y2k |
| FR3 | https://lh3.googleusercontent.com/85-m0YEJ133802RwpLZuDlOV-EVXCI5zCRDJkP9R8b8ui-sBlA5XpsufMn9RVSHfOUTTn3k4GL025KNK9v6iahlRR_HOir-rKU7MK1Sq82dxHPsCD_KGTQug-GWwIch1F-LUbAc |
| FR4 | https://lh3.googleusercontent.com/QEVs2mcypD71QJguGAPgKThfB86yLbbi_83aJXm4-enaqSfWfJS856pVUTVTxBJxKyIQWivh1znQAZ0A8QLl3QDjpg6QYFvczfNqTv5fOL1r97yH0BHYelXHFPU2EeqH6imSoZo |
| FR6 | https://lh4.googleusercontent.com/ovF3o9LjLSX8vUxXk6VbBgN6ltWVLIMobYOSk0wv8pKqj4AH_D7guUZdoc2hU8lgSCupugG3kJorqePNPg6u2yOXzY0EtCHtvekR_ZZJpVux8W34wo86nYb5QwToV7_ITnz0fx8 |

Таблиця 5 – Засоби OUTPUT-потоків

1.5.1.2 Опис інтерфейсу iз зовнішніми пристроями

|  |  |
| --- | --- |
| Ідентифікатор функції | Зовнішній пристрій |
| FR1 | - Desktop-персональний комп`ютер;  - Notebook;  - смартфон;  - мобільний телефон; |
| FR2 |
| FR3 |
| FR4 |
| FR5 |
| FR6 |
| FR7 |
| FR8 |
| FR9 |

Таблиця 6 – Опис інтерфейсу з зовнішніми пристроями

1.5.1.3 Опис програмних інтерфейсів

Версії операційних систем та програмних бібліотек, які знадобляться при реалізації більшості функцій ПП.

|  |  |
| --- | --- |
| Версії операційних систем та програмних бібліотек, які знадобляться при реалізації більшості функцій ПП. | * Windows * Linux * PHP * CMS |

Таблиця 7 – Опис програмних інтерфейсів

1.5.1.4 Опис інтерфейсів передачі інформації

Iнтерфейси передачі інформації, які знадобляться при

реалізації більшості функцій ПП:

* провідні інтерфейси:

- Ethernet

* безпровідні інтерфейси:

- Wi-Fi

1.5.1.5 Опис атрибутів продуктивності

|  |  |
| --- | --- |
| Ідентифікатор  функції | Максимальний час реакції ПП на дії  користувачів, секунди |
| FR1 | 5 |
| FR2 | 2 |
| FR3 | 5 |
| FR4 | 2 |
| FR5 | 1 |
| FR6 | 3 |
| FR7 | 5 |
| FR8 | 2 |
| FR9 | 2 |

Таблиця 8 – Опис атрибутів продуктивності

2 ПЛАНУВАННЯ ПРОЦЕСУ РОЗРОБКИ ПРОГРАМНОГО ПРОДУКТУ

2.1 Планування ітерацій розробки програмного продукту

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ідентифікатор  функції | Назва функції | Функціональні  залежності | Вплив на  досягнення  мети, % | Пріоритет  функції |
| FR1 | Реєстрацiя користувача | - | 10% | M (Must) |
| FR2 | Перегляд основної інформації | - | 20% | S (Should) |
| FR3 | Зміна профілю | FR1 | 5% | S (Should) |
| FR4 | Перегляд сторінок | FR2 | 15% | M (Must) |
| FR5 | Звернення в підтримку | - | 5% | C (Could) |
| FR6 | Вийти | FR1 | 5% | C (Could) |
| FR7 | Управління профілем | FR1 | 10% | W (Want) |
| FR8 | Додавання або зміна контенту | - | 20% | M (Must) |
| FR9 | Комунікація з користувачами | - | 10% | C (Could) |

Таблиця 9 – Планування ітерацій розробки

2.2 Концептуальний опис архітектури програмного продукту

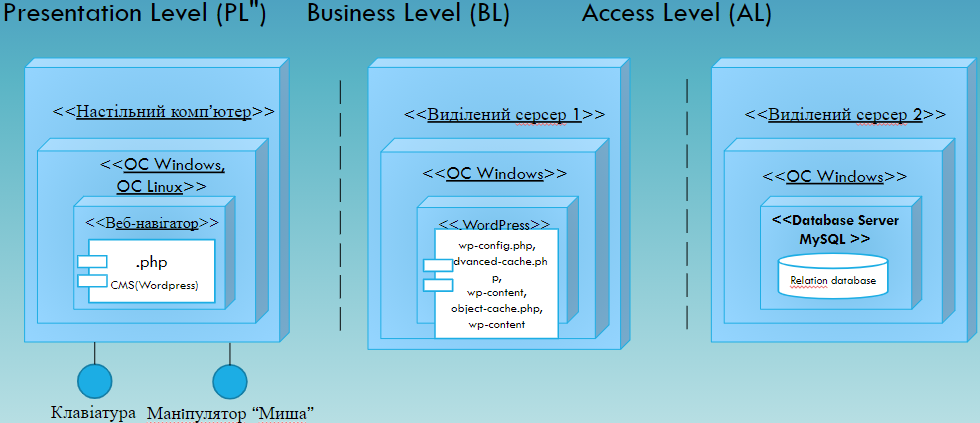


Рис. 6 – Концептуальний опис архітектури ПП

* 1. План розробки ПП

2.3.1 Оцінка трудомісткості розробки ПП

1. Визначення вагових показників акторів

|  |  |  |
| --- | --- | --- |
| Назва актора | Тип актора | Ваговий коефіцієнт |
| Вiдвiдувач | Простий | 1 |
| Зареєстрований користувач | Складний | 3 |
| Адміністратор | Середній | 2 |

Таблиця 10 – Вагові коефіцієнти акторів

А = 1 + 3 + 2 = 6

1. Визначення вагових показників прецедентів UC

|  |  |  |  |
| --- | --- | --- | --- |
| Назва прецедента | Тип  прецедента | Кількість кроків  сценарію | Ваговий  коефіцієнт |
| FR1 | Складний | 4-7 | 10 |
| FR2 | Середній | <= 3 | 5 |
| FR3 | Середній | 4-7 | 10 |
| FR4 | Середній | <= 3 | 5 |
| FR5 | Середній | 4-7 | 10 |
| FR6 | Середній | 4-7 | 10 |
| FR7 | Складний | 4-7 | 10 |
| FR8 | Складний | 4-7 | 10 |
| FR9 | Простий | <= 3 | 5 |
| Визначення UUCP:  A = 1 + 3 + 2 = 6; UC = 10 + 5 +10 + 5 + 10 + 10 + 10 + 10 + 5 = 75;  UUCP = A + UC = 6 + 75 = 81 | | | |

