|  |
| --- |
| Національний лісотехнічний університет України |
| (повне найменування вищого навчального закладу) |
|  |
| Навччально-науковий інститут деревообробних та |
| комп’ютерних технологій і дизайну |
| (повне найменування інституту, назва факультету (відділення)) |
|  |
| Кафедра інформаційних технологій |
| (повна назва кафедри (предметної, циклової комісії)) |
|  |

|  |
| --- |
| **Пояснювальна записка** |
| до дипломної роботи |
| ОКР – бакалавр |
| (освітньо-кваліфікаційний рівень) |
|  |

|  |  |
| --- | --- |
| на тему: | Розробка веб-орієнтованої інформаційної системи "Збирання Instagram |
|  | аудиторій для реклами у Facebook" на основі MVC CodeIgniter |
|  |  |

|  |  |
| --- | --- |
| Виконав: | студент **ІV** курсу, групи **КН - 41** |
|  | спеціальності |
|  | 122 – “Комп’ютерні науки” |
|  | (шифр і назва напряму підготовки, спеціальності) |
|  | Манжула Д. В. |
|  | (прізвище та ініціали) |
| Керівник | Бекас Б. О. |
|  | (прізвище та ініціали) |
| Рецензент |  |
|  | (прізвище та ініціали) |

Львів – 2020 року

Національний лісотехнічний університет України

( повне найменування вищого навчального закладу )

Інститут ННІ деревообробних та комп’ютерних технологій і дизайну

Кафедра інформаційних технологій

Освітньо-кваліфікаційний рівень бакалавр

Спеціальність 122 “Комп’ютерні науки”

(шифр і назва)

**ЗАТВЕРДЖУЮ**

**Завідувач кафедри**

\_\_\_\_\_\_\_\_\_\_\_\_ Соколовський Я.І.

“\_\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_2020 року

**ЗАВДАННЯ**

**НА ДИПЛОМНИЙ РОБОТУ СТУДЕНТУ**

Манжулі Давиду Вадимовичу

(прізвище, ім’я, по батькові)

1. Тема роботи Розробка веб-орієнтованої інформаційної системи "Збирання

Instagram аудиторій для реклами у Facebook" на основі MVC CodeIgniter

керівник роботи Бекас Богдан Олексійович

( прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від “ 16 ” грудня 2019 року

№ С-662

2. Строк подання студентом проекту (роботи) 09 червня 2020 року

3. Вихідні дані до роботи Постановка задачі та її формалізація

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

4.1. Стан проблемної області

4.2. Інформаційне та математичне забезпечення

4.3. Алгоритм вирішення завдання та його опис. Програмна реалізація та

налагодження програми.

4.4. Програмне та технічне забезпечення

4.5. Висновки. Список використаної літератури. Додатки

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень) Підготовка матеріалу для доповіді.

6. Дата видачі завдання “ 17 ” грудня 2019 року

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Назва етапів дипломної роботи | Строк виконання етапів роботи | Примітка |
| 1 | Огляд літературних та інших джерел згідно досліджуваної теми. | 15.02.2020 р. | *Виконано* |
| 2 | Системний аналіз об’єкту дослідження та предметної області. | 29.02.2020 р. | *Виконано* |
| 3 | Постановка задачі та її формалізація. | 13.03.2020 р. | *Виконано* |
| 4 | Вибір та обґрунтування методів і засобів розв’язання завдання. | 27.03.2020 р. | *Виконано* |
| 5 | Реалізація завдання та його опис | 10.04.2020 р. | *Виконано* |
| 6 | Програмна реалізація. | 24.04.2020 р | *Виконано* |
| 7 | Відлагодження програми. | 8.05.2020 р. | *Виконано* |
| 8 | Аналіз результатів розв’язування контрольного прикладу | 22.05.2020 р. | *Виконано* |
| 9 | Здати пояснювальну записку на перевірку та рецензування. | 09.06.2020 р. | *Виконано* |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Студент |  |  | | | | ***Манжула Д. В.*** |
|  | (підпис) |  | | ( прізвище та ініціали ) | | |
| Керівник роботи |  |  |  | | ***Бекас Б. О.*** | |
|  | (підпис) |  | | ( прізвище та ініціали ) | | |

# РЕФРЕРАТ

Дипломна робота присвячена розробці веб-застосунку за допомогою технологій PHP CodeIgniter та Twitter Bootstrap, з можливістю інтеграції з службами аналітики Google Analytics та Facebook Pixel. Обґрунтовано актуальність вибраного проєкту. Створено веб-додаток. Реалізовано функціонал реєстрації та авторизації. Інтерфейс сервісу є красивим, сучасним, і водночас простим.

Дипломна робота містить 65 сторінок пояснювальної записки, 26 рисунків, 3 додатків, 22 джерел.

Ключові слова: Instagram, просування, мультипосилання, візитка, аналітика, CodeIgniter, MVC, PHP.

# ABSTRACT

The thesis is devoted to the development of a web application using PHP CodeIgniter and Twitter Bootstrap technologies, with the possibility of integration with Google Analytics and Facebook Pixel. The relevance of the selected project is substantiated. Web application created. Implemented registration and authorization functionality. The interface of the service is beautiful, modern, and at the same time simple.

Thesis contains 65 pages of explanatory note, 26 figures, 3 appendices, 22 sources.

Keywords: Instagram, promotion, multi-link, business card, analytics, CodeIgniter, MVC, PHP.

# ТЕХНІЧНЕ ЗАВДАННЯ

Розробити веб-застосунок для створення та керування особистої онлайн-сторінки візитки.

Необхідно реалізувати наступний функціонал:

* Реєстрація, авторизація, відновлення паролю;
* Додавання, редагування, видалення посилань;
* Налаштування зовнішнього вигляду сторінки;
* Статистика переглядів та переходів по посиланнях;
* Інтеграція з службами аналітики Google Analytics та Facebook Pixel;
* Можливість інтеграції з сайтами за допомогою плаваючого віджета;
* Форма для збору повідомлень, наприклад, номерів телефону або адрес електронної пошти;
* Локалізації;
* Адмін-панель для зручного перегляду списку користувачів та інформації про них;
* Платний функціонал, інтеграція з платіжними системами для оплати платної підписки.

Сайт повинен бути оптимізованим для пошукових систем, мати інтуїтивно простий інтерфейс, не тормозити, простим в управлінні контентом.

[**ЗМІСТ**](#_v7hfl1wbpeg7)7

[**ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ**](#_1ty1riy80a8e)8

[**ВСТУП**](#_c8ietwt4rzfe)9

[**1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ**](#_qepox2k5rbmx)10

[1.1 Огляд проблемної області](#_y9ixjxxvfpcx) 10

[1.2 Аналітика](#_2c6tk112do46) 10

[1.3 Аналоги Choko.Link](#_334n3rro9eli) 13

[TapLink](#_nmhu5fov09fu) 13

[Mssg.me](#_1868ohlv11ff) 14

[tapid.ru](#_ikfz9qejiwj2) 14

[BMBE.ru](#_rpvq50ebvpnk) 15

[LinkTree](#_fuvd6p1jzb1) 15

[**2. ІНФОРМАЦІЙНЕ ТА МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ**](#_evpbzfvdkmo)17

[2.1 Веб-застосунок](#_hsmgufhxgb74) 17

[2.2 LAMP-стек](#_463nsx1pnrbf) 19

[2.3 MVC](#_zhshattta63i) 22

[2.4 CodeIgniter](#_dsfm04d2mgyv) 23

[2.5 Bootstrap](#_jghq5yxd867u) 27

[**3. ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ**](#_ihi8gbxx8zzt)30

[3.1 GIT](#_bzpvloclslll) 30

[3.2 Середовище розробки](#_4denrm4mu7ur) 31

[3.3 Суть проєкта](#_i3egkfue66ie) 32

[3.4 Структура маршрутів](#_p5bkcyw7db2q) 36

[3.5 Структура проєкту](#_5oq7i7mcktiw) 37

[3.6 Структура бази даних](#_hxtvdqb8yrqj) 40

[3.7 Влаштування сторінки візитки](#_rydnpf8p27fl) 42

[**ВИСНОВКИ**](#_p35qpqa7xqbx)46

[**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**](#_sj5rcrqeno3x)47

[**ДОДАТКИ**](#_btjai0c03uy3)48

[Account\_controller.php](#_3nkjgxf9brk4) 48

[Profile\_controller.php](#_p7w1qq2wl15w) 60

[Form\_model.php](#_ow9brhu8gkxl) 65

# ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

**Аналітика** – онлайн-сервіс, що дозволяє збирати аналітичні дані про відвідувачів веб-сайту. Також аналітикою називають отримані в результаті дані.

**Сервер** – комп’ютерна програма, що запущена на серверному апаратному забезпеченні і призначена для обробки HTTP-запитів.

**CodeIgniter** – один з безлічі фреймворків, призначених для розробки веб-застосунків, написаних на PHP/

**SQL** (Structured query language) – Мова структурованих запитів для взаємодії користувача з базами даних.

**CSS** (CascadingStyleSheets) – Каскадні таблиці стилів – Мова для опису зовнішнього вигляду сторінок, написаних мовами розмітки.

**Twitter Bootstrap** ‑ CSS-фреймворк для швидкого створення сучасної, кросбраузерної і стандартизованої верстки.

**MVC** (Model-View-Controller) – Модель-Вигляд-Контролер – це архітектурний шаблон для розробки програмного забезпечення.

**HTML** (HypertextMarkupLanguage) ‑ Мова розмітки для web-сторінок та web-додатків.

**PHP** (PersonalHomePageTool) – Мова програмування яка генерує HTML-сторінки на сторінці web-сервера.

# ВСТУП

**Актуальність**. Останнім часом все більше і більше населення користується соціальними мережами. Однією з таких є Інстаграм. Інстаграм це дуже потужна платформа, але в ній є свої обмеження. Одним з них обмеження на посилання. Можна вказати тільки **одне** посилання в шапці профілю. Посилання в інших місцях будуть не клікабельними, а ще їх не можна навіть скопіювати. Щоб вийти з ситуації, можна створити сторінку-візитку, і вже на ній розмістити стільки посилань, скільки душа забажає.

**Об'єкт дослідження** - розробка веб-орієнтованої системи на базі MVC CodeIgniter.

**Предмет дослідження** - розробка сервісу для створення сторінок-візиток з функціоналом збору Instagram-аудиторій для подальшої реклами у Facebook.

**Мета роботи** - розробити сервіс, що полегшить процес створення сторінки візитки для простих користувачів та дасть можливість обійти обмеження платформи Instagram.

**Завдання**, які потрібно вирішити:

* Реєстрація користувачів;
* Додавання, редагування та видалення посилань;
* Налаштування зовнішнього вигляду сторінки візитки;
* Інтеграція з службами аналітики Google Analytics та Facebook Pixel;
* Інтеграція віджету для веб-сайту;
* Збір контактних даних за допомогою форми зворотнього зв’язку;

**Практичне значення** полягає в вирішенні однієї з безлічі проблем користувача, таким чином зробивши світ трішечки кращим.

# 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

## 1.1 Огляд проблемної області

В соцмережах зареєстроване практично все населення країни, що здатне тримати в руках смартфон чи натискати клавіші стаціонарного комп’ютера. Однією з найпопулярніших серед молоді являється Instagram, і не дарма. Його простий функціонал припав до душі багатьом користувачам, а особливо молоді. Ця соцмережа дає змогу заявити про себе як експерта і просувати свій особистий бренд. Попри багатий функціонал і безліч переваг, в інстаграм є і обмеження. Одним з таких обмеження на посилання. Можна вказати тільки **одне** посилання в шапці профілю. Посилання в пості, коментарях чи інших місцях будуть не клікабельними, а також їх не можна скопіювати.

Перевірене часом рішення - створити сторінку-візитку, і вже на ній розмістити стільки посилань, скільки душа забажає. Можна, звісно, зразу розмістити посилання на сайт, але, на відміну від сайту, який є великим і тяжким, сторінка-візитка легка і мінімалістична - тільки ім’я, опис і посилання. Та й сайт іноді буває зайвим - користувачу треба розмістити тільки свої контакти на месенджери, тримати цілий сайт буде накладно в плані як часових, так і фінансових ресурсів: оплата за домен, хостинг, а здебільшого і розробку й підтримку сайту буде непосильним тягарем для блогера-початківця, студента-фрілансера чи просто школяра-підлітка, що вирішив спробувати щось нове.

Посилання на сторінку-візитку можна розмістити не тільки в Instagram-профілі, а й інших соцмережах або надрукувати на паперовій візитці. В таких сервісів здебільшого короткий і естетично красивий домен, а адресу своєї сторінки можна обрати самому.

## 1.2 Аналітика

Свого клієнта треба знати в лице. Необхідно розуміти, з ким доведеться працювати та кому продавати. Щоб не тратити час та бюджет на просування даремно, маркетингові активи мусять бути направлені на цільову аудиторію.

Цільова аудиторія - це всі потенційні клієнти, що заінтересовані в товарі чи послузі. Реальні і потенційні клієнти схожі між собою за різними ознаками: географічне розташування, вік, стать, сімейний стан, фінансовий стан тощо.

Раніше, щоб проаналізувати ці дані, потрібна була ціла команда аналітиків, вони проводили опитування, шукали закономірності, робили різні дослідження. Завдяки сучасним технологіям, все це відійшло в минуле.

Якщо в проєкта вже є деяка клієнтська база, можна зібрати дані про цільову аудиторію за допомогою сервісів аналітики. Найпопулярніші серед них - Google Аналітика, Facebook Pixel, Яндекс.Метрика. Google Аналітика більше підійде, якщо планується реклама в Google Adwords, Яндекс.Метрика - в аналогічному сервісі російської корпорації “Яндекс”, а Facebook Pixel - для просування в соцмережах Facebook та Instagram.

В цих службах аналітики можна переглянути дані аудиторій по багатьох критеріях - віком, географією, статтю, мовою та іншими. Також тут можна формувати аудиторії користувачів, що виконали конверсію - певну цільову дію - для подальшого аналізу чи показу їм певної реклами. Наприклад, можна показувати рекламу всім користувачам, що колись додали товар в корзину але не здійснили покупку. Саме тому іноді ми “магічним” чином бачимо всюди в інтернеті рекламу саме того смартфона, характеристики якого на днях шукали.

Крім збирання аудиторій, служби аналітики також відстежують статистику конверсій і допомагають проаналізувати, які заходи збільшують конверсії, а які не впливають чи зменшують її. Наприклад, можна перевірити, як впливає реклама в YouTube на продажі через сайт, чи впливає блок з аксесуарами на середній чек, яка доля користувачів авторизована на сайті. Ці всі знання допоможуть менше витрачати фінанси на непотрібні речі і направити їх на потрібні і, таким чином, більше заробляти.

Щоб сервіс аналітики збирав аудиторії і аналітику, в першу чергу необхідно встановити код служби аналітики собі на сайт. Він відносно малий. Він лише ініціалізує конфігурацію і ідентифікатор кампанії, а потім завантажує основний код, який вже відстежує кліки, конверсії і все що треба.

Всі служби аналітики для своєї роботи зберігають ідентифікатор користувача в Cookies, щоб мати до нього доступ незалежно на якому сайті знаходиться скрипт аналітики. Якщо куку встановив скрипт з домену google-analytics.com, то навіть якщо цей скрипт підключити до свого сайту, скрипт матиме доступ до своїх кук. Тож код аналітики завжди знатиме що за користувач переглядає сторінку.