Таблиця 11 – Вагові показники прецедентів

1. Визначення технічної складності проекту

|  |  |  |
| --- | --- | --- |
| Показник | Опис показника | Вага |
| T1 | Распределенная система | 1 |
| T2 | Высокая производительность (пропускная способность) | 3 |
| T3 | Работа конечных пользователей в режиме онлайн | 2 |
| T4 | Сложная обработка данных | 1 |
| T5 | Повторное использование кода | 0,5 |
| T6 | Простота установки | 0 |
| T7 | Простота использования | 3 |
| T8 | Переносимость | 2 |
| T9 | Простота внесения изменений | 2 |
| T10 | Параллелизм | 1 |
| T11 | Специальные требования к безопасности | 3 |
| T12 | Непосредственный доступ к системе со стороны внешних пользователей | 1 |
| T13 | Специальные требования к обучению пользователей | 0,5 |
| Визначення TCF: TCF = 0,6 + (0,01 \* (STi \* Вага\_i)) = 0,6 + (0,01 \* (37)) = 0,97 | | |

Таблиця 12 – Визначення технічної складності проекту

1. Визначення рівня кваліфікації розробників

|  |  |  |
| --- | --- | --- |
| Показник | Опис показника | Вага |
| F1 | Знакомство с технологией | 1,5 |
| F2 | Опыт разработки приложений | 1 |
| F3 | Опыт использования объектно-ориентированного подхода | 2 |
| F4 | Наличие ведущего аналитика | 3 |
| F5 | Мотивация | 4 |
| F6 | Стабильность требований | 1 |
| F7 | Частичная занятость | 2 |
| F8 | Сложные языки программирования | 1 |
| Визначення EF:  EF = 1,4 + (- 0,03 \* (SFi \* Вагаi)) = 1,4 + (- 0,03 \* (21)) = 0,77 | | |

Таблиця 13 – Визначення рівня кваліфікації розробників

1. Визначення UCP

UCP = UUCP \* TCF \* EF

UUCP = 81

TCF = 0,97

EF = 0,77

UCP = 81 \* 0,97 \* 0,77 = 60,4989

1. Оцінка трудомісткості проекту

Трудомісткість = UCP \* 20 = 1 209,978 люд\год

2.3.2 Визначення дерева робіт з розробки ПП

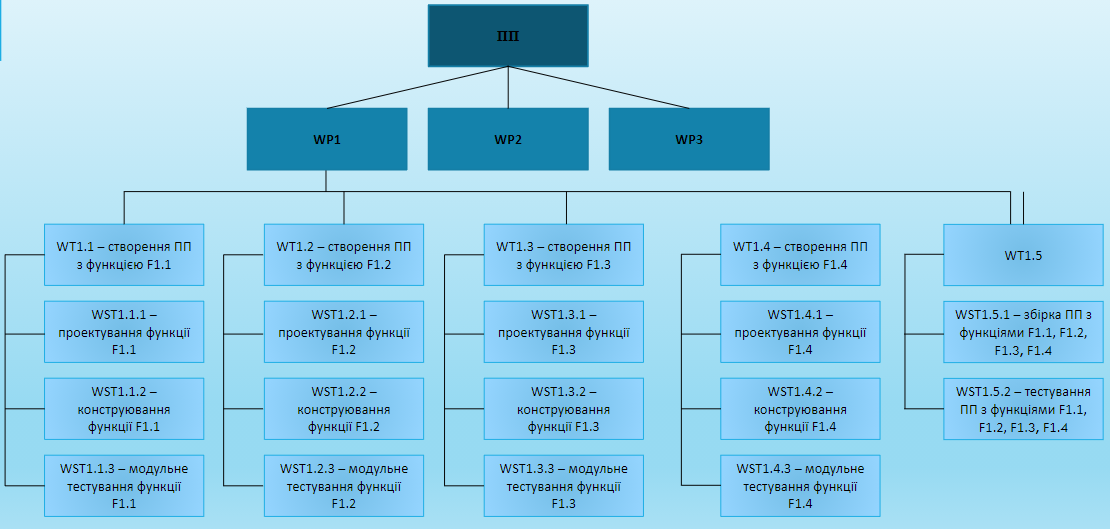


Рис. 7 – WBS дерево робіт

|  |  |
| --- | --- |
| Iдентифiкатор | Хто виконував |
| 1.1\* | Мудрик В. С. |
| 1.2\* | Мудрик В. С. |
| 1.3\* | Глаголев Р. Ю. |
| 1.4\* | Глаголев Р. Ю. |
| 1.5\* | Глаголев Р. Ю. |