До того ж, Cookies служб аналітики неможливо позбутись. Це не тому що в них налаштовані якісь права доступу, що користувач не може їх видалити. Все набагато простіше: сьогодні код служб аналітики встановлений практично на всіх сайтах і, видаливши куки, через мить, перейшовши на будь-який сайт, вони будуть створені знову і продовжуватимуть збирати дані.

На зорі інтернету, коли служби аналітики тільки зароджувались, для аналітики підключалось зображення розміром 1x1 піксель, що знаходилось на сервері служби аналітики. Коли браузер користувача посилав запит на сервер служби аналітики, служба аналітики таким чином отримувала випадковий ідентифікатор користувача з Cookie, його IP, і, мабуть не більше. Таким чином можна було визначити кількість унікальних користувачів, що відвідали сайт, кількість повторних відвідувачів, їхню країну і іноді місто - по IP. І цього було цілком достатньо для реалій того часу. Саме через розмір картинки в один піксель багато сучасних служб аналітики, серед яких Facebook Pixel та VK Pikel мають в назві це слово.

Та час не стоїть на місці, служби аналітики стрімко рухаються вперед. Зараз вони відстежують всі дії на сторінці - кліки, скрол, натискання клавіш, динаміку взаємодії. До прикладу, в Яндекс.Метрики є служба вебвізор - вона дозволяє записувати і відтворювати цілі сесії взаємодії користувача. (Рисунок 1.1)

А за допомогою цих даних, статистичних моделей та машинного навчання служби аналітики можуть дізнатись вік користувача, його сімейний, фінансовий стан, тип характеру та багато інших речей, що дають причину для хвилювання.

Ще цікавіше, коли служба аналітики належить корпорації соцмережі, як Facebook Pixel. Тоді служба аналітики зв’язує відвідувача сайту з його профілем в соцмережі і вже знає його вік, сімейний стан, коло спілкування, куди він їздив у відпустку, а соцмережа знає які сайти відвідує користувач, робить точнішою його модель і знає, яку рекламу йому краще впарити, щоб збільшити свій прибуток.

Те, скільки про нас знають інтернет-гіганти просто вражає і дає привід для хвилювання. Інтернет-гіганти зацікавлені тільки в прибутку і їх не хвилює приватність даних користувачів. Останнім часом чути про багато викриттів та скандалів, особливо пов’язаних з корпорацією Facebook, що передавала персональні дані користувачів третім сторонам, а також була неодноразово спіймана на зберіганні даних, які вона зарікалась не збирати. Але не будем про погане.

## 1.3 Аналоги Choko.Link

### TapLink

TapLink - мабуть один найпопулярніших сервісів мультипосилання для Instagram. В нього багатий набір віджетів та налаштувань: можна створити просто візитку, можна зробити мікролендинг, а можна навіть приймати платежі і тримати міні-магазин. Більшість кольорових тем та базових віджетів доступно в безкоштовному тарифі, а в платних тарифах можна відображати блоки по розкладу, вставляти зображення та відео, інтегруватись з службами аналітики, платіжними системами та CRM.

Ціна: до 200 гривень в місяць.

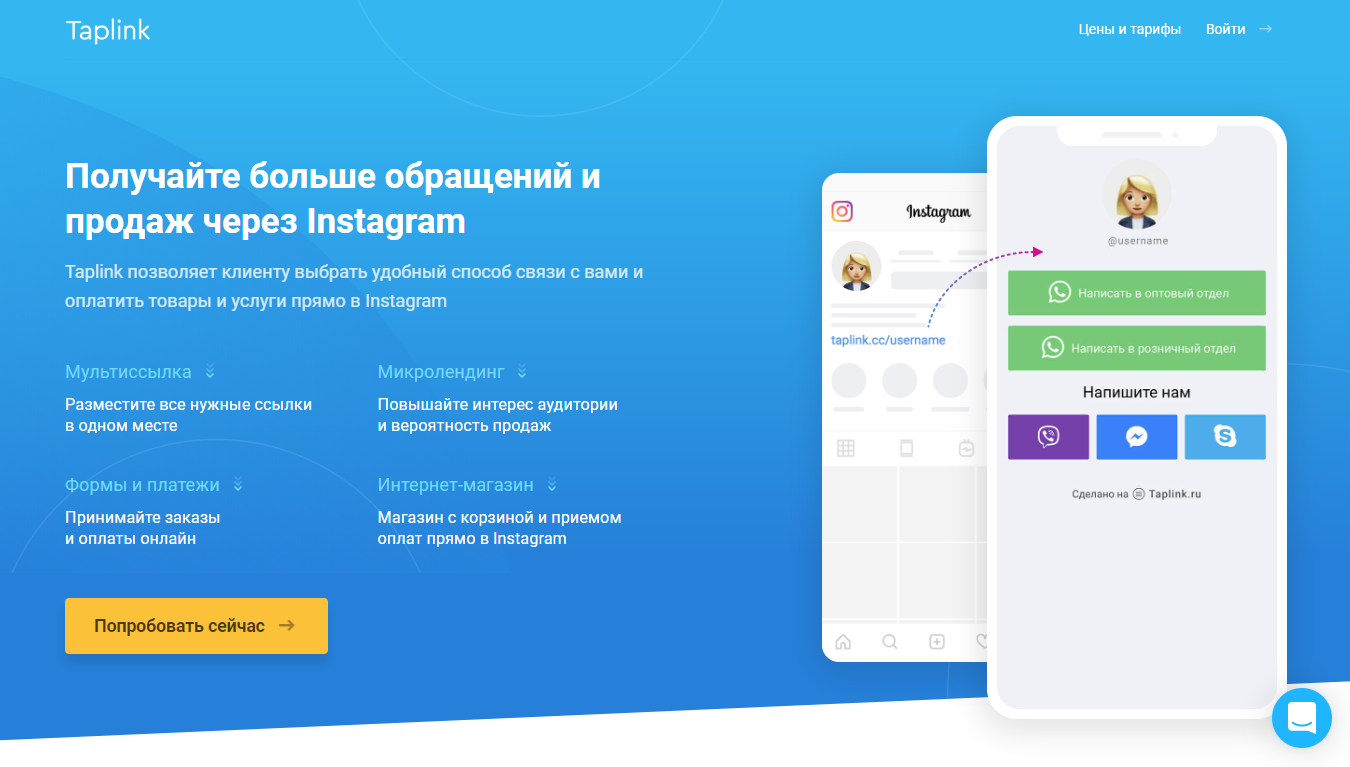


Рисунок 1.1 - Cайт TapLink

### Mssg.me

Mssg.me дозволяє створити красиву сторінку-візитку з месенджерами та інтернет-магазином. Також він дозволяє створити віджет на сайт, згенерувати QR-код і підпис для E-mail - і це все включено в безкоштовний тариф. З мінусів - можна додати тільки одне посилання на месенджер, чого дуже мало.

Ціна платного тарифу - 7 доларів в місяць, що трохи менше 200 гривень.

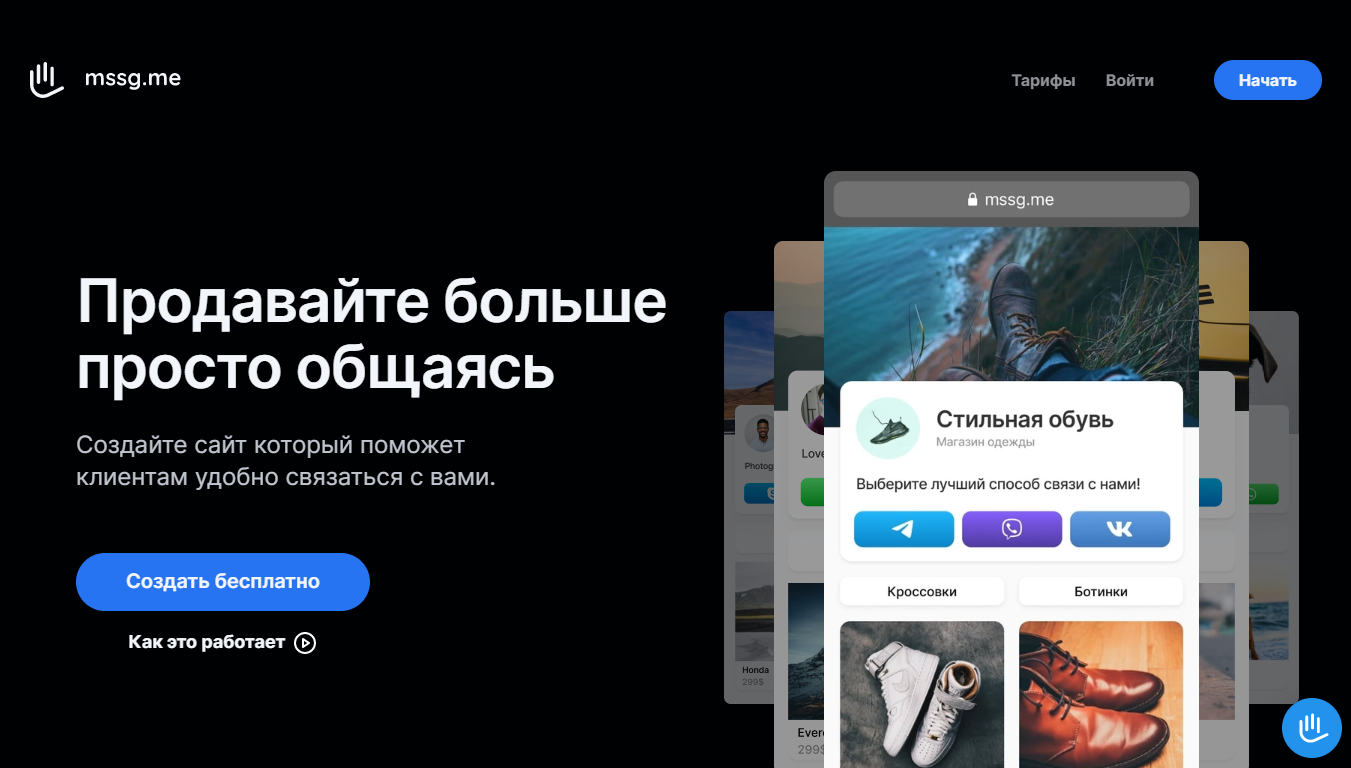


Рисунок 1.2 - Сайт mssg.me

### tapid.ru

TapID, як і попередні сервіси, дозволяє створити і сторінку-візитку, і сторінку з посиланнями, і інтернет магазин, і форму прийому заявок. В формі заявок можна налаштувати сповіщення в Телеграм і ВК, що є великим плюсом до зручності.

Ціна: до 150 гривень в місяць.

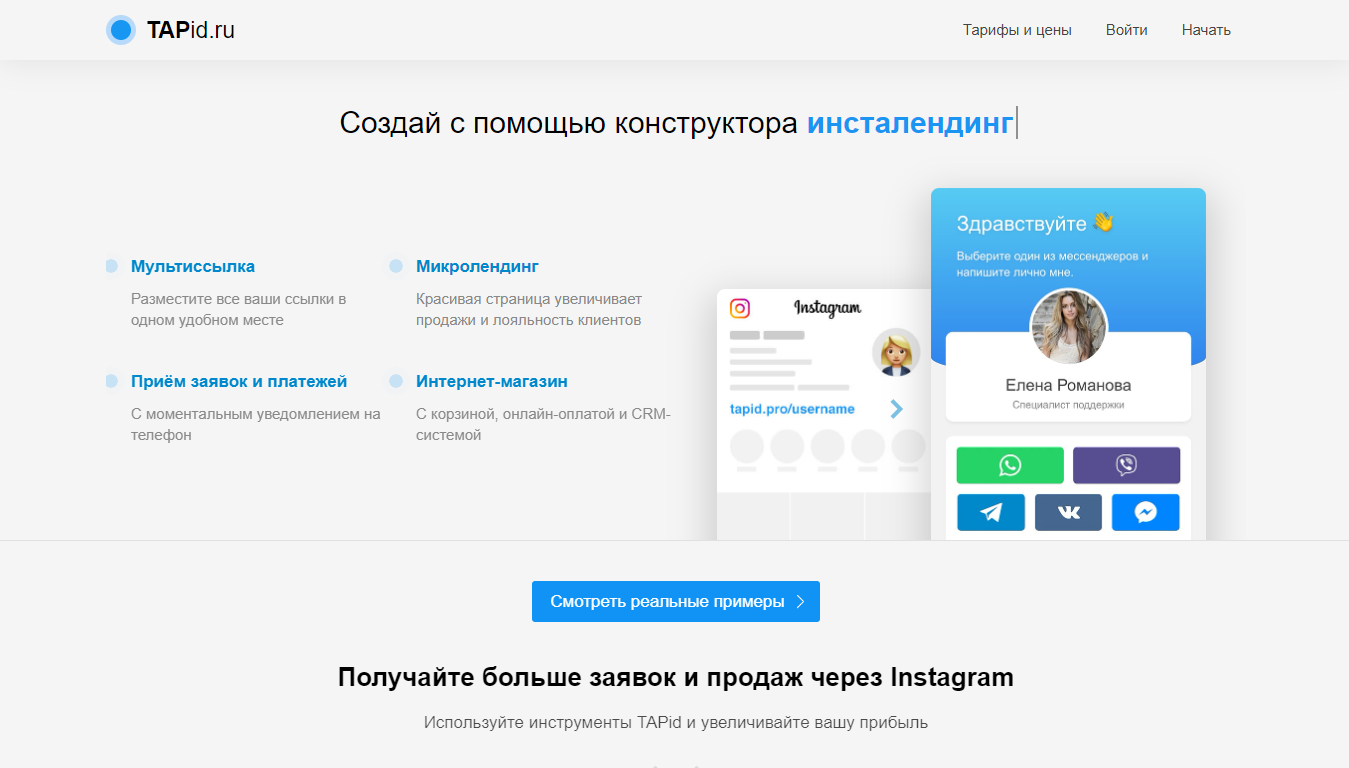


Рисунок 1.3 - Сайт TAPId.ru

### BMBE.ru

BMBE - просто монстр для роботи з Instagram. В нього є все що необхідно для просування в Instagram - сервіс для оргнізовування конкурсів в коментарях, визначення найактивніших підписників, агрегатор діплінків та сторінка-візитка, а також автопостинг, шаблони для текстів, аналіз конкурентів, хештегів, делегування доступу працівникам та аналітика. Функціонал тільки трішки перетинається з Choko.Link - в Choko.Link є тільки сторінка-візитка.

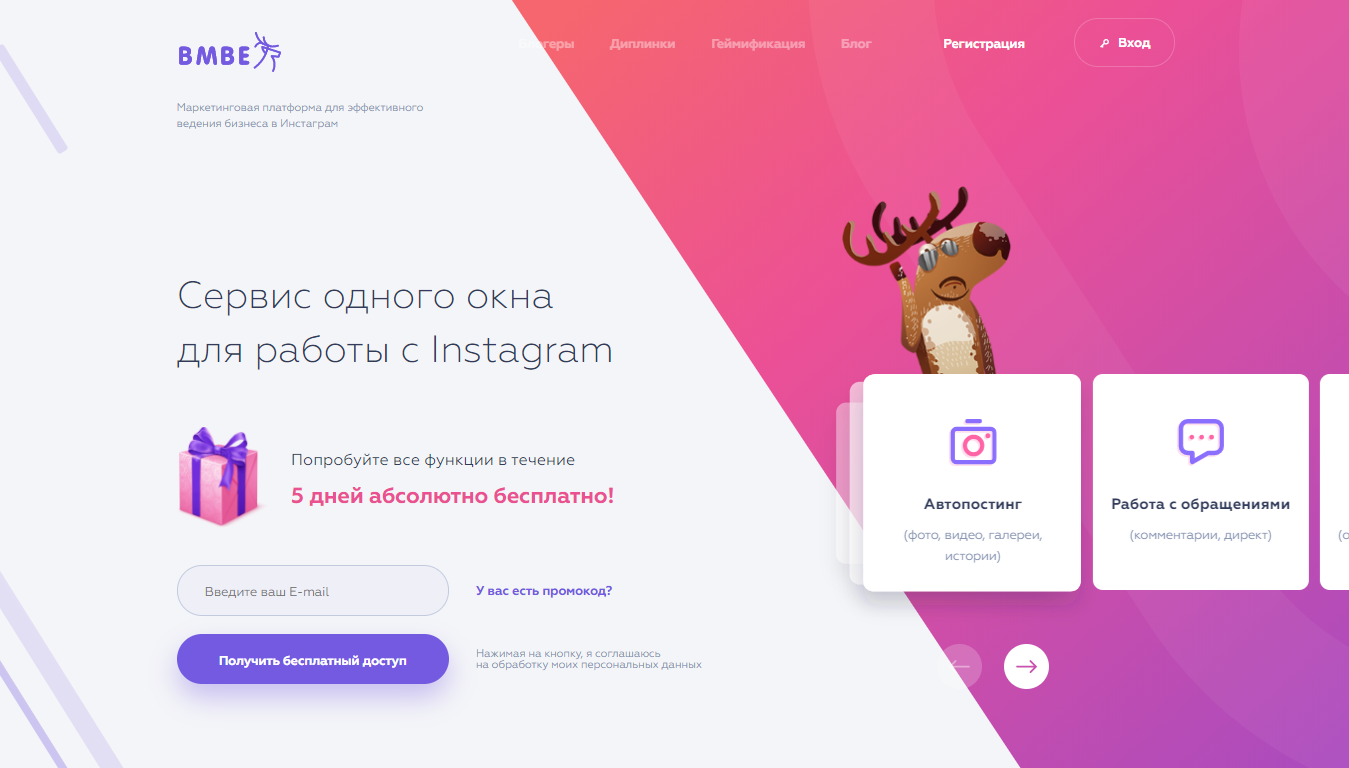


Рисунок 1.4 - Сайт BMBE.ru

### LinkTree

LinkTree - мабуть один з перших сервісів для мульти посилання, який я бачив. Його функціонал досить мінімалістичний, найбільш схожий з моїм сервісом. Також в нього досить багато функцій платні. Додавати посилання - це все що можна безкоштовно. Статистика, додаткові кольорові теми, форма збору заявок - доступні тільки на платній основі.

Ціна платного тарифу - 6 доларів в місяць, що трохи більше ніж 150 гривень.

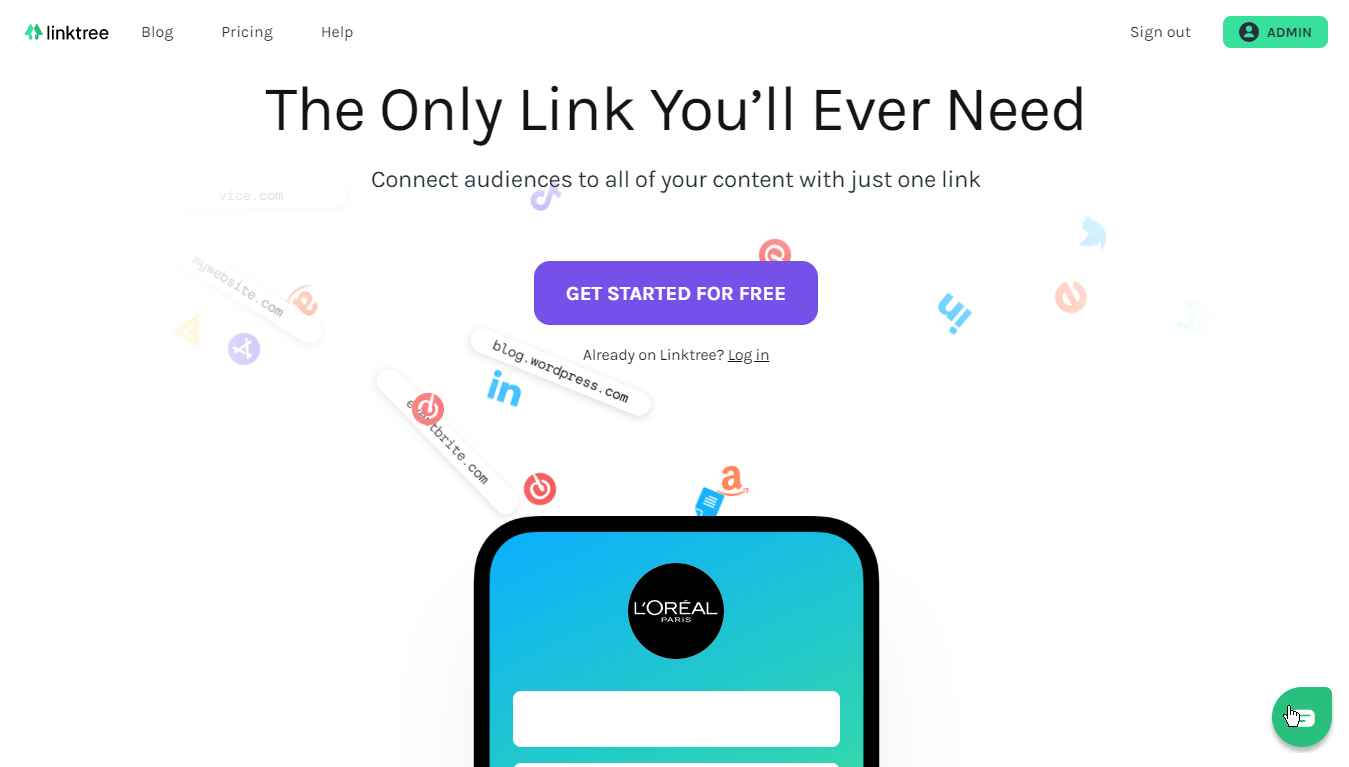


Рисунок 1.5 - Сайт LinkTree

# 2. ІНФОРМАЦІЙНЕ ТА МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

## 2.1 Веб-застосунок

Веб-сайт - це просто набір веб-сторінок, що розміщений на веб-сервері і доступний в мережі інтернет, об’єднаний одним доменним іменем. Веб-сайт може бути сайтом-візиткою, портфоліо, блогом, лендингом тощо.

Веб-застосунок (веб-додаток) - це комп’ютерна програма, що використовує браузер в якості свого клієнта. Веб-додаток може бути простим, типу списку завдань чи форма зворотнього зв’язку, так і складним, як текстовий редактор чи 3d-гра.

На відміну від веб-сайту, веб-додаток є інтерактивним. В ньому користувач може додавати чи редагувати інформацію, а не тільки читати, як на простих сайтах.

В порівнянні з нативними застосунками, в веб-застосунків є як переваги, так і недоліки.

Мабуть, найпомітнішою перевагою є кросплатформеність. Це означає, що веб-додаток не залежить ні від апаратного забезпечення, ні від операційної системи користувача (в адекватних межах). Його можна запускати як на комп’ютері, так і на смартфоні, годиннику, холодильнику чи іншому пристрої, який має браузер.

Також пропадають проблеми з оновленням - після оновлення сторінки користувач буде бачити нову версію. Для розробників це також перевага - в разі випуску “глючного” оновлення розробникам не прийдеться хвилюватись, що хтось не зможе відкотитись до попередньої версії - після оновлення сторінки користувачі знову будуть користуватись старою версією, звісно, за умови, що розробники зробили відкат.

Веб-додатки не потрібно завантажувати з магазинів додатків, таких як Google Play та AppStore, їх не потрібно встановлювати, таким чином це економить місце на пристрої користувачів. А розробникам таким чином легше випустити додаток, так як публікація додатка в магазинах не безкоштовна і, до того ж, займає час.

В веб-додатках більшість важливої логіки винесена на сервер, а отже такі додатки складніше “крякнути” чи “зламати”. З другого боку, так як клієнтська частина веб-додатків здебільшого пишеться на інтерпретовуваних мовах програмування, таких як JavaScript чи Dart, високорівневий код може з легкістю прочитати і інтерпретувати як браузер, так і користувач. В додатках, що компілюються в exe чи apk файл, код проходить оптимізацію і стає дуже низькорівневим, який потім складно прочитати і проаналізувати навіть за допомогою спеціалізованих інструментів. JavaScript код можна прочитати навіть за допомогою текстового редактора, приклавши лише трохи зусиль на форматування коду, якщо він був мініфікований. Іноді для захисту клієнтського коду його обфускують, тобто роблять таким, щоб комп’ютер його все ще розумів і міг виконувати, проте людина вже не змогла. Але, якщо запастись терпінням і кавою, обфускований код можна проаналізувати.

З недоліків, в першу чергу, для веб-застосунків, це залежність від інтернету. Хоча сучасні браузери надають потужні можливості для кешування, для повноцінної роботи веб-додатку не обійтись без інтернету. Крім того, браузери обмежують додатку доступ до різного апаратного забезпечення пристрою.

Компанії-розробники браузерів хочуть зменшити різницю між веб-додатками та простими. Вони розробили стандарт PWA, що означає Progressive Web Application - прогресивний веб-додаток. Такий веб-додаток можна встановити на комп’ютер чи телефон, в нього буде іконка, як в справжнього додатка, а коли його відкрити, в браузера не буде додаткових панелей, таких як адресний рядок чи панель закладок - на вигляд справжнісінький застосунок. Віднедавна PWA можна навіть опублікувати в магазині додатків Google Play.

Крім того, розробники впроваджують нові API, щоб для створення складного додатку не потрібно було працювати з низькорівневими системними API. Так, в Google Chrome вже є API для роботи з веб-камерою, мікрофоном, геопозиціонуванням, захопленням зображення з екрану (тобто можна реалізувати трансляцію екрана прямо з браузера без використання AnyDesk чи TeamViewer), можна навіть створювати примітивну музику за допомогою Web Audio API - не просто відтворювати mp3-файл, а саме генерувати хвилі різної частоти та форми.

Для збільшення продуктивності було введено TypedArray - типізовані масиви. Мається на увазі не масиви типу String чи Float, TypedArray - це масиви типів Uint8, Uint16 і подібних, низькорівнево оптимізованих саме для цього. Наприклад, для обробки аудіо буферів чи криптографії.

А зовсім недавно було впроваджено WebAssembly. WebAssembly - це новий крок в Front End розробці. Він дозволяє компілювати JavaScript в машинний код, таким чином збільшивши продуктивність коду, а також більше захистити його від втручання.

Web-розробка сьогодні - зовсім інакша ніж була 20, 10 чи навіть 5 років назад. Навіть за один рік встигає змінитись стільки, що дуже легко відстати від часу. Тому потрібно постійно стежити за новинками, пробувати щось нове, розвиватись.

## 2.2 LAMP-стек

LAMP - мабуть один з найбільш поширених стеків для побудови динамічних веб-сайтів та веб-додатків. Більшість сучасних веб-хостингів мають на борту саме LAMP, тобто серверна ОС на базі **L**inux, веб-сервер **A**pache, СКБД **M**ySQL, та мова програмування **P**HP (іноді - Pearl або Python) - саме це означає абревіатура LAMP. В LAMP є багато варіацій, наприклад, WAMP, де замість ОС Linux використовується Windows, або LEMP, де замість Apache використовується веб-сервер nginx (читається як Engine-X).



Рисунок 2.1 - Пінгвін - логотип та талісман Linux

**Linux** - загальна назва для операційних систем на базі ядра GNU/Linux. На відміну від Mac OS та Windows, Linux має відкритий код, він доступний всім безкоштовно для розповсюдження та модифікацій. Зазвичай для веб-хостингу використовують такі популярні збірки, як Ubuntu, CentOS чи CloudLinux. В кожної з ОС є свої переваги та недоліки, вони відрізняються стабільністю, частотою оновлення, хоча здебільшого головні відмінності лежать в архітектурі та ядрі, які не помітні користувачу з неозброєним оком.



Рисунок 2.2 - Логотип Apache

**Apache** - модульний веб-сервер з відкритим кодом. Він працює практично на всіх популярних платформах - як на Linux, так і на Windows і Mac OS та інших. Він обслуговує практично половину всіх веб-сайтів в інтернеті. Адміністратори обирають його завдяки гнучкості, потужності, поширеності та простоті. Його можна дуже просто розширити за допомогою динамічно завантажуваних модулів, а також він здатний виконати програми на більшості інтерпретовуваних мов програмування, без використання зовнішнього програмного забезпечення. Apache має гнучку структуру конфігурацій: крім глобальної конфігурації, що знаходиться в файлі httpd.conf, він підтримує конфігурацію рівня папки. Розташувавши в папці файл .htaccess, можна задати певні налаштування, які будуть діяти тільки для даної папки. Таким чином можна обмежити доступ до певних директорій, наприклад, по паролю, або в цілях безпеки вимкнути обробник PHP для файлів з папки, в яку можуть завантажувати прості користувачі.



Рисунок 2.3 - Логотип Nginx

**Nginx** (читається як Engine-X) - веб-сервер, розроблений російським програмістом Ігорем Сисоєвим. Має версії для UNIX-подібних систем та Windows. Завдяки подієво-орієнтованій архітектурі nginx здатний обробити одночасно безліч з’єднань. Він використовує, за нагоди, більш ефективні методи вводу-виводу операційної системи, чим ще більше збільшує продуктивність. В nginx відсутня попапкова конфігурація, що робить його менш гнучким, проте значно збільшує його швидкість.

Також Nginx використовують в зв’язці з Apache - вони чудово доповнюють один одного і збільшують продуктивність та відмовостійкість сервера. Поширеним варіантом є використання Nginx як реверс-проксі для Apache. В такому випадку nginx називають фронтендом, а Apache - бекендом. При такому Nginx буде обслуговувати всі вхідні запити клієнтів і ми отримуємо перевагу завдяки його можливості обробляти одночасно багато запитів. Nginx буде самостійно обробляти статичний контент і повертати користувачам файли, а для динамічного контенту буде передавати запит Apache, який, скоріше всього, передасть його PHP, а потім поверне результат. Ця конфігурація дає можливість горизонтально масштабувати сервер і встановити кілька Apache серверів і налаштувати маршрутизацію через Nginx, який буде самостійно розподіляти навантаження між ними.



Рисунок 2.4 - Логотип MySQL

**MySQL** - мабуть одна з найвідоміших, якісних і швидких СКБД серед існуючих. MySQL може похвалитись високою швидкістю роботи, стійкістю, простотою налаштування, вона доступна для більшості операційних систем, тому MySQL є чудовим вибором для невеликих проєктів. Вона, як і більшість СКБД, має відкритий код, тобто ми можемо вільно використовувати та розповсюджувати його. Також її код можна модифікувати, щоб адаптувати до власних потреб, наприклад, вирізати непотрібний функціонал щоб збільшити продуктивність. MySQL працює по клієнт-серверній моделі, тобто функціонує окремим процесом, а якщо хтось хоче скористатись базою даних, він встановлює з’єднання по виділеному порту (здебільшого - 3306) і надсилає запити мовою SQL щоб отримати чи змінити дані.

MySQL - далеко не єдина СКБД на ринку, проте вона є однією з найпопулярніших. Серед аналогів найпопулярнішими є PostgreSQL, MongoDB, Redis, Elasticsearch, SQLite. В кожної з них своя методика збереження даних, вони оптимізовані для різних видів завдань.