Таблиця 14 –Опис пiдзадач iз закрiпленням виконавцiв

* + 1. Графік робіт з розробки ПП

2.3.3.1 Таблиця з графіком робіт



Рис. 8 – Графік робіт

2.3.3.2 Діаграма Ганта

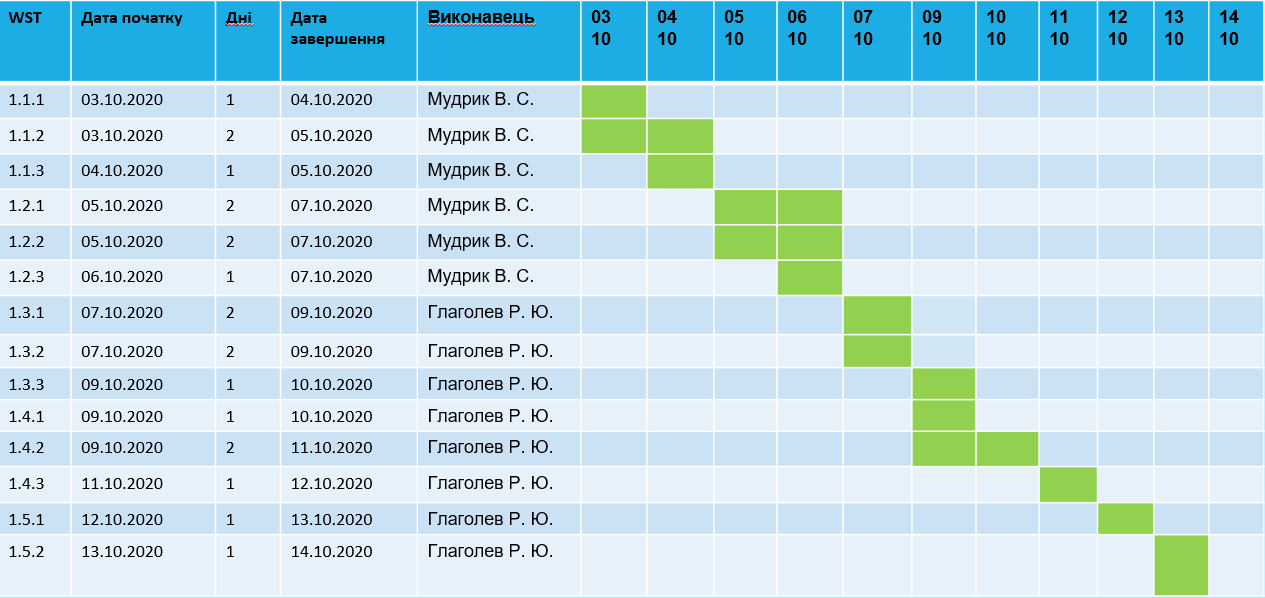


Рис. 9 – Діаграма Ганта

3 ПРОЕКТУВАННЯ ПП

3.1 Концептуальне та логічне проектування структур даних ПП

3.1.1 Концептуальне проектування на основі UML-діаграми концептуальних класів

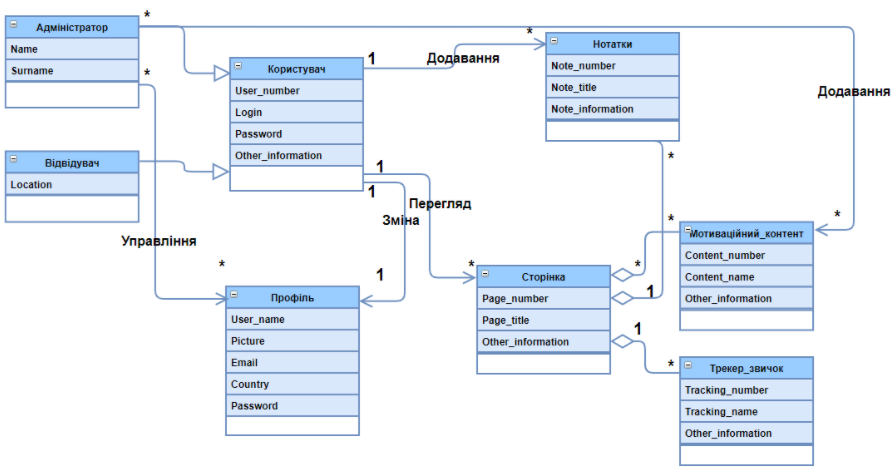


Рис. 10 – UML-діаграма концептуальних класів

* + 1. Логічне проектування структур даних

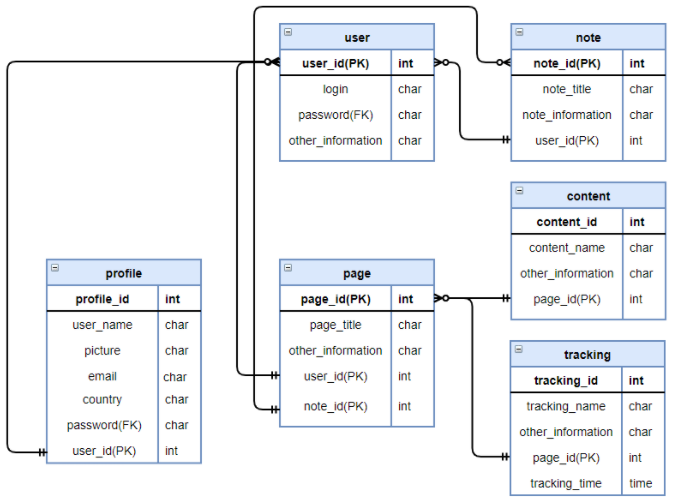


Рис. 11 – Структура БД

* 1. Проектування програмних класів

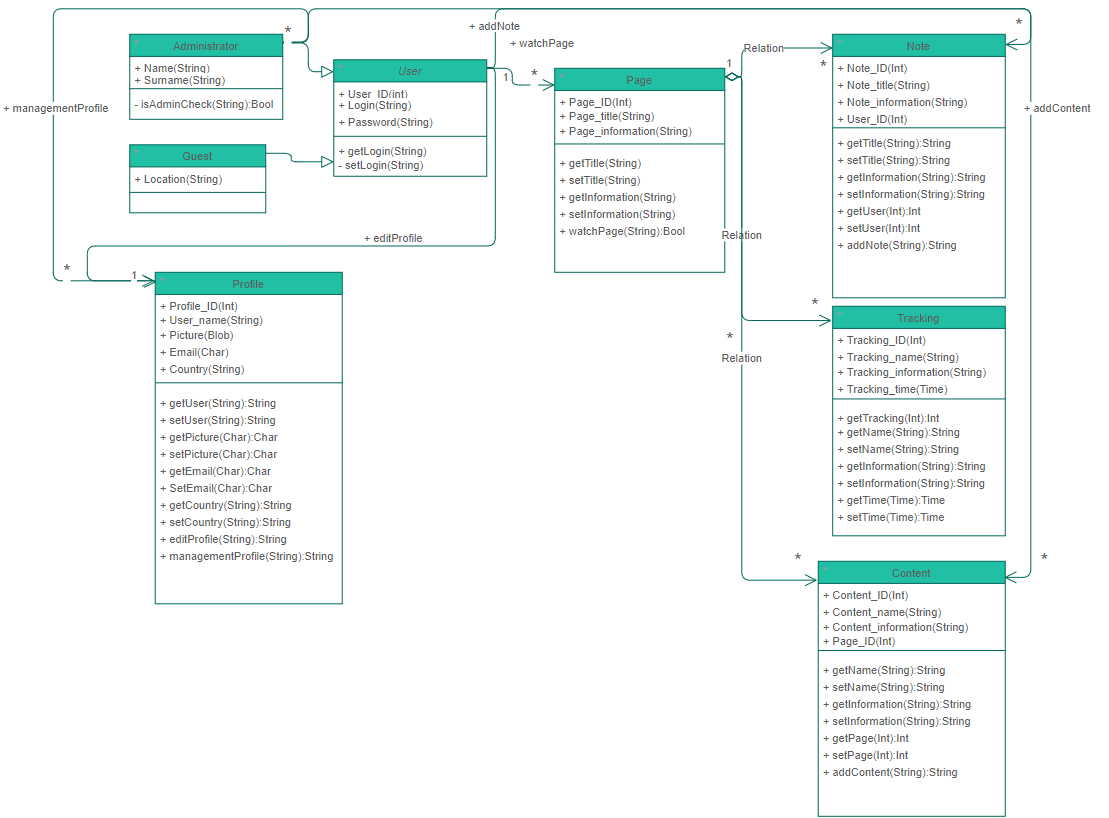


Рис. 12 – Проектування програмних класів

* 1. Проектування алгоритмів роботи методів програмних класів

@startuml  
title signup()  
start  
    :Вивiд на екран відвідувача полів даних для реєстрації;  
    if (Дані введені не в усі обов'язкові поля або вказані невірно) then (Так);  
        :Вивiд помилки реєстрації;  
        stop  
    else (Нi);  
        if ( Немає підключення до бази даних ) then (Так);  
            :Вивiд помилки реєстрації;  
            stop  
        else (Нi);  
            :Введення в базу даних нового облікового запису користувача;  
            :Висновок користувачеві повідомлення про вдалу реєстрацію;  
            :Перенаправлення користувача на головну сторінку;  
 stop  
@enduml

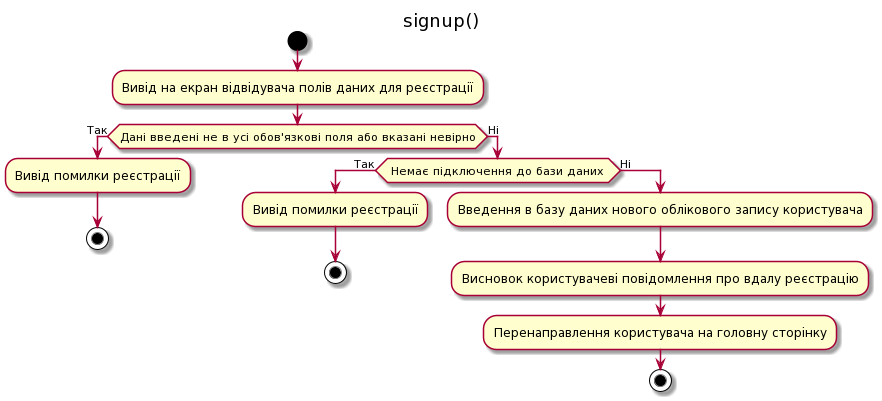


Рис. 13 – UML дiаграма для Signup

@startuml  
title editingprofile()  
start  
    :Завантаження шаблону сторінки редагування профілю;  
    if (Дані змінено невірно або введені не в усі поля) then (Так);  
        :Вивiд помилки редагування;  
        stop  
    else (Нi);  
        if ( Немає підключення до бази даних ) then (Так);  
            :Вивiд повідомлення про помилку;  
            stop  
        else (Нi);  
            :Введення в базу даних нових параметрів облікового запису користувача;  