Рисунок 2.5 - Логотип PHP

**PHP** - одна з найпоширеніших мов програмування в галузі бекенд-розробки. І не спроста. Щоб написати сайт на PHP можна обійтись найпростішим текстовим редактором, не потрібно ніякого компілятора, важкої IDE і інших речей, які відлякують початківців. PHP створено спеціально для веб-розробки, тому PHP-код можна вставляти прямо в HTML. Абревіатура PHP являється рекурсивним акронімом і означає “PHP: Hypertext Preprocessor”. На відміну від Java та Node.js, в яких основний процес, що обробляє вхідні запити, PHP починає новий процес для кожного запита, який обробляє його, надсилає відповідь і завершується, звільняючи пам’ять і інші ресурси. Таким чином в PHP складно добитись великих втрат оперативної пам’яті - буквально через долі секунди пам’ять звільниться. Також PHP автоматично видаляє всі тимчасові файли, що пов’язані з завершеним процесом, що дуже зручно.

## 2.3 MVC

MVC означає “Model, View, Controller” - “Модель, Вигляд, Контролер” - один із архітектурних шаблонів розробки програмного забезпечення. Останнім часом вона стала доволі популярною в сфері веб-розробки. Суть шаблона MVC полягає в розділенні даних застосунку, його відображення (верстки) та керуючої логіки на три незалежні компоненти, таким чином зміна кожного компонента може здійснюватись незалежно один від одного.

* Model - компонент, що відповідає за структуру даних. Здебільшого, класи моделей реалізують функціонал додавання, видалення та редагування даних в базі або взаємодію з зовнішніми API.
* View - компонент, що відповідає за відображення інформації, одержане на виході. Здебільшого - це сторінки або фрагменти сторінок, такі як “шапка”, “футер” чи “навігація”. Також це може бути RSS-сторінка чи будь-який інший формат сторінки.
* Controller - компонент, що зв’язує між собою View, Model і інші ресурси, що необхідні для обробки HTTP-запиту та формування веб-сторінки. Саме завдяки контролеру застосунок реагує на дії користувача, отримує дані з моделі на відображає користувачу за допомогою View. По суті, це щось схоже на мозок MVC-застосунку.

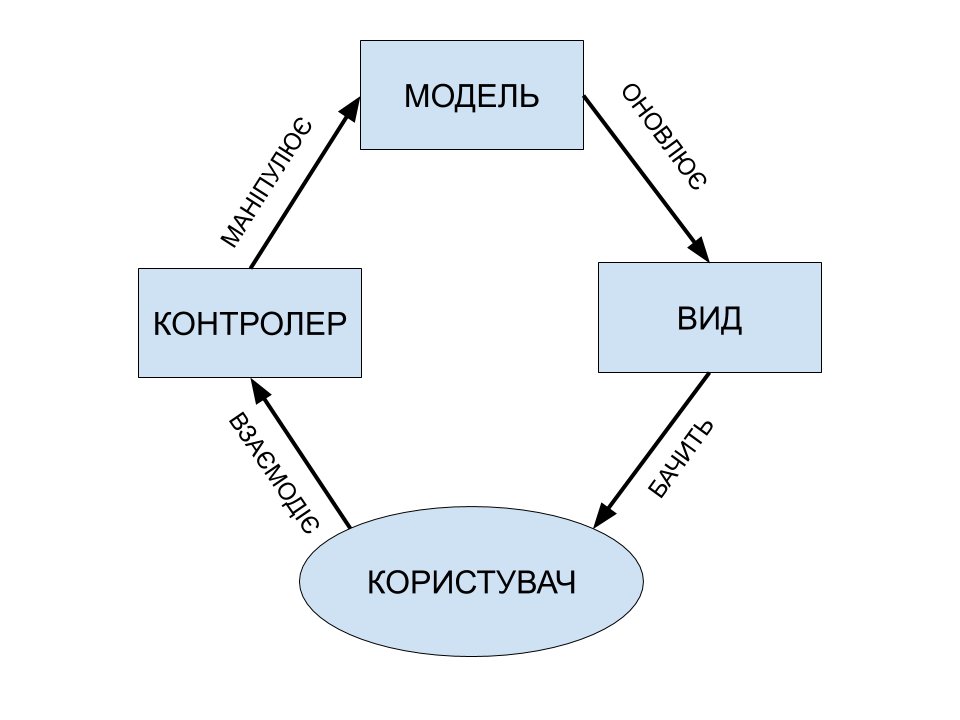


Рисунок 2.6 - блок-схема MVC

## 2.4 CodeIgniter

CodeIgniter - один з безлічі фреймворків, призначених для розробки веб-застосунків, написаних на PHP. Його призначення - зробити розробку веб-застосунків швидшою, ніж якби прийшлось розробляти все з нуля, надаючи набір готових бібліотек для частих потреб, таких як робота з базою, потоками вводу-виводу, валідацією даних тощо. CodeIgniter дозволяє творчо зосередитись на проєкті, мінімізуючи кількість коду, необхідного для написання того чи іншого функціоналу.



Рисунок 2.7 - Логотип CodeIgniter

CodeIgniter буде чудовим рішенням, якщо:

* Необхідна виняткова продуктивність.
* Потрібна широка сумісність з більшістю стандартних-провайдерів, які працюють на різних версіях та конфігураціях PHP.
* Хочеться фреймворк, який вимагає мінімум налаштування.
* Хочеться фреймворк, що не вимагає взаємодію через CLI.
* Потрібна ретельна, чітка документація.
* Ви уникаєте складності, надаючи перевагу простим рішенням.
* Не цікавлять масштабні монолітні бібліотеки типу PEAR.

Кожен фреймворк вимагає певних зусиль, щоб розібратись з ним. Розробники CodeIgniter намагались зробити цей процес максимально приємним. Також CodeIgniter є надзвичайно легким: його ядро використовує всього кілька малих бібліотек. Це помітно в порівнянні з іншими фреймворками, що вимагають набагато більше ресурсів. Додаткові бібліотеки підключаються на ходу, в залежності від потреби конкретного процесу, тож система є дуже легкою і швидкою. Також, як зазначалось вище, CodeIgniter використовує MVC-шаблон для розділення логіки, даних та верстки, що вже є неписаним стандартом в сфері веб-розробки. Ще однією перевагою є те, що CodeIgniter може працювати навіть без бібліотек-шаблонізаторів. Таким чином він досягає максимальної продуктивності.

Програмісти люблять писати код і ненавидять писати документацію, проте вони засмучуються, коли доводиться працювати з погано документованим інструментом. CodeIgniter має чудову документацію з прикладами коду, а код ядра фреймворка дуже чистий і містить коментарі.

Самі по собі функції досить мізерні самі по собі, але зібрані разом і правильно структуровані вони дуже допомагають в розробці свого веб-застосунку. Ось короткий список функцій CodeIgniter:

* MVC-архітектура
* Надзвичайна легкість
* Повнофункціональні класи для роботи з різними базами даних.
* Query Builder API
* Форми та валідація даних
* Безпека та XSS-фільтрування
* Робота з сесіями
* Клас для надсилання Email. Підтримує вкладені файли, як простий текст, так і HTML, різні протоколи (sendmail, SMTP, та Mail) тощо.
* Бібліотека для роботи з зображеннями (обрізання, зміна розміру, поворот та ін.). Підтримує GD, ImageMagick, та NetPBM.
* Клас для завантаження файлів на сервер
* FTP клас
* Локалізації
* Пагінація
* Шифрування
* Бенчмарки (заміри швидкодії різних етапів)
* Повне кешування сторінок
* Логування помилок
* Профілювання
* Клас календаря
* Клас User Agent
* Клас движка шаблонізатора
* Клас для юніт-тестування
* ЧПУ, тобто URL-адреси, що дружні до користувачів та пошукових механізмів
* Гнучкий URI-роутинг
* Величезна бібліотека з допоміжними функціями

Нижче зображено блок-схему того, як дані циркулюють в системі:

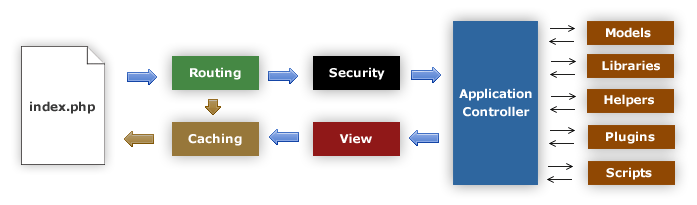


Рисунок 2.8 - Блок-схема роботи CodeIgniter

* index.php є єдиною точкою входу, що підключає та ініціалізує базові ресурси, необхідні для запуску CodeIgniter.
* Маршрутизатор (Router) аналізує HTTP-запит, і вирішує, що з ним робити далі.
* Якщо існує файл кешу, він надсилається безпосередньо в браузер, пропускаючи звичайне виконання системи.
* Безпека. В цілях безпеки HTTP-запит та всі вхідні дані фільтруються.
* Контролер завантажує моделі, основні бібліотеки, допоміжні функції та будь-які інші ресурси, необхідні для обробки конкретного запиту.
* Фінальна верстка генерується, а потім надсилається користувачу. Якщо кешування ввімкнено, сторінка спершу кешується, щоб надалі пропускати обробку запита.

Розробники CodeIgniter прагнули досягти максимальної гнучкості, продуктивності та працездатності у мінімальному можливому пакеті і, на мою думку, їм це вдалося. При розробці автори відкидали все, що не сприяє меті, вони заміряли, рефакторили, спрощували та переосмислювали на кожному етапі процесу.

З технічної точки зору CodeIgniter був розроблений з наступними цілями:

* Динамічне підключення. В CodeIgniter усі компоненти завантажуються не глобально, а динамічно, за запитом. Система не сміє робити ніяких припущень щодо того, що саме може знадобитись окрім мінімальних основних ресурсів, тому система за замовчуванням надзвичайно легка. HTTP-події та розроблені програмістом контролери та представлення будуть визначати, що викликається.
* Слабкий зв’язок. Чим менше компоненти залежні один від одного, тим більш гнучкою і універсальною є система. Ціллю авторів CodeIgniter була дуже слабо зв’язана система.
* Особливість компонентів. Особливість - це сфокусованість компонента на окремому призначенні. В CodeIgniter кожен клас і його функції дуже автономні і забезпечують максимальну користь.

CodeIgniter - це динамічна система з слабо зв’язаними особливими компонентами. Він прагне до простоти, гнучкості та високої продуктивності в невеликих пакетах.

Будь-який середньостатистичний PHP-програміст здатний розібратись з CodeIgniter за кілька днів.

## 2.5 Bootstrap

Спочатку Bootstrap розроблявся як внутрішній інструмент компанією Twitter, проте пізніше його опублікували для загального використання.

Twitter Bootstrap це CSS-фреймворк для швидкого створення сучасної, кросбраузерної і стандартизованої верстки. Продумана структура дає можливість створювати безліч самих різноманітних елементів інтерфейсу і сітку сайту.

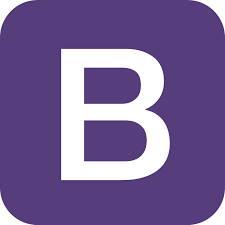


Рисунок 2.9 - Логотип Bootstrap

Основні компоненти Bootstrap:

* Сітки - заздалегідь задані адаптивні розміри колонок, які можна відразу ж використовувати, наприклад колонка з класом col-md-6 буде займати половину простору на великих екранах і буде на всю ширину на мобільних телефонах.
* Типографіка - класи для налаштування шрифтів, задання розміру та кольору тексту. Також описані стилі для оформлення цитат чи коду.
* Медіа - дає можливості оформлення зображень та відео.
* Таблиці - дає можливість стилізувати таблиці, задавати колір меж і навіть додає функціонал сортування
* Форми - класи для оформлення форм, полів вводу - не тільки текстових, а й випадаючих списків, радіокнопок, повідомлень валідації
* Навігація - оформлення для панелей навігації, вкладок і меню.
* Алерти - оформлення спливаючих вікон, підказок, алертів і діалогових вікон

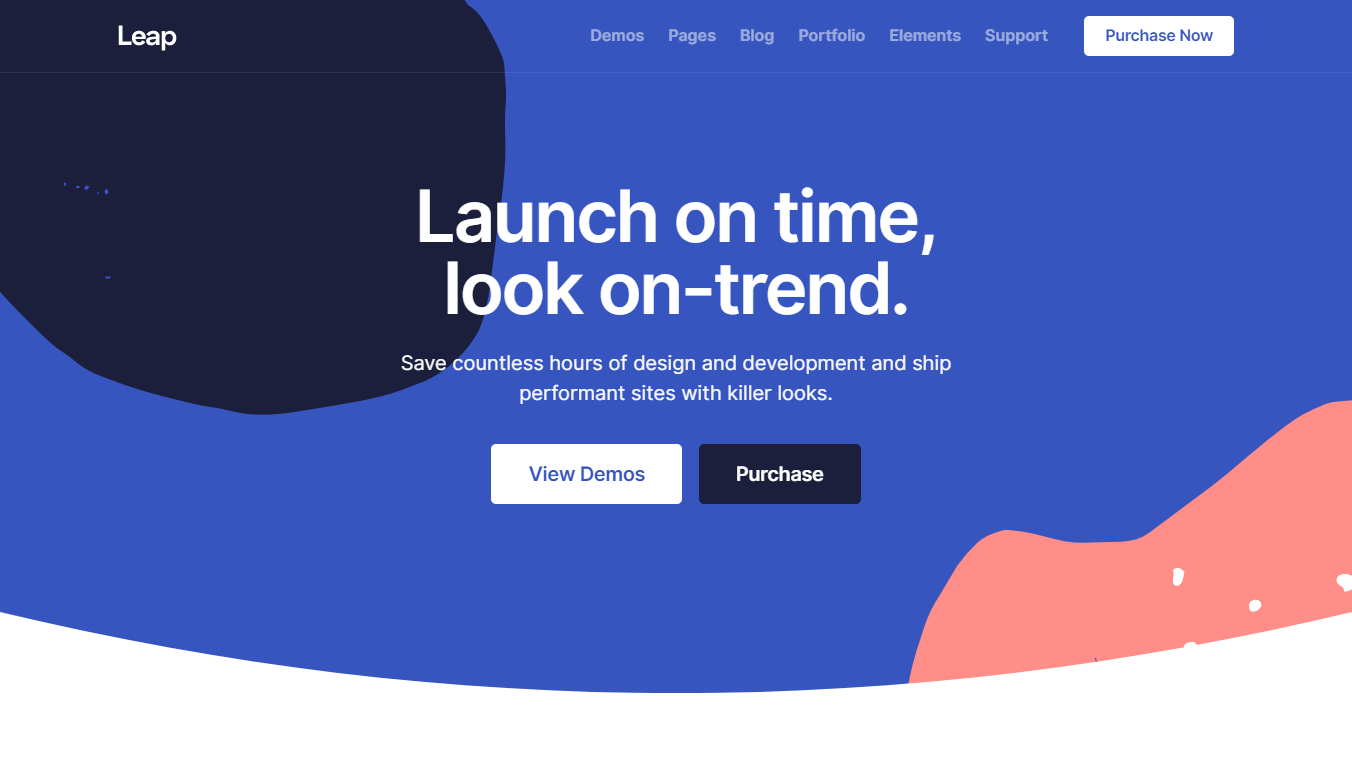


Рисунок 2.10 - Приклад сайту, зверстаного за допомогою Bootstrap

# 

# 3. ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

## 3.1 GIT

Перед початком роботи над будь-яким проєктом спершу слід налаштувати систему контроля версій. Система контроля версій - це система, що допомагає легко організувати зміни в коді, порівнювати дві різних версії проєкту. Завдяки ньому над одним проєктом одночасно можуть працювати багато людей. Сумніваюсь що хтось це читатиме. Мабуть найпопулярнішою системою контролю версій є Git. Його розробив Лінус Торвальдс, автор ядра операційної системи Linux, власне, для управління розробкою ядра Linux. Git має підтримку гілок, тобто можна працювати над копією проєкту, а потім злити зміни з оригіналом.

Git-репозиторій представляє собою папку, в якій лежать файли конфігурації репозиторія, файли журналів, що зберігають операції, виконані над репозиторієм, індекс, що описує розташування файлів і сховище, власне, з файлами. Структура сховища файлів не відображає структуру файлового дерева репозиторія, вона орієнтована на прискорення операцій, виконуваних над репозиторієм. Коли ядро обробляє команду зміни, воно створює в сховищі нові файли, що відповідають новим станам змінених файлів. З сховища файлів не видаляються старі стани файлів, тобто репозиторій зберігає всі версії всіх файлів.

Зазвичай репозиторій знаходиться в папці “.git” в кореневій папці проєкту. Будь-яку папку з файлами можна перетворити в Git-репозиторій. Також репозиторій може бути імпортований з мережі. При імпорті автоматично створюється робоча копія файлів, відповідаюча останньому зафіксованому стану репозиторія.

Git можна використовувати локально, хоча набагато зручніше користуватись Git-хостингом, таким як GitHub чи GitLab. Вони надають зручний веб-інтерфейс, завдяки ним можна опублікувати свій проєкт відкритим кодом, а також синхронізовувати версії проєкту між багатьма пристроями. Автори GitHub називають його соціальною мережею для розробників. Крім розміщення коду, тут можна коментувати код один одного, підписуватись на інших розробників, спілкуватись між собою.

Більшість програмістів працюють з Git за допомогою консолі, та я не належу до них. GitHub має досить зручний desktop-клієнт, тому я його використовую.

Ось так виглядає репозиторій в клієнті GitHub Desktop:

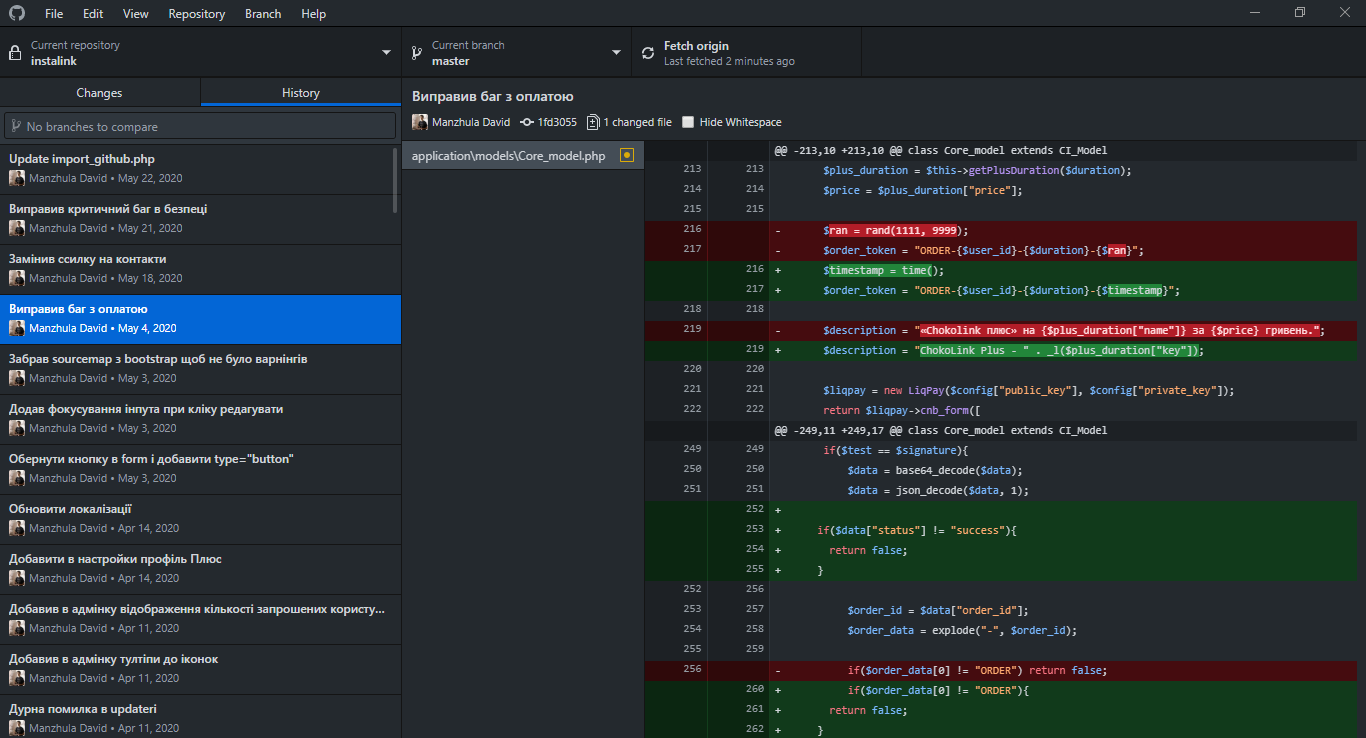


Рисунок 3.1 - Інтерфейс додатку GitHub

## 3.2 Середовище розробки

Для веб-розробки є багато редакторів коду та інтегрованих середовищ розробки (IDE).

Найпопулярніші серед них:

1. VisualStudio Code
2. WebStorm
3. PHPStorm
4. Atom
5. SublimeText
6. Notepad++
7. Vim

WebStorm і PHPStorm є розробкою компанії JetBrains. Вони між собою дуже схожі, просто PHPStorm заточений для PHP-розробки, а WebStorm - для веб-розробки взагалі.

Різниця між редактором і IDE полягає в глибині інтеграції. Редактор коду - це просто додаток, в якому можна редагувати код. Максимум, що там є крім стандартних функцій текстових редакторів - підсвітка синтаксису та автодоповненя. IDE, як каже сама його назва - це інтегроване середовище розробки. Це означає що він з коробки постачається з статичним аналізатором, що може вказати на слабкі місця коду не запускаючи його, лінтером, що перевіряє код на відповідність поширеним стандартам. Також IDE мають інтеграцію з системами контролю версій, такими як Git.

Для написання проєкту я вибрав редактор Notepad++, так як він дуже легкий і дуже швидко працює, тому що написаний на C++.

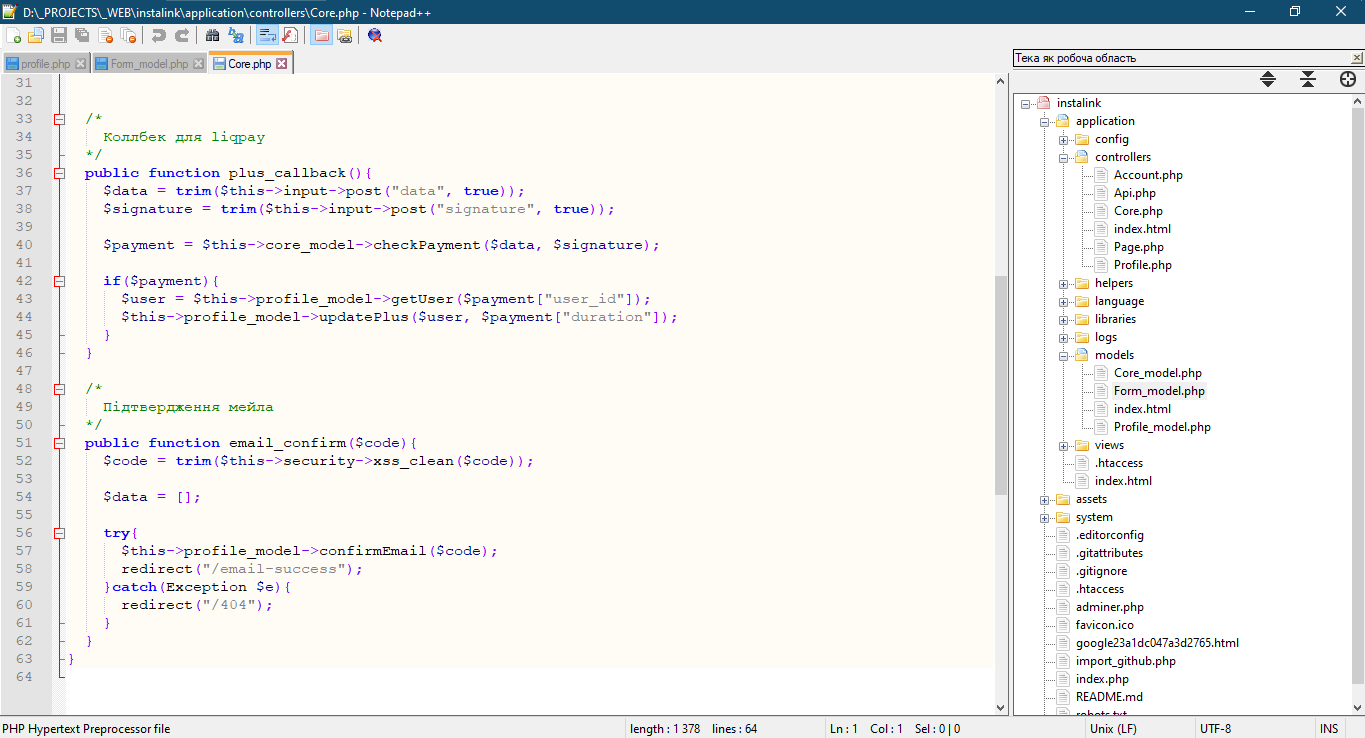


Рисунок 3.2 - Редактор Notepad++

## 3.3 Суть проєкта

Як було вище зазначено, в Instagram можна розмістити тільки одне посилання і, щоб обійти це обмеження, створюють так звані сторінки-візитки, на яких розміщують решту посилань та контактів.

Я вирішив створити власний сервіс для створення таких сторінок. Я назвав його ChokoLink. Частина “link” перекладається як “посилання”, “choko” означає шоколад, просто тому що тому що більшість людей люблять шоколад і в них приємні асоціації з цим словом. Спершу я хотів назвати проєкт instalink - instagram link, проте, як виявилось, ця назва була вже зайнята. До того ж, Instagram не одоблює назви брендів/сервісів, що містять в назві “Insta” або “Gram”. Все ж, в розробці використовується кодова назва instalink.

Сайт підтримує кілька мов - українську, російську та англійську. Завдяки правильно вибраній структурі, за бажання можна додати скільки завгодно мов не переписуючи при цьому код сайту - треба тільки створити папку з перекладами і вставити туди потрібні рядки. Проєкт написано на PHP фреймворку CodeIgniter, для верстки використовується відомий CSS-фреймворк Bootstrap 4.

Не забув я і про монетизацію. Сервіс монетизується за рахунок функціоналу, доступного по платній підписці. Безкоштовно на сайті є кілька обмежень, які ніяк не заважають більшості користувачів. Для прикладу, можна додати не більше 5 посилань, а кольорових тем доступна тільки четвертина.

Ось так виглядає посилання в профілі користувача (Рисунок 3.3), ось так виглядає сторінка з посиланнями (Рисунок 3.4).