            :Вивiд користувачеві повідомлення про вдалу зміну даних;  
            :Перенаправлення користувача на попередню сторінку;  
 stop  
@enduml



Рис. 14 – UML дiаграма для Editingprofile

@startuml  
title login()  
start  
    :Введення даних для ідентифікації і аутентифікації користувача;  
    :Отримання даних з глобального масиву;  
    if(Якщо немає підключення до бази даних) then (Так);  
        :Висновок повідомлення про помилки;  
        stop  
    else (Нi);  
        if ( У таблиці користувачі не існує запису з заданими параметрами ) then (Так);  
            :Вивiд повідомлення про неіснуючого користувача;  
            stop  
        else (Нi);  
            :Створення в глобальному масиві поля залогіневшегося користувача і присвоювання йому значення які були отримані з глобального масиву;  
            :Перенаправлення користувача на головну сторінку сайту;  
stop  
@enduml



Рис. 15 – UML дiаграма для Login

@startuml  
title contentAdd()  
start  
repeat  
    :Завантаження шаблону сторінки для додавання нових записів;  
    :Додавання заголовка і основного змісту запису;  
    if( Якщо немає з'єднання з базою даних) then (Так);  
        :Вивiд повідомлення про помилку;  
        stop    
    else (Нi);  
        :Аналіз існуючих записів на сторінці;  
        :Перевірка на існування подібного запису;  
        if(Така запicь вже була створена раніше) then (Так);  
            :Вивiд повідомлення про помилку;  
        else (Нi);  
            :Збереження в базу даних;  
            :Інформація про успішне додавання запису;  
        stop  
@enduml

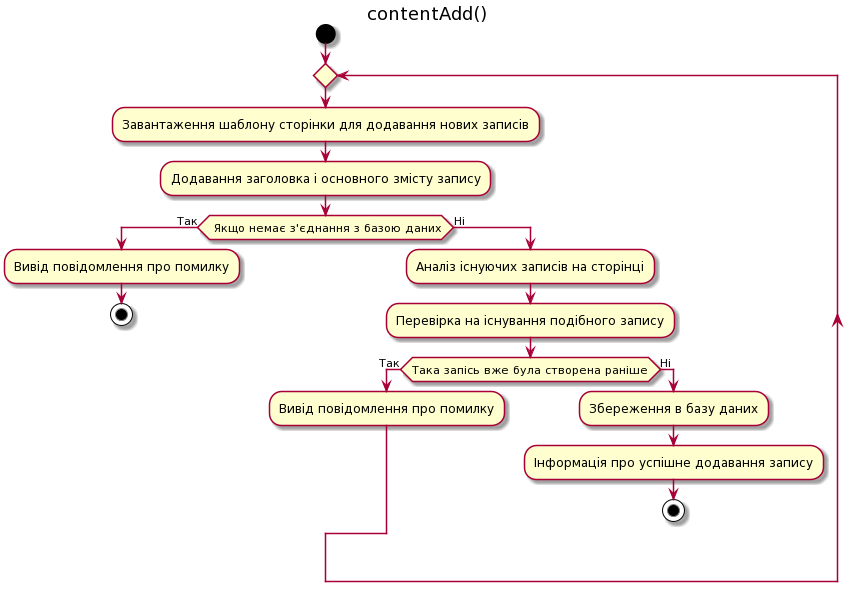


Рис. 16 – UML дiаграма для Contentadd

|  |  |  |  |
| --- | --- | --- | --- |
| № | Назва методу | Глаголев Р.Ю. | Мудрик В.С. |
| 1 | Signup.php | + |  |
| 2 | Showinfo.php |  | + |
| 3 | Editingprofile.php | + |  |
| 4 | Support.php |  | + |
| 5 | Login.php | + |  |
| 6 | Logout.php |  | + |
| 7 | Contentadd.php | + |  |
| 8 | Search.php |  | + |

Таблиця -15 Таблиця участників

* 1. Проектування тестових наборів методів програмних класів

|  |  |  |  |
| --- | --- | --- | --- |
| Назва функції | Номер тесту | Опис значень вхідних даних | Опис очікуваних значень результату |
| Signup() | 1 | user\_name = user01  password =qwertyu123 | Користувач успішно зареєстрований |
| Signup() | 2 | user\_name = 01user  password =qwertyu123 | Помилка: рядок user\_name повинна починатися з латинської літери |
| Signup() | 3 | user\_name = 01uuussseeerrr01  password =qwertyu123 | Помилка: перевищення розміру логіна користувача |
| Signup() | 4 | user\_name = user01  password =1234 | Помилка: довжина рядка password повинна бути не менше 8 символів |
| Login() | 1 | user\_name = “правильне значення”  password = “правильне значення” | Користувач успішно увійшов в акаунт |
| Login() | 2 | user\_name = “правильне значення логiна”  password = “правильне значення пароля” | Помилка: логін або пароль введено невірно |
| Login() | 3 | user\_name = “неправильне значення логiна”  password = “неправильне значення пароля” | Помилка: логін або пароль введено невірно |
| Contentadd() | 1 | title = “Чи не усувайте погані звички - поміняйте їх”  text = “вводимо будь-який текст” | Запис успішно додано |
| Contentadd() | 2 | title = “вводимо неприпустимі символи”  text = “вводимо будь-який текст” | Помилка: будь ласка, введіть коректну назву статті |
| Contentadd() | 3 | title = “вводимо більше 50 символів”  text = “вводимо будь-який текст” | Помилка: назва занадто довга, будь ласка скоротіть його |
| Contentadd() | 4 | title = “залишаємо поле порожнім”  text = “залишаємо поле порожнім” | Помилка: запис не може бути порожнім, будь ласка внесіть дані |
| SetCountry() | 1 | “Вибираємо будь-яку країну їхнього списку” | Країна успішно змінена |
| SetNickname() | 2 | nickname = firstuser01 | Нові дані успішно збережені |
| SetNickname() | 3 | nickname = “вводимо неприпустимі символи” | Помилка: нiкнейм містить неприпустимі символи, будь ласка спробуйте ще раз |
| SetNickname() | 4 | nickname = “вводимо більше 15 символів” | Помилка: нiкнейм занадто довгий |
| SetNickname() | 5 | “Залишаємо поле порожнім і зберігаємо” | Помилка: поле нiкнейму не може бути порожнім, будь ласка внесіть дані |
| SetEmail() | 6 | email = namesurname@gmail.com | Нова електронна пошта успішно збережена |
| SetEmail() | 7 | “Залишаємо поле порожнім і зберігаємо” | Помилка: поле не може бути порожнім, будь ласка вкажіть пошту |

Таблиця 16– Тестові набори

|  |  |  |  |
| --- | --- | --- | --- |
| № | Назва методу | Глаголев Р.Ю. | Мудрик В.С. |
| 1 | Signup.php | + |  |
| 2 | Showinfo.php |  | + |
| 3 | Editingprofile.php | + |  |
| 4 | Support.php |  | + |
| 5 | Login.php | + |  |
| 6 | Logout.php |  | + |
| 7 | Contentadd.php | + |  |
| 8 | Search.php |  | + |

Таблиця 17 - Таблиця учасників

4 КОНСТРУЮВАННЯ ПП

4.1 Особливості конструювання структур даних

4.1.1 Особливості інсталяції та роботи з СУБД

Будь-якому починаючому розробнику WordPress необхідно ознайомитися з двома речами: з технічною документацією WordPress і з особливостями архітектури бази даних, як мінімум, для того щоб вміти створювати складні WordPress шаблони і простенькі плагіни, сьогодні ми познайомимося з архітектурою бази даних сайту на WordPress.

Перше, що слід сказати: як системи управління базами даних WordPress використовує MySQL сервер версії 5.0.15 і вище. Як вже говорилося: кінцевому користувачеві інформація про БД WordPress навряд чи колись стане в нагоді, багатьом розробникам для створення тем та плагінів WordPress буде досить набору функцій WordPress для роботи з базами даних, але іноді бувають ситуації, коли API WordPress недостатньо.

Тобто, іноді виникає потреба звертатися до баз даних безпосередньо. Тому потрібна інформація про те, як WordPress зберігає свої дані і які є залежності і обмеження між таблицями бази даних WordPress. Щоб звернутися до баз даних WordPress безпосередньо, слід використовувати WPDB клас

4.1.2 Особливості створення структур даних

Функція dbDelta ()

Вбудована функція dbDelta () дозволяє створювати таблиці в базі даних WordPress, і вносити зміни в їх структуру. Але перед створенням нової таблиці, необхідно визначитися з її найменуванням і кодуванням.

Назва нової таблиці повинно мати той же префікс, який використовується ядром WordPress (за замовчуванням wp\_) і додатковий префікс для нашого плагіна або проекту, наприклад: wp\_my\_table\_name. Префікс wp\_ можна отримати за допомогою методу get\_blog\_prefix () глобального об'єкта $ wpdb, а за допомогою властивостей charset і collate можна визначити використовувану кодування:

global $wpdb;

$table\_name = $wpdb->get\_blog\_prefix() . 'my\_products';

$charset\_collate = "DEFAULT CHARACTER SET {$wpdb->charset} COLLATE {$wpdb->collate}";

Далі за допомогою функції dbDelta () ми можемо створити нову таблицю:

require\_once( ABSPATH . 'wp-admin/includes/upgrade.php' );

$sql = "CREATE TABLE {$table\_name} (

id int(11) unsigned NOT NULL auto\_increment,

name varchar(255) NOT NULL default '',

price int(11) unsigned NOT NULL default '0',

PRIMARY KEY (id),

KEY price (price)

) {$charset\_collate};";

// Создать таблицу.

dbDelta( $sql );

Функція dbDelta () відрізняється від простого MySQL запиту в базу даних WordPress. Вона розбиває структуру таблиці на частини і порівнює її з тією таблицею, яка вже існує. Це дозволяє dbDelta () вносити зміни в структуру таблиці, без необхідності видаляти і знову створювати таблицю.