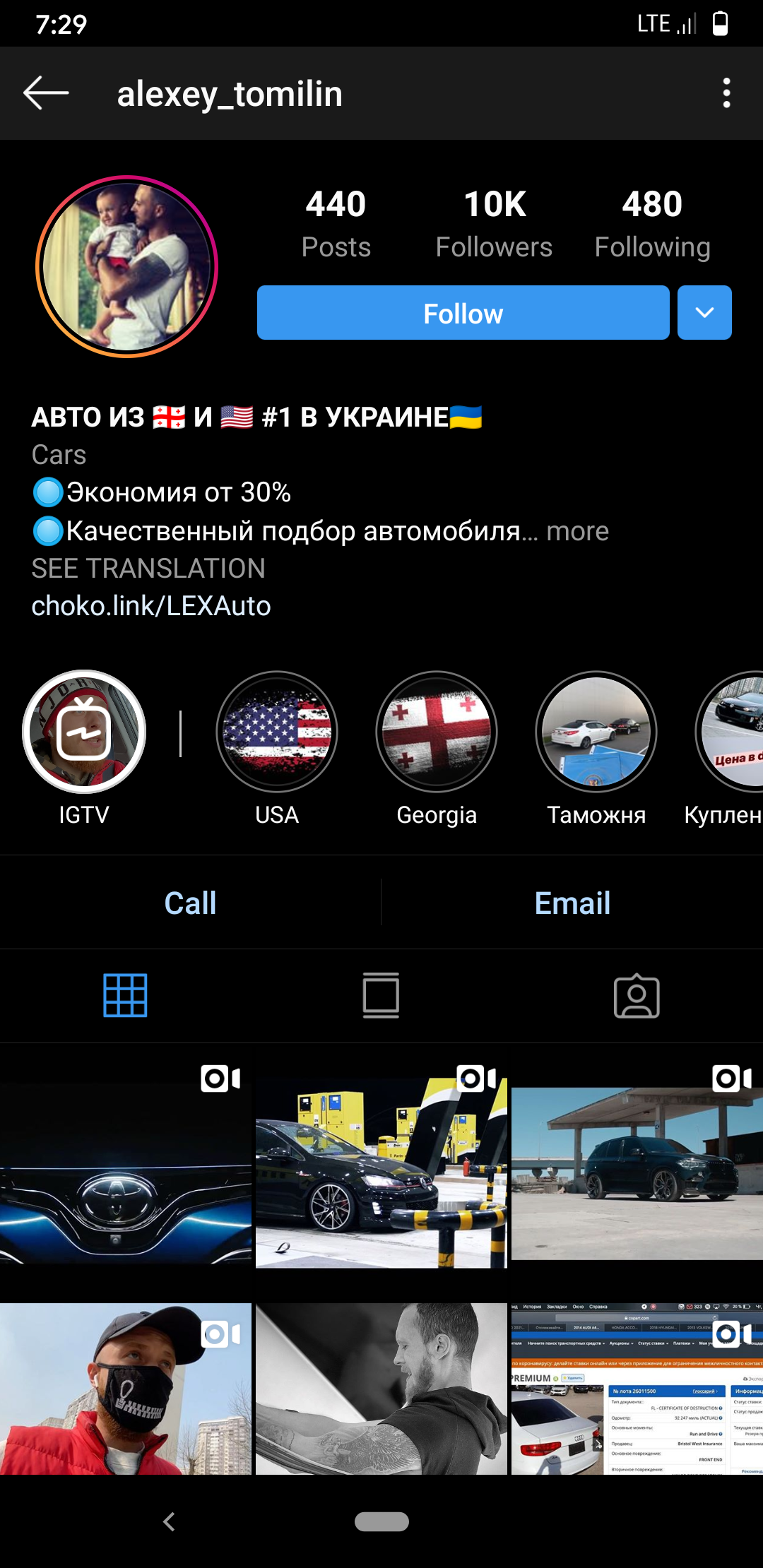


Рисунок 3.3 - Посилання на сторінку-візитку в профілі користувача

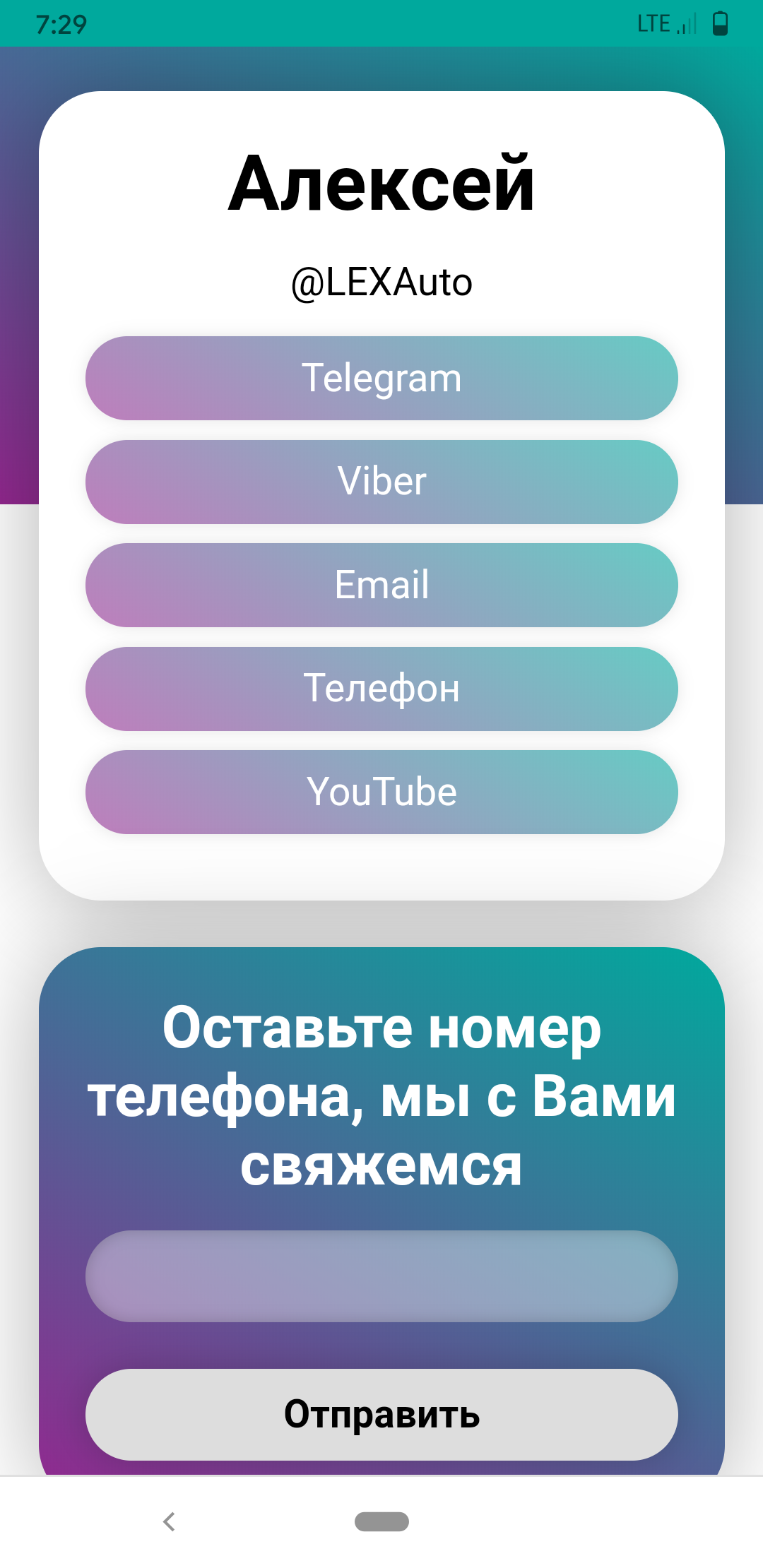


Рисунок 3.4 - Сторінка-візитка

В особистому кабінеті користувач може додавати посилання, упорядковувати, видаляти. Також можна налаштувати зовнішній вигляд сторінки - налаштувати відображуване ім’я та опис, також на вибір представлено 27 кольорових схем (в безкоштовному тарифі доступно тільки 6 перших), та 3 варіанта заокруглених кутів.

Ось так виглядає сторінка особистого кабінету (Рисунок 3.5)

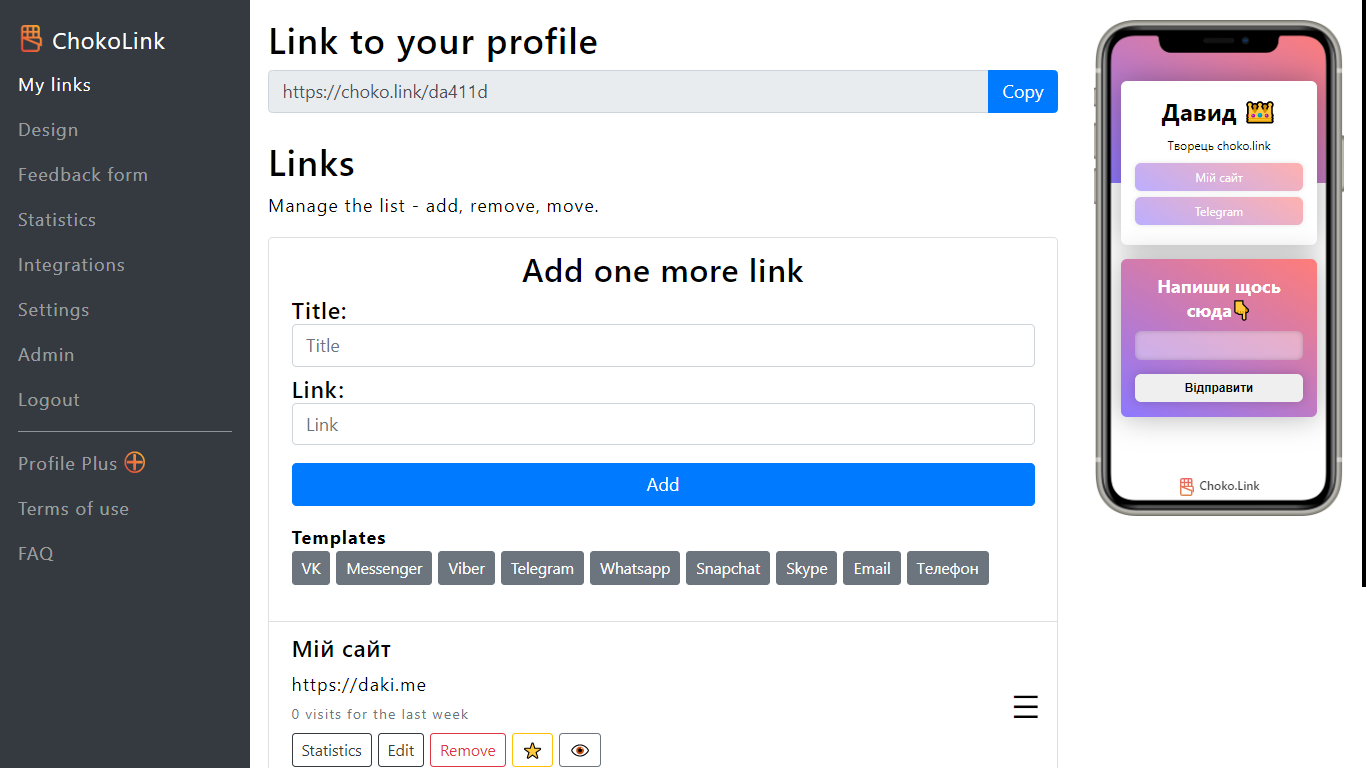


Рисунок 3.5 - Особистий кабінет

Ось такий вигляд має перший блок головної сторінки, що описує переваги сервісу та містить заклик до реєстрації: (Рисунок 3.6)

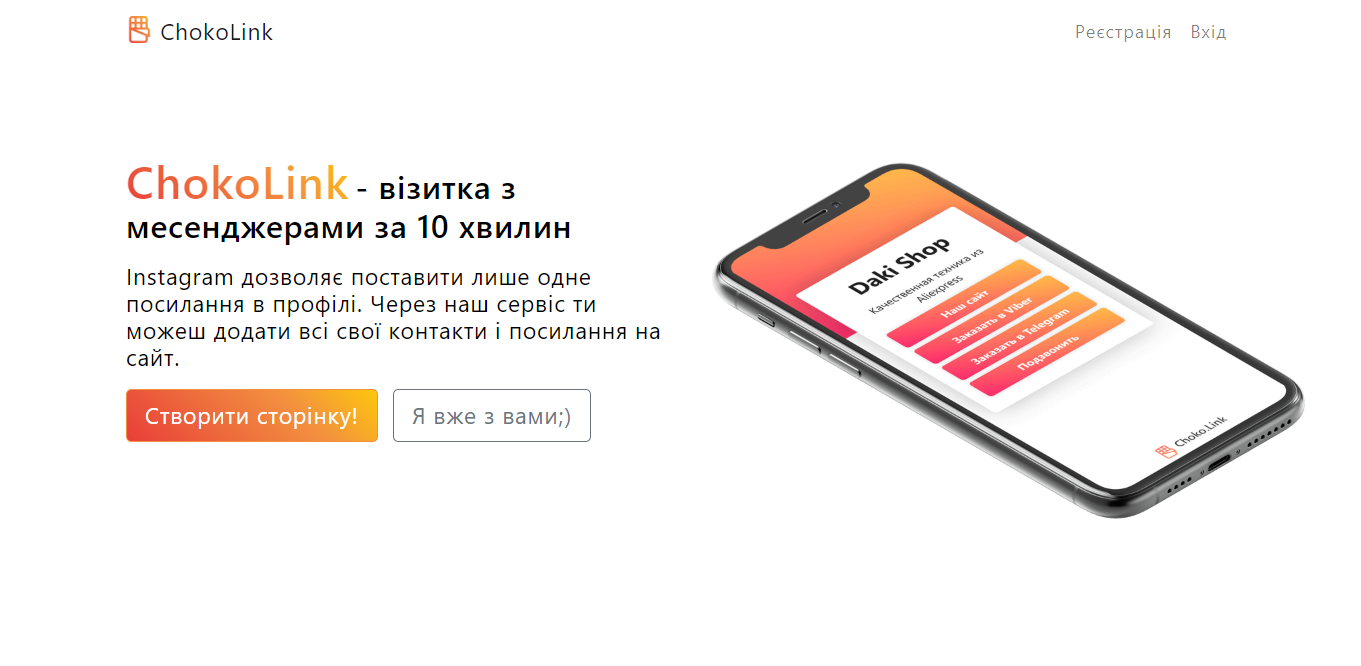


Рисунок 3.6 - Головна сторінка Choko.Link

Наступний вигляд має сторінка підключення платної підписки (Рисунок 3.7)

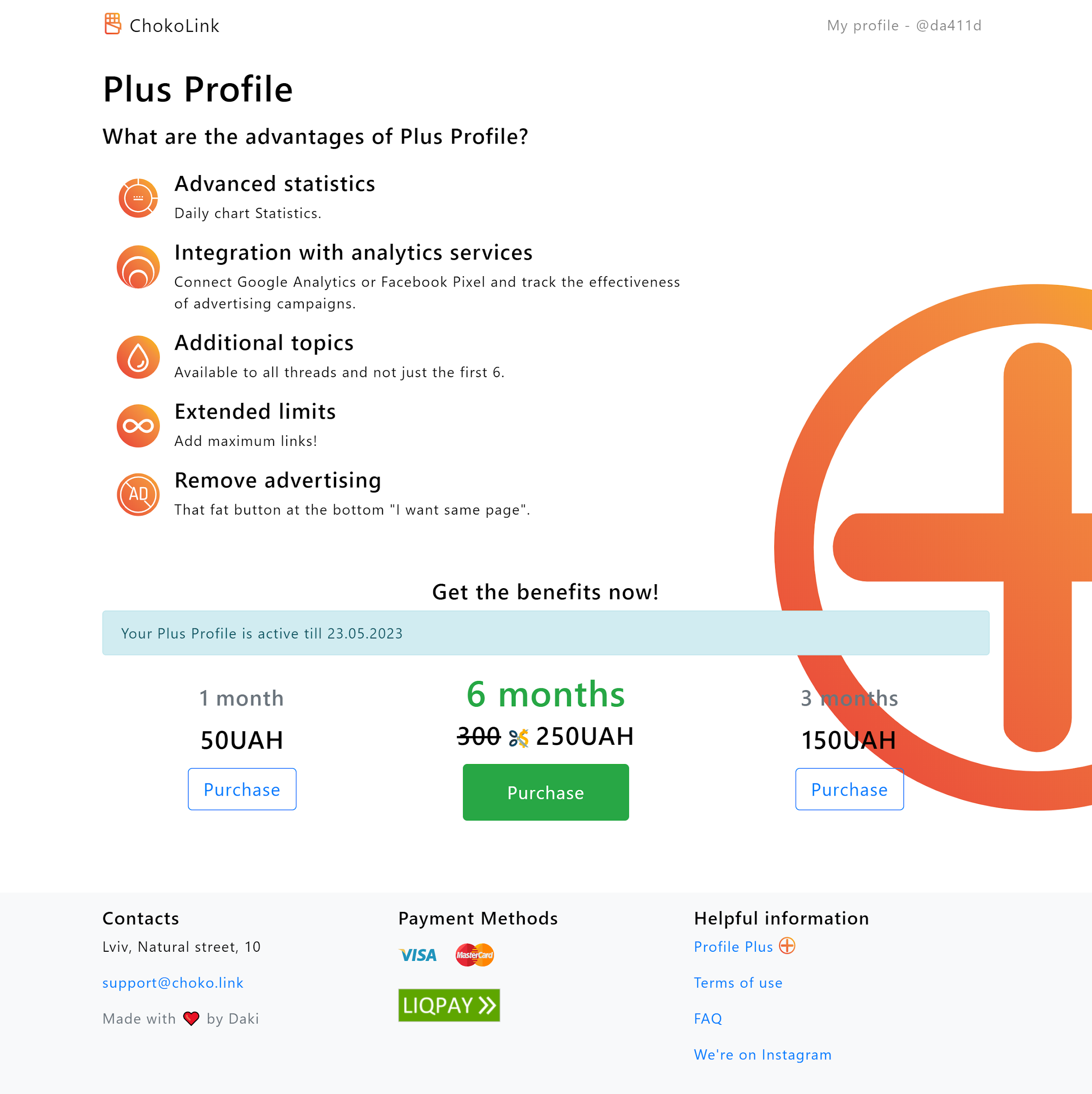


Рисунок 3.7 - Сторінка підключення платної підписки

## 3.4 Структура маршрутів

Маршрутизатор (роутер) - це модуль, що на вхід отримує рядок URI і вирішує, якому з контролерів передати керування. В своєму проєкті я використовую наступні контролери:

account - для особистого кабінету. Цей контролер обробляє вхід/вихід з облікового запису, реєстрацію, сторінку особистого кабінету і мінімальну адмін-панель.

api - саме завдяки цьому контролеру особистий кабінет взаємодіє з сервером, додає/видаляє посилання, міняє налаштування тощо.

page - контролер для статичних сторінок. Вони знаходяться в окремій папці і просто підключаються без всякої динаміки.

profile - сторінка-візитка. Та, якою користувач ділиться з користувачами (Рисунок 3.4)

Нижче наведено приклад частини конфігурації маршрутизатора

|  |
| --- |
| $route['default\_controller'] = 'page/index';  $route['(?!my)([a-zA-Z0-9]{2})'] = 'page/index/home/$1';  // STATIC PAGES  $route['faq'] = 'page/index/faq';  $route['faq/(:any)'] = 'page/index/faq-$1';  $route['policy'] = 'page/index/policy';  $route['email-success'] = 'page/index/email-success';  $route['password-success'] = 'page/index/password-success';  // Translations  $route['(?!my)([a-zA-Z0-9]{2})/(:any)'] = 'page/index/$2/$1';  // API  $route['API/(.+)'] = 'api/index/$1';  // ACCOUNT  $route['my'] = 'account/index';  $route['my/preview'] = 'account/preview';  $route['my/admin'] = 'account/admin';  $route['my/logout'] = 'account/logout';  // LOGIN/LOGOUT  $route['login'] = 'account/login';  $route['register'] = 'account/register';  $route['password'] = 'account/password';  // PLUS  $route['my/plus'] = 'account/plus';  $route['my/plus/(:num)'] = 'account/plus/$1';  // PAGE  $route["([a-zA-Z0-9\\_]{3,})"] = 'profile/index/$1';  $route["([a-zA-Z0-9\\_]{3,})/widget"] = 'profile/widget/$1';  $route["([a-zA-Z0-9\\_]{3,})/(:any)"] = 'profile/index/$1/$2'; |

## 3.5 Структура проєкту

Структура файлів проєкту має наступний вигляд:

|  |
| --- |
| ├───application  │ ├───config  │ ├───controllers  │ ├───helpers  │ ├───language  │ │ ├───en  │ │ ├───ru  │ │ └───uk  │ ├───libraries  │ │ ├───LiqPay  │ │ └───SxGeo  │ ├───logs  │ ├───models  │ └───views  │ ├───email  │ ├───layout  │ ├───pages  │ ├───profile  │ └───static  │ ├───en  │ ├───ru  │ └───uk  ├───assets  │ ├───css  │ │ └───style  │ ├───img  │ │ └───icons  │ └───js  │ ├───lang  │ └───script  └───system |

* В папці system розташовані файли ядра фреймворка CodeIgniter. При розробці проєкту вони майже не змінювались (крім кількох патчів безпеки).
* В папці assets розташовані файли для front-end частини - css-стилі, зображення, іконки та javascript-файли. Javascript-файли розкладено по папках, які потім комбінуються за допомогою .htaccess RewriteRule. Наприклад, якщо перейти по адресі /assets/js/sctipt.js (насправді такого файла на сервері немає), сервер поверне всі скрипти, що знаходяться в папці /assets/js/script/. Так само працюють css-стилі.
* Папка application - папка з проєктом CodeIgniter. В ній знаходяться папки з моделями, контролерами, файлами конфігурацій, версткою та іншими.
  + В папці config знаходяться файли конфігурацій. Там є як загальні файли конфігурації сайту, так і конфігурація бази даних, автозавантаження модулів, маршрутизації.
  + Папка controllers містить файли класів контролерів
  + В папці helpers лежить код функцій-помічників. Наприклад, функція для перетворення коду кольору з hex в rgb, чи функція для генерування красивого але відносно складного пароля.
  + Папка language містить файли локалізацій - переклади сайту. Файл локалізації - це PHP-файл, який оголошує масив ключ-значення з рядками та їх перекладами (Рисунок 3.8)
  + Папка libraries призначена, щоб саме там розміщати зовнішні бібліотеки. Я використовував бібліотеку SxGeo для визначення міста по IP-адресі і бібліотеку LiqPay для роботи з платіжною системою LiqPay.
  + В папці logs записуються логи помилок чи для відлагодження. Я її не використовую.
  + Папка models містить файли класів моделей.
  + Папка views містить верстку проєкту.
    - В підпапці email знаходяться шаблони для надсилання електронної пошти - для підтвердження пошти та для відновлення пароля
    - Папка layout - це головні частини сторінки - шапка сайту, футер та блок head.
    - pages - динамічні сторінки, типу сторінки входу/реєстрації, особистого кабінету тощо.
    - profile - шаблони для сторінки користувача.
    - static - статичні сторінки типу політики конфіденційності, головної сторінки, розділу допомоги. Кожна сторінка зберігається в трьох екземплярах - перекладена на три мови.

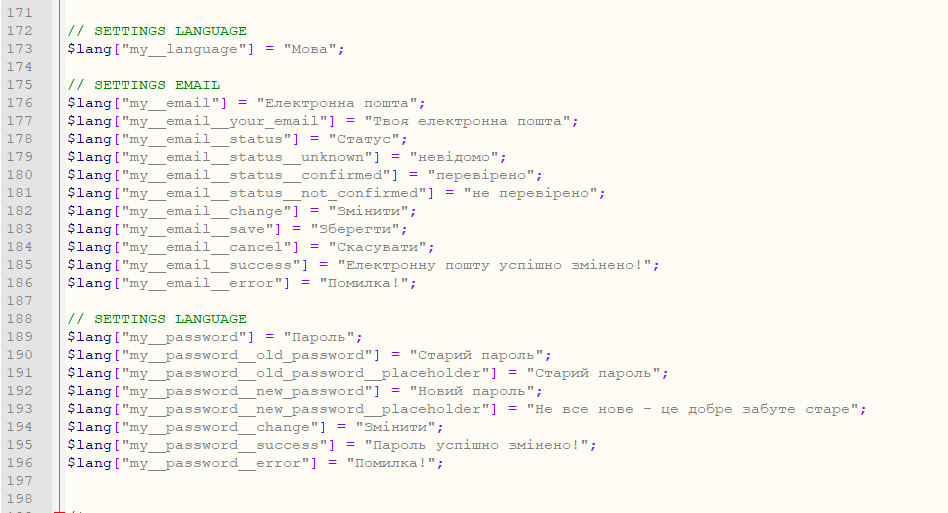


Рисунок 3.8 - Файл локалізації

## 3.6 Структура бази даних

Таблиця “profiles”. В ній зберігається список користувачів та їх налаштування.

* id (int) - ідентифікатор користувача
* username (string) - логін користувача
* password (string) - md5-хеш пароля
* email (string) - електронна пошта
* email\_confirmed (boolean) - флаг чи електронна пошта була підтверджена
* email\_confirm\_token (string) - код підтвердження електронної пошти
* pixel (string) - ідентифікатор Facebook Pixel для підключення аналітики
* google\_analytics (string) - ідентифікатор Google Analytics для підключення аналітики
* plus\_expires (timestamp) - термін дії платної підписки
* name (string) - ім’я користувача, відображається на візитці
* description (string) - опис, також відображається на візитці
* lang (string) - двохзначний код мови інтерфейсу
* theme\_color (int) - номер кольорової теми
* theme\_corner (int) - номер варіанта заокруглення кутів
* links (string) - масив посилань користувача в форматі JSON
* form (string) - конфігурація форми зворотнього зв’язку в форматі JSON
* register (timastamp) - мітка часу реєстрації
* last\_login (timestamp) - мітка часу останнього входу
* last\_redirect (timestamp) - мітка часу останнього переходу по посиланню
* referral (string) - логін користувача, від якого користувач зареєструвався (якщо є)

Таблиця “history” містить історію переходів по посиланнях сервісу. Потім ці дані використовуються для аналітики. За допомогою cron-завдань дані, що старіші ніж 30 днів автоматично видаляються.

* id (int) - порядковий номер запису, чисто ключове поле
* link\_id (string) - ідентифікатор посилання
* ip (string) - IP-адреса користувача, що перейшов по посиланню
* country (string) - країна
* city (string) - місто
* ref (string) - джерело переходу
* timestamp (timestamp) - мітка часу переходу

Таблиця “form\_messages” зберігає повідомлення, залишені в формі зворотнього зв’язку. При видаленні повідомлення насправді воно не видаляється з бази, а встановлюється флаг deleted = true.

* id (int) - ідентифікатор повідомлення
* timestamp (timestamp) - мітка часу повідомлення
* target (int) - ідентифікатор користувача-власника форми
* message (string) - текст повідомлення
* deleted (boolean) - флаг чи є повідомлення видаленим

## 3.7 Влаштування сторінки візитки

Мабуть найважливішою сторінкою, сторінкою, яку бачать навіть не зареєстровані користувачі, є сторінка-візитка. Вона складається з:

* Імені користувача
* Опису сторінки
* Список посилань
* Форма зворотнього зв’язку (якщо увімкнена)
* Код аналітики Facebook Pixel (якщо увімкнений)
* Код аналітики Google Analytics (якщо увімкнений)
* Код реферальної системи



Рисунок 3.9 - код аналітики Facebook Pixel

Так як сторінка-візитка є візитною карткою нашого сервіса, вона має працювати стабільно і швидко на всіх пристроях. Для оптимізації швидкості я забрав всі можливі зовнішні ресурси, такі як css-стилі, зображення та javascript-код. Вони всі занесені прямо в файл, тобто стилі знаходяться в тезі <style>, а зображення закодовані в формат base64. Тобто сторінка завантажується всього одним http-запитом, за виключенням служб аналітики, які підключаються вже з своїх cdn-серверів. Сторінка достатньо красиво виглядає як на ПК, так і на телефоні, як на сучасних браузерах, так і на Internet Explorer.

На (зображенні) приклад підключення аналітики Facebook Pixel.

На рядках 371-375 знаходиться захист від спрацювання служб аналітики на локальному сервері. Рядки 376-392 - це стандартний код підключення Facebook Pixel.

Після цього, якщо сторінка відкрита не напряму в браузері, а підключена як віджет на сайт, ми очікуємо повідомлення ззовні що віджет було відкрито. Інакше - зразу надсилаємо в службу аналітики стандартну подію “PageView”, що дає їм знати, що користувач переглянув сторінку (рядки 393-401). Після цього робимо, щоб при кліку на посилання надсилалась кастомна подія “LinkClick”, що дасть знати службі аналітики, що користувач перейшов по посиланню (рядки 402-409). Далі надсилаємо подію “Contact” коли користувач почне заповнювати форму зворотнього зв’язку - рядки 410-414. В кінці підключається зображення, огорнуте в тег <noscript> - це означає що зображення завантажується тільки в випадку, якщо відключений javascript в браузері. Це досить старий спосіб відстежувати навіть тих користувачів, в яких вимкнено javascript, хоча сумніваюсь, що в наш час хтось так робить. Також сумніваюсь, що хтось дочитає до цього місця.

Після цього блока знаходиться ідентичний блок для Google Analytics.

Далі йде код примітивної реферальної системи, що дозволяє відстежувати, від якого користувача прийшло найбільше клієнтів. Цей до болі примітивний код зображено нижче на рисунку 3.10.

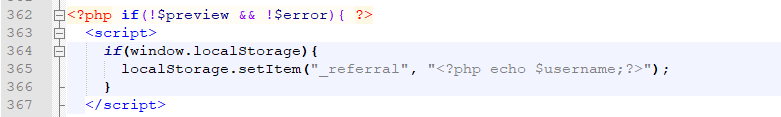


Рисунок 3.10 - код реферальної системи

Форма зворотнього зв’язку також не без секретів. Для захисту від ботів, в першу чергу, робилась фільтрація по User-Agent. Також я використовував просте, але надійне javascript-рішення. Ось так виглядає приховане рішення для захисту від ботів (Рисунок 3.11):



Рисунок 3.11 - Код форми зворотнього зв’язку

В рядку 265 находится приховане поле з іменем “\_XSRF”. Ця комбінація букв не означає абсолютно нічого, просто красиво виглядає і співзвучна “CSRF”, про яке мова буде нижче.

В 266 рядку знаходиться поле з іменем “\_CSRF”. В цьому полі зберігається CSRF-токен, який служить для захисту від CSRF-атак. Ця функція реалізована на рівні ядра CodeIgniter (модуль Security) і відсікає всі POST-запити, що не пройшли перевірку. В нашому випадку цей токен спеціально розвернутий задом-наперед, тобто токен як би неправильний і перевірка не буде пройдена.

В рядку 269 знаходиться поле для вводу повідомлення. Прошу помітити, воно без імені, тобто якщо надіслати форму просто так, це поле не буде включено до запиту.

В 272 рядку ми додаємо колбек-функцію на ввід в текстове поле.

В 273-278 виконуються дії, не зв’язані з захистом - це просто автопідстановка висоти і фільтрування поля вводу від не-цифр якщо це поле вводу номера телефона.

279-285 рядки - наш захист. Що тут відбувається:

1. CSRF-токен знов розвертається задом-наперед, тобто він стане правильним
2. Наше безіменне поле вводу отримує ім’я, що складається з слова “input” та значення поля “\_XSRF”
3. Назва поля “\_XSRF” змінюється на іншу, хоча дуже схожу. Це робиться для того, щоб зміна відбулася тільки один раз.

Тобто, за допомогою javascript, форма може бути надісланою тільки в тому випадку, якщо справжній користувач власноруч вводив повідомлення.

Сервер, отримавши запит, спершу перевіряє CSRF-токен. Це робиться автоматично на рівні ядра фреймворка. Далі сервер бере значення поля “\_XCRF”, наприклад, воно буде “qbme1e”. Насправді в ньому закодована мітка часу. Якщо мітка часу розходиться з теперішнім часом більш ніж на 60 хвилин - повідомлення не буде збережено. Таким чином відсіюється повторне використання коду. Після цього сервер додасть зліва до значення поля слово “input”, отримуючи рядок “inputqbme1e”. Якщо в запиті було передано значення з таким ключем - повідомлення передається далі для валідації та збереження в базу.

Таким чином, без застосування дратуючої системи CAPTCHA, суть якої полягає в введені коду з картинки, автоматично відсіюється більша частина спам-ботів. При бажанні, звісно, можна розібратись в способах цього захисту і розробити рішення його обходу, проте для базових випадків це рішення більш ніж чудово підходить.

# ВИСНОВКИ

Було розроблено сервіс для створення сторінки-візитки для подальшого використання в соцмережах Instagram та інших. Була реалізована інтеграція з службами веб-аналітики Google Analytics та Facebook Pixel, розроблена форма зворотнього зв’язку. Також була передбачена можливість вставити візитку на сайт в вигляді плаваючого віджета.

Сервіс був запущений перевірений на практиці - був придбаний домен та хостинг, сервіс був завантажений в інтернет, працював деякий час і продовжує працювати. Станом на 10 червня на сайті зареєстровано 325 користувачів, з них 4 користувача оплатили платну підписку і користуються розширеним функціоналом, що ще раз підтверджує його корисність.