Наприклад, щоб додати нову колонку в нашу таблицю, ми можемо змінити наш основний запит CREATE TABLE наступним чином:

$sql = "CREATE TABLE {$table\_name} (

id int(11) unsigned NOT NULL auto\_increment,

name varchar(255) NOT NULL default '',

price int(11) unsigned NOT NULL default '0',

color varchar(255) NOT NULL default '',

PRIMARY KEY (id),

KEY price (price)

) {$charset\_collate};";

4.2 Особливості конструювання програмних модулів

4.2.1 Особливості роботи з інтегрованим середовищем розробки

При верстці сайту можна використовувати різні редактори коду або системи керування вмістом. Чому ми вибрали саме WordPress?

Основні переваги це:

* легкойсть поновлення
* професійні шаблони
* неймовірно потужний движок
* повний контроль і право власності

Але існують і недоліки, це:

* процес навчання
* технічне обслуговування

Що краще? WordPress або HTML?

Якщо в майбутньому оновлення сайту не настільки важливо, тоді в будь-якому випадку вибирати HTML сайт буде правильно. Так ви швидше за все досягнете своєї мети.

Якщо ж вам необхідно вільніше, контролювати свій сайт і не витрачати даремно гроші щомісяця наймаючи розробників для додавання зображень, то WordPress очевидно в перевазі.

Використовуючи WordPress, ви можна легко додавати контент на сайт. легко созадть будь-яку кількість сторінок, скільки забажаєте, в будь-який момент. Ви можете додати блог, щоб ваші клієнти дізнавалися про оновлення, або зробити власну розсилку, щоб залучити більше клієнтів. Замість того, щоб платити розробникам багато грошей за виконання простих завдань, ви можете робити це самостійно з такою ж швидкістю.

4.2.2 Особливості створення програмної структури з урахуванням спеціалізованого фреймворку

Це просунутий спосіб, він більш складний, але разом зі складністю він відкриває широкі можливості. За допомогою цього способу можна задати шаблон будь-якій сторінці, записи, категорії, будь публікації на сайті або взагалі групі будь-яких публікацій.

// фільтр передає змінну $ template - шлях до файлу шаблону.

// Змінюючи цей шлях ми змінюємо файл шаблону.

add\_filter( 'template\_include', 'my\_template' );

function my\_template( $template ) {

# Аналог другого способу

// якщо це сторінка з складаючи portfolio, використовуємо файл шаблону page-portfolio.php

// використовуємо умовний тег is\_page ()

if( is\_page('portfolio') ){

if ( $new\_template = locate\_template( array( 'page-portfolio.php' ) ) )

return $new\_template ;

}

# Шаблон для групи рубрик

// цей приклад буде використовувати файл з папки теми tpl\_special-cats.php,

// як шаблон для рубрик з ID 9, назвою "Без рубрики" і складаючи "php"

if( is\_category( array( 9, 'Без рубрики', 'php') ) ){

return get\_stylesheet\_directory() . '/tpl\_special-cats.php';

}

# Шаблон для запису по ID

// файл шаблону розташований в папці плагіна /my-plugin/site-template.php

global $post;

if( $post->ID == 12 ){

return wp\_normalize\_path( WP\_PLUGIN\_DIR ) . '/my-plugin/site-template.php';

}

# Шаблон для сторінок довільного типу "book"

// передбачається, що файл шаблону book-tpl.php лежить в папці теми

global $post;

if( $post->post\_type == 'book' ){

return get\_stylesheet\_directory() . '/book-tpl.php';

}

return $template;

}

4.2.3 Особливості створення програмних модулів

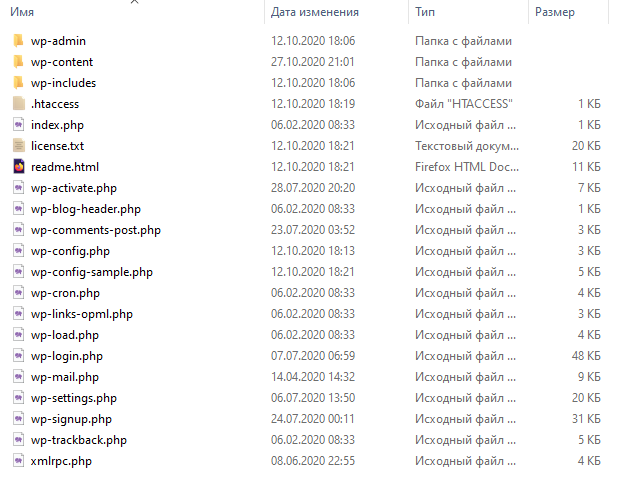


Рис. 17 - Файлова структура

При розробці сайту було використано **Open Server Panel.** Це портативна серверна платформа і програмне середовище, створена спеціально для веб-розробників з урахуванням їх рекомендацій і побажань.

Всі програмні мудулі знаходяться на локально диску і в разі необхідності запусткаются з відкритим портом в глобальну мережу Інтерет.

4.2.4 Особливості розробки алгоритмів методів програмних класів або процедур/функцій

Класичний клас

Це найбільш частий і зрозумілий метод. Використовується метод \_\_construct () для створення екземпляра класу. Але він же найбільш проблемний через те що багато програмісти можуть його неправильно готувати.

Приклад:

class MyPlugin1

{

public function \_\_construct()

{

add\_action('plugins\_loaded', array($this, 'loaded'));

}

public function loaded()

{

// do stuff

}

}

Cтатичний клас

Цей підхід використовую найчастіше. За його гнучкість і можливість перемикання в інші стилі і підходи. Часто такі механіки можна зустріти в WooCommerce.

Приклад:

class My\_Plugin2

{

public static function init ()

{

add\_action ('plugins\_loaded', array (\_\_ CLASS\_\_, 'loaded'));

}

public static function loaded ()

{

// do stuff

}

}

My\_Plugin2 :: init ();

Сінглтон

Часто застосовуються там де потрібно гарантовано мати один екземпляр об'єкта в системі без дублювання.

Плюс більш передбачувана і зрозуміла робота зі станом об'єкта ніж у Статичної підходу.

Бував два типу старту: через init () або instance ()

Приклад:

class MyPlugin3

{

private static $ ins;

public static function init ()

{

add\_action ('plugins\_loaded', array (\_\_ CLASS\_\_, 'instance'));

}

public static function instance ()

{

is\_null (self :: $ ins) && self :: $ ins = new self;

return self :: $ ins;

}

private function \_\_construct ()

{

add\_action ('plugins\_loaded', array ($ this, 'loaded'), 20);

}

public function loaded ()

{

// do stuff

}

}

4.3 Тестування програмних модулів

Сценарій тестування функції реєстрації:

Передумови початку: користувач хоче зареєструватися

Сценарій:

* відвідувач переходить на сторінку реєстрації;
* потім вводить логін, нікнейм, країну і пошту;
* після чого вводить два рази пароль;
* потім користувач натискає кнопку реєстрації;
* підтверджуї свої дії;
* останній зареєстрований учасник переходити на сторінку входу в аккаунт;
* вводить логін;
* вводить пароль;
* потім користувач натискає на кнопку входу;
* потім користувач може перейти в особистий кабінет і при бажанні може змінити свої дані.

Сценарій тестування функції редагування профілю:

Передумови початку: користувач хоче змiнити особистi данi

Сценарій:

* користувач переходить в особистий кабінет
* заходить в розділ детальної інформації
* змінити логін він не може
* він змінює нікнейм, при бажанні
* вказує іншу країну при бажанні
* пошту змінити він теж не може
* потім подтвержадет збереження
* проходить тест на бота
* оновлює сторінку

5 РОЗГОРТАННЯ ТА ВАЛІДАЦІЯ ПП

5.1 Інструкція з встановлення ПП

Основні кроки установки програмного продукту:

1. Клонувати репозиторій з сайтом
2. Зберегти всі дані зі сховищ в корінну доменну папку веб сервера
3. Запустити веб сервер
4. На веб сервері створити базу даних з вiдповiдною назвою
5. Перейти на розташування веб серверу (домене ім’я:порт)