Під час розробки сервісу я ознайомився з принципами розробки веб-застосунків за допомогою PHP-фреймворка CodeIgniter, реалізував гнучку структуру застосунка, а також спробував себе в ролі дизайнера, копірайтера та маркетолога, коли оформляв лендинг та запускав рекламу (яка провалилась), щоб залучити більше користувачів. Також я набув досвід організації часу та завдань, використовуючи Trello для обліку ідей вдосконалення та багів, що слід виправити.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

Sypex Geo - https://sypexgeo.net/

Mozilla Developer Network - https://developer.mozilla.org/uk/

Вікіпедія - uk.wikipedia.org

Хабр - https://habr.com/

https://www.hostinger.ru/rukovodstva/

CodeIgniter User Guide - https://codeigniter.com/userguide3/index.html

Bootstrap - https://getbootstrap.com/

GitHub - https://github.com/

Notepad++ - https://notepad-plus-plus.org/

http://nltu.edu.ua

NIC.UA - https://nic.ua/

LiqPay - https://www.liqpay.ua/documentation/uk/

Google Translate - https://translate.google.com/

Instagram - https://instagram.com/

https://ru.hexlet.io/

YouTube - https://www.youtube.com/

tproger - https://tproger.ru/

PHP - https://www.php.net/

OpenServer - https://ospanel.io/

Яндекс.Метрика - metrika.yandex.ua

FreelanceHunt - https://freelancehunt.com/

Денис Каплунов - Нейрокопірайтинг

# ДОДАТКИ

## Account\_controller.php

<?php

defined("BASEPATH") OR exit("No direct script access allowed");

class Account extends CI\_Controller

{

private $user;

private $language;

public function \_\_construct(){

parent::\_\_construct();

$this->load->model("profile\_model");

$this->load->model("core\_model");

$this->load->helper("form");

$this->load->helper("url");

$this->load->helper("b6413");

$this->load->helper("lang");

$this->load->library("pagination");

$this->load->config("core", true);

$this->user = $this->profile\_model->getUser();

$this->language = $this->core\_model->getLangCode($this->user);

$this->lang->load([

"header",

"footer",

"email",

"error",

"strings",

], $this->language);

}

public function index($param = ""){

if(!$this->user){

header("Location: /login?back=" . b6413\_encode("my"));

return;

}

$data = [];

$data["user"] = $this->user;

$data["lang"] = $this->language;

$data["title"] = "my\_profile";

$data["form\_field\_types"] = $this->core\_model->getFormFieldTypes();

$data["theme\_colors"] = $this->core\_model->getColors($this->user["plus"]);

$data["theme\_corners"] = $this->core\_model->getCorners($this->user["plus"]);

$data["langs"] = $this->core\_model->getLanguages();

$data["nav\_links"] = [];

$data["nav\_links"][] = [

"title" => "header\_\_links",

"link" => "/my#links",

"plus\_only" => false,

];

$data["nav\_links"][] = [

"title" => "header\_\_design",

"link" => "/my#design",

"plus\_only" => false,

];

$data["nav\_links"][] = [

"title" => "header\_\_form",

"link" => "/my#form",

"plus\_only" => true,

];

$data["nav\_links"][] = [

"title" => "header\_\_statistics",

"link" => "/my#statistics",

"plus\_only" => true,

];

$data["nav\_links"][] = [

"title" => "header\_\_integrations",

"link" => "/my#integrations",

"plus\_only" => true,

];

$data["nav\_links"][] = [

"title" => "header\_\_settings",

"link" => "/my#settings",

"plus\_only" => false,

];

$admins = $this->config->item("admins", "core");

if(in\_array(strtolower($this->user["username"]), $admins)){

$data["nav\_links"][] = [

"title" => "header\_\_admin",

"link" => "/my/admin",

"plus\_only" => false,

];

}

$data["nav\_links"][] = [

"title" => "header\_\_logout",

"link" => "/my/logout",

"plus\_only" => false,

];

$this->load->view("layout/head", $data);

$this->load->view("pages/account", $data);

$this->load->view("layout/foot", $data);

}

/\*

Логінимся

\*/

public function login(){

$user = $this->user;

$back = trim($this->input->get("back", true));

$back = trim(b6413\_decode($back));

$back = str\_replace(["//", ":"], "?", $back);

$back = preg\_replace("/([^0-9a-zA-Z\-\/\=\.\$\?])/", "", $back);

if($user){

if($back){

header("Location: {$back}");

}else{

header("Location: /my");

}

return;

}

$username = $this->input->post("login", true);

$password = $this->input->post("password", true);

if(strlen($username) && strlen($password)){

if($this->profile\_model->login($username, $password)){

if($back){

header("Location: {$back}");

}else{

header("Location: /my");

}

}else{

$data["error\_mess"] = \_l("LOGIN\_OR\_PASSWORD\_INCORRECT");

}

}

$data["lang"] = $this->language;

$data["user"] = $user;

$data["title"] = "login";

$data["navbar\_light"] = true;

$data["csrf"] = [

"name" => $this->security->get\_csrf\_token\_name(),

"hash" => $this->security->get\_csrf\_hash(),

];

$this->load->view("layout/head", $data);

$this->load->view("layout/header", $data);

$this->load->view("pages/login", $data);

$this->load->view("layout/foot", $data);

}

/\*

Відновлюєм пароль

\*/

public function password(){

$user = $this->user;

if($user){

header("Location: /my");

return;

}

$email = $this->input->post("email", true);

if(strlen($email)){

try{

$this->profile\_model->recoverPassword($email);

header("Location: /password-success");

}catch(Exception $e){

$data["error\_mess"] = \_l($e->getMessage());

}

}

$data["lang"] = $this->language;

$data["user"] = $user;

$data["title"] = "password-recover";

$data["navbar\_light"] = true;

$data["csrf"] = [

"name" => $this->security->get\_csrf\_token\_name(),

"hash" => $this->security->get\_csrf\_hash(),

];

$this->load->view("layout/head", $data);

$this->load->view("layout/header", $data);

$this->load->view("pages/password", $data);

$this->load->view("layout/foot", $data);

}

/\*

Розлогінюємся

\*/

public function logout(){

$this->profile\_model->logout();

header("Location: /login");

}

/\*

Реєструємся

\*/

public function register(){

$user = $this->user;

if($user){

header("Location: /my");

return;

}

$data = [];

$error\_mess = false;

$act = trim($this->input->post("act", true));

$name = trim($this->input->post("name", true));

$description = trim($this->input->post("description", true));

$login = trim($this->input->post("login", true));

$password\_id = trim($this->input->post("rand", true));

if(!strlen($password\_id)) $password\_id = "1";

$password\_id = md5($password\_id \* 2);

$password = trim($this->input->post($password\_id, true));

$email = trim($this->input->post("email", true));

$referral = trim($this->input->post("referral", true));

if($act == "register"){

try{

$this->profile\_model->registerUser([

"name" => $name,

"description" => $description,

"login" => $login,

"password" => $password,

"email" => $email,

"referral" => $referral,

]);

$this->profile\_model->login($login, $password);

if($this->profile\_model->getUser()){

header("Location: /my");

return;

}

}catch(Exception $e){

$error\_mess = \_l($e->getMessage());

}

}

$data["field\_salt"] = rand(111111, 99999999);

$data["field\_id"] = md5($data["field\_salt"] \* 2);

$data["error\_mess"] = $error\_mess;

$data["navbar\_light"] = true;

$data["name"] = $name;

$data["description"] = $description;

$data["login"] = $login;

$data["email"] = $email;

$data["password"] = $password;

$data["lang"] = $this->language;

$data["user"] = $user;

$data["title"] = "register";

$data["csrf"] = [

"name" => $this->security->get\_csrf\_token\_name(),

"hash" => $this->security->get\_csrf\_hash(),

];

$this->load->view("layout/head", $data);

$this->load->view("layout/header", $data);

$this->load->view("pages/register", $data);

$this->load->view("layout/foot", $data);

}

/\*

Профіль плюс

\*/

public function plus($duration = false){

$user = $this->user;

$data = [];

$data["title"] = "Профіль Плюс";

$data["lang"] = $this->language;

$data["user"] = $user;

$data["plus"] = $user["plus"];

$data["plus\_expires"] = $user["plus\_expires"];

$data["navbar\_light"] = true;

if($user && $duration){

echo $this->core\_model->genPaylink($user["id"], $duration);

echo "<script>document.querySelector('#liqpay\_form').submit()</script>";

return;

}

$data["features"] = \_l("my\_\_plus\_\_features");

$data["payment\_durations"] = $this->core\_model->getPlusDurations();

$this->load->view("layout/head", $data);

$this->load->view("layout/header", $data);

$this->load->view("pages/plus", $data);

$this->load->view("layout/footer", $data);

$this->load->view("layout/foot", $data);

}

/\*

Адмінка

\*/

public function admin($param = ""){

$user = $this->user;

if(!$this->user){

header("Location: /login?back=" . b6413\_encode("my/admin"));

return;

}

$admins = $this->config->item("admins", "core");

if(!in\_array(strtolower($this->user["username"]), $admins)){

header("Location: /my");

return;

}

$action = trim($this->input->get("action", true));

if($action == "plus"){

$id = (int) trim($this->input->get("id", true));

$targetUser = $this->profile\_model->getUser($id);

$duration = ((int) trim($this->input->get("duration", true))) \* 24 \* 60 \* 60;

$this->profile\_model->updatePlus($targetUser, $duration);

header("Location: ?");

return;

}

$data = [];

$data["lang"] = $this->language;

$data["user"] = $user;

$data["title"] = "Адмінка";

$total\_users = count($this->profile\_model->getUsers());

$counters = [

"day" => count($this->profile\_model->getUsers(["register >" => time() - 24\*3600])),

"week" => count($this->profile\_model->getUsers(["register >" => time() - 7\*24\*3600])),

"month" => count($this->profile\_model->getUsers(["register >" => time() - 28\*24\*3600])),

"total" => $total\_users,

];

$data["counters"] = $counters;

$filter["offset"] = $this->input->get("offset", true);

$filter["limit"] = 50;

$users = $this->profile\_model->getUsers($filter, true);

$data["users"] = $users;

$base = "?" . preg\_replace("/&offset=([0-9]+)/", "", $\_SERVER["QUERY\_STRING"]);

$this->pagination->initialize([

"base\_url" => $base,

"total\_rows" => $total\_users,

"num\_links" => 7,

"cur\_tag\_open" => "<span class=\"btn btn-sm btn-secondary active\">",

"cur\_tag\_close" => "</span>",

"attributes" => [

"class" => "btn btn-sm btn-secondary",

],

"per\_page" => $filter["limit"],

"page\_query\_string" => true,

"query\_string\_segment" => "offset",

]);

$data["pagination"] = $this->pagination->create\_links();

$data["filter"] = $filter;

$this->load->view("layout/head", $data);

$this->load->view("layout/header", $data);

$this->load->view("pages/admin", $data);

$this->load->view("layout/foot", $data);

}

/\*

Прев"ю

\*/

public function preview(){

if(!$this->user){

header("Location: /login?back=" . b6413\_encode("my/preview"));

return;

}

header("Location: /{$this->user["username"]}/preview");

}

}

## Profile\_controller.php

<?php

defined("BASEPATH") OR exit("No direct script access allowed");

class Profile extends CI\_Controller

{

public function \_\_construct(){

parent::\_\_construct();

$this->load->model("profile\_model");

$this->load->model("core\_model");

$this->load->model("form\_model");

$this->load->library("user\_agent");

$this->load->helper("color");

$this->load->helper("lang");

$this->user = $this->profile\_model->getUser();

$this->language = $this->core\_model->getLangCode($this->user);

$this->lang->load([

"header",

"footer",

"email",

"error",

"strings",

], $this->language);

}

public function index($u = -1, $param = false){

$data = [];

$user = $this->profile\_model->getUser($u);

if(!$user){

$this->error404();

return;

}

if($param){

if($this->profile\_model->try\_redirect($u, $param)){

return;

}

}

if($this->input->method() == "post"){

$success = $this->processForm($user["id"]);

header("Location: " . ($success ? "#result-success" : "#"));

return;

}

$data["username"] = $user["username"];

$data["name"] = $user["name"];

$data["description"] = $user["description"];

$data["profile\_links"] = $user["links"];

$data["plus"] = $user["plus"];

$data["pixel"] = $user["pixel"];

$data["ga"] = $user["google\_analytics"];

$data["theme\_color"] = $user["theme"]["color"];

$data["theme\_corner"] = $user["theme"]["corner"];

$data["preview"] = $param == "preview";

$data["embed"] = $param == "embed";

$data["error"] = false;

$data["lang"] = $user["lang"];

// Форма

$data["form"] = $user["form"];

$data["form\_token"] = base\_convert(time() + rand(-60, 60), 10, 36);

$data["csrf"] = [

"name" => $this->security->get\_csrf\_token\_name(),

"hash" => $this->security->get\_csrf\_hash(),

];

$this->load->view("profile/profile", $data);

if(!$data["preview"]){

$this->profile\_model->pushHistory($user["id"] . "-" . $user["username"]);

}

}

public function processForm($target){

// Validating temp token

$protection = $this->input->post("\_XCRF", true);

$protectionTimestamp = base\_convert($protection, 36, 10);

$now = time();

$is\_bot = $protectionTimestamp < $now - 60\*15 || $protectionTimestamp > $now + 60;

$fieldId = "input" . $protection;

$message = trim($this->input->post($fieldId, true));

if(empty($message)){

return false;

}

$this->form\_model->add($target, $message, $is\_bot);

return true;

}

public function error404(){

http\_response\_code(404);

$data = [];

$data["name"] = \_l("profile\_\_error");

$data["description"] = \_l("profile\_\_error\_\_page\_not\_found");

$data["profile\_links"] = [];

$data["plus"] = true;

$data["pixel"] = false;

$data["ga"] = false;

$cnt = count($this->core\_model->getColors());

$data["theme\_color"] = $this->core\_model->getColor(rand(1, $cnt));

$data["form"] = false;

$data["preview"] = false;

$data["error"] = true;

$data["lang"] = $this->language;

$this->load->view("profile/profile", $data);

}

public function widget($u = -1){

header("Content-type: text/javascript");

$data = [];

$user = $this->profile\_model->getUser($u);

if(!$user || !$user["plus"]){

return;

}

$data["username"] = $user["username"];

$data["name"] = $user["name"];

$data["description"] = $user["description"];

$data["profile\_links"] = $user["links"];

$data["plus"] = $user["plus"];

$data["theme\_color"] = $user["theme"]["color"];

$data["theme\_corner"] = $user["theme"]["corner"];

$data["base\_url"] = base\_url();

$data["lang"] = $user["lang"];

$data["preview"] = $data;

$this->load->view("profile/widget", $data);

}

}

## Form\_model.php

<?php

defined("BASEPATH") OR exit("No direct script access allowed");

class Form\_model extends CI\_Model

{

private $profiles\_table = "profiles";

private $table\_messages = "form\_messages";

public function \_\_construct(){

parent::\_\_construct();

$this->load->database();

$this->load->library("user\_agent");

}

function add($target, $message, $is\_bot = false) {

// Фільтруєм отримувача

$target = (int) $target;

if($target <= 0){

return;

}

// фільтруєм повідомлення

$message = trim($this->security->xss\_clean($message));

if(mb\_strlen($message) > 512){

$message = mb\_substr($message, 0, 512);

}

if(empty($message)){

return;

}

// Якщо юзерагент бота - значить все-таки бот

if($this->agent->is\_robot()){

$is\_bot = true;

}

$this->db->insert($this->table\_messages, [

"timestamp" => time(),

"target" => $target,

"message" => $message,

"deleted" => $is\_bot,

]);

}

function list($filter = []){

$this->db->select()->from($this->table\_messages);

$order = isset($filter["order"]) ? $filter["order"] : "ASC";

$this->db->order\_by("id", $order);

$offset = isset($filter["offset"]) ? $filter["offset"] : 0;

$limit = isset($filter["limit"]) ? $filter["limit"] : 0;

if($limit && $offset){

$this->db->limit($limit, $offset);

}else if($limit){

$this->db->limit($limit);

}

foreach($filter as $key => $val){

if(in\_array($key, ["offset", "limit", "order"])){

continue;

}

$this->db->where($key, $val);

}

$messages\_all = $this->db->get()->result();

$messages\_all = array\_map(function($item){

$item->id = (int) $item->id;

$item->timestamp = (int) $item->timestamp;

$item->target = (int) $item->target;

$item->deleted = (bool) (int) $item->deleted;

return $item;

}, $messages\_all);

return $messages\_all;

}

function delete($user, $messageId){

$this->db

->where("target", $user["id"])

->where("id", (int) $messageId)

->update($this->table\_messages, [

"deleted" => 1,

]);

}

}