5.2 Інструкція з використання ПП

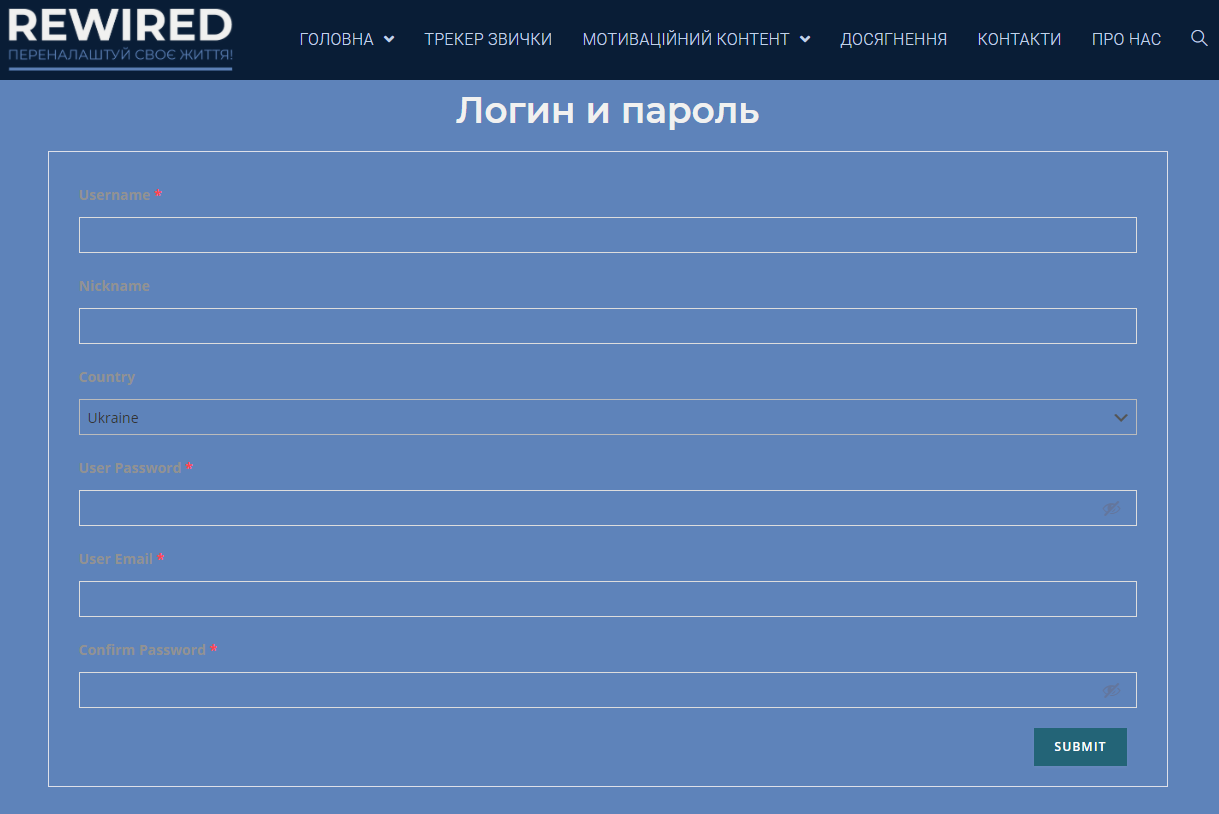


Рис. 22 – Регестрація

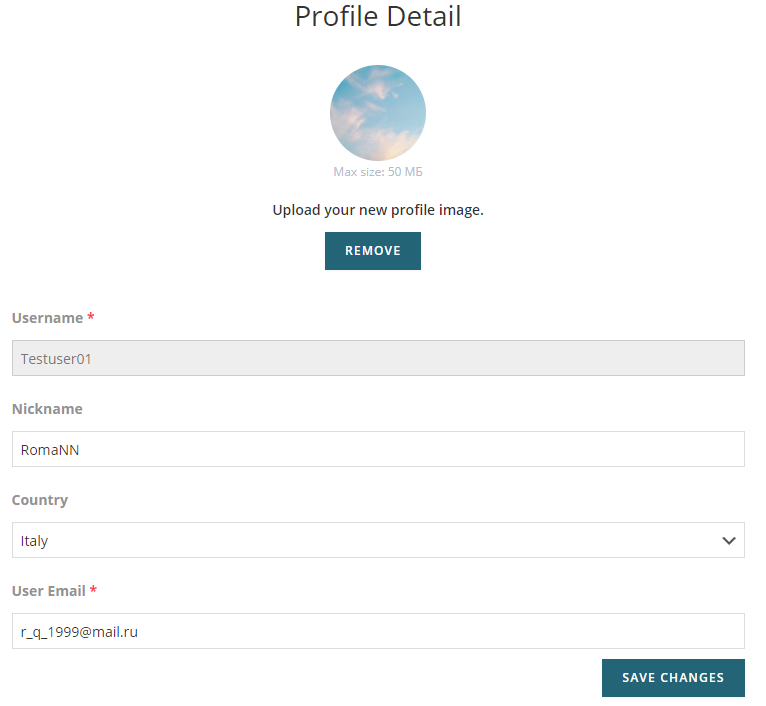


Рис. 18 – Редагування профілю як результат успішної реєстрації



Рис. 19 – Перегляд статей на сайті

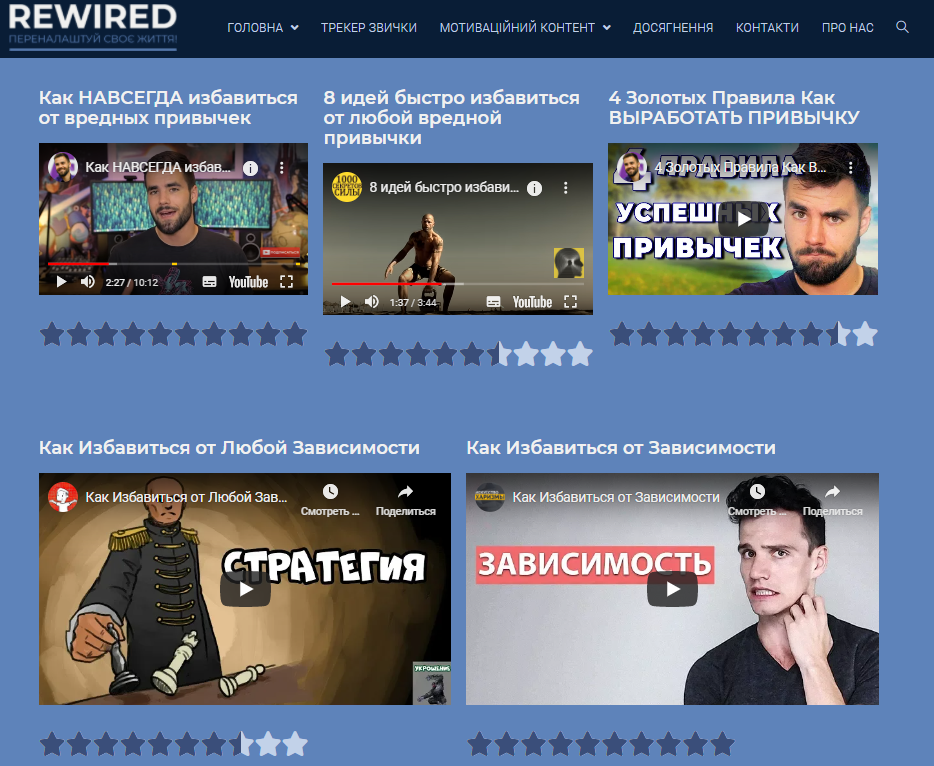


Рис. 20 – Перегляд роликів на сайті

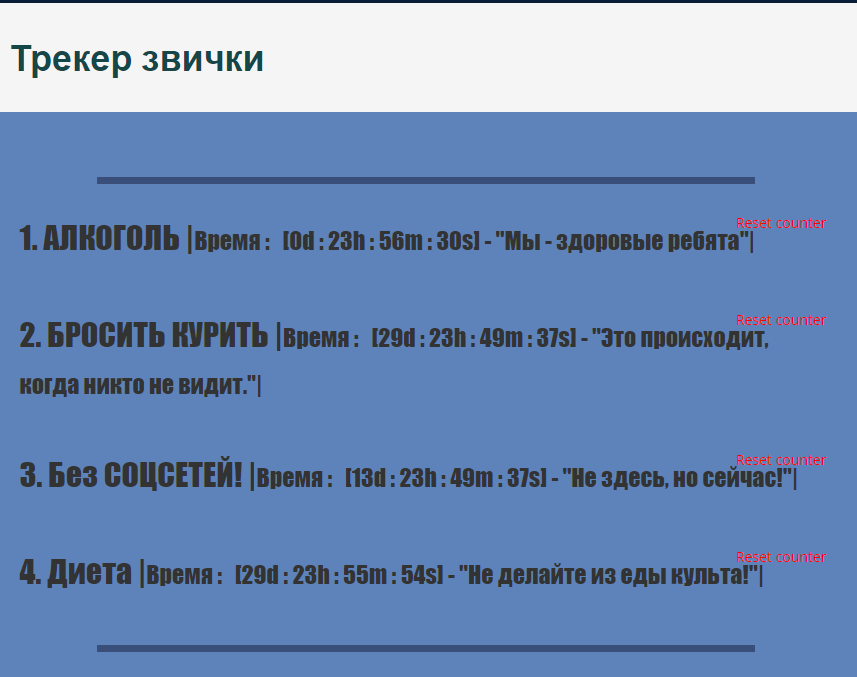


Рис. 21 Робота таймера звички

5.3 Результати валідації ПП

Мета створення ПП була підвищення рівня доступності  програмного продукту для всіх соціальних рівнів и допомога користувачам в налагодженні життєвого ритму, саморозвитку і особистісному зростанні.

Формула розрахунку доступності така:

Availability = (AST - DT) / AST × 100 = Servise or Component Availability (%)

де AST (agreed service time) - узгоджений час надання послуги;

DT (actual downtime during agreed service time) - фактичний час, коли послуга була недоступна протягом узгодженого часу її надання.

Поки що можна лише припустити, що ПП пройшов валідацію і буде виконувати свої функції, так як програмний продукт вийшов непоганий, але поки не вийшов на ринок.

ВИСНОВКИ

Як результат, було отримано непоганий програмний продукт, який допоможе людям подолати свої залежностi і страхи, не на самоті, а разом.

Переналаштуй своє життя - Rewired

В процесі створення програмного продукту виникли такі труднощі:

1) обмеженість в часі;

2) складність розробки ПП;

3) недостатні знання де-яких інструментів.

ДЖЕРЕЛА

1. https://wpmag.ru/2014/custom-tables-dbdelta/
2. https://wp-kama.ru/id\_5177/3-sposoba-sozdat-shablon-stranitsyi.html
3. https://wpcraft.ru/blog/3-podhoda-k-programmirovaniyu-klassov-v-wordpress/