МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
Факультет електроніки та інформаційних технологій

Кафедра комп'ютерних наук

Кваліфікаційна робота бакалавра

СИСТЕМА ПЕРСОНАЛІЗОВАНИХ ДОМЕННИХ ІМЕН

Здобувач освіти гр. ІН-82 Денис ЗАХЛЄБАЄВ

Науковий керівник, Олена ПРОЦЕНКО
кандидат ф.-м. наук, доцент

Завідувач кафедри Анатолій ДОВБИШ
доктор технічних наук,професор.

Суми 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук

Затверджую_____
Зав. кафедри Довбиш А.С.
"_____" _____2022 р.

ЗАВДАННЯ

до кваліфікаційної роботи бакалавра

Студента 4-го курсу, групи ІН.82 спеціальності 122 -Комп'ютерні науки, денної форми навчання Захлєбаєва Дениса Станіславовича.

Тема: Система персоналізованих доменних імен

Затверджена наказом по СумДУ
№_____ від _____2022 р.

Зміст пояснювальної записки: 1) аналітичний огляд літератури; 2) постановка завдання; 3) вибір методів рішення завдання; 4) практична реалізація поставленого завдання; 5) висновки

Дата видачі завдання "_____"_____2022 р.
Керівник випускної роботи _____ Олена ПРОЦЕНКО
Завдання прийняв до виконання _____ Денис ЗАХЛЄБАЄВ

РЕФЕРАТ

Записка: 41 стор., 21 рис., 1 табл., 1 додаток, 8 джерел.

Об’єкт дослідження — Система персоналізованих доменних імен

Мета роботи — створення поштового сервісу з індивідуальною системою імен електронної пошти без власного доменного імені

Результати — розроблено поштовий сервіс з індивідуальною системою імен електронної пошти без доменного імені. Створений сервіс зручний у користуванні, має швидку реєстрацію та не містить нічого зайвого. Розробка проводилась на базі мови програмування PHP та з використанням cPanel, RoundCube. У ході тестування проблем не виявлено.

PHP, CPANEL, ROUNDCUBE, E-MAIL, WEB-ПРОГРАМУВАННЯ

ЗМІСТ

ВСТУП

Світ пронизаний комунікацією за допомогою сучасних технологій. Відтепер надіслати листа звичайною поштою - це просто для атмосферності, для романтики, адже зв’язатися з людиною швидко та надійно можна багатьма способами: передзвонити, надіслати повідомлення за допомогою месенджеру або на електронну пошту.

Електронна пошта це не лише засіб для комунікації. Це універсальний засіб, що допомагає реєструватися та авторизуватися до інтернет-ресурсів, отримувати інформацію від сервісів що вас цікавлять. Під час переддипломної практики ми розглянемо саме комунікацію.

Бувають випадки, коли потрібно якомога швидше отримати поштову скриньку. Аналоги, які ми розглянемо під час практики, пропонують безліч сервісів на додачу, але також мають ускладнену реєстрацію.

І якщо казати про корпоративних юзерів, то отримання корпоративного домену - це складна процедура, яка крім очікування та перевірки наявності вимагає зайвих коштів.

А що як нам потрібна пошта з унікальним доменом? Без сервісів, які ми ніколи не планували використовувати? Просто поштова скринька з унікальною адресою. Саме вирішенням даного питання ми займемося безпосередньо під час переддипломної практики та написання дипломної роботи.

Метою роботи є розробка індивідуальної системи імен електронної пошти без власного доменного імені. Об’єктом дослідження є процеси створення, обробки та використання поштового сервісу. Предметом є розробка індивідуальної системи імен електронної пошти без власного доменного імені, що дозволяє реалізувати ці процеси.

Для досягнення мети були поставлено наступні завдання:

- Зробити огляд схожих за функціоналом сервісів
- Провести аналіз поштових сервісів та знайти шляхи для створення більш зручного та ефективного аналогу
- Розробка та тестування сервісу

1 АНАЛІЗ ВІДОМИХ РІШЕНЬ

Перш ніж розробити власний поштовий сервіс, потрібно ознайомитися з аналогами. В цьому переліку не буде GMail або Outlook, адже дані сервіси не дозволяють дуже швидко створити пошту, а тим паче на унікальному домені, безкоштовно. Окрім цього, ці сервіси також мають в собі безліч додаткових функцій, які не пов’язані з електронною поштою. Безумовно, це плюс, проте коли питання лише у поштовій скриньці, то користувачу легко буде заплутатися у безлічі функцій та сервісів.

1.1 Огляд існуючих рішень

1.1.1 ProtonMail

Першою електронною поштою у нашому переліку є ProtonMail [1]. Першопочатково даний сервіс було розроблено у 2013 році дослідниками CERN, а бета-тестування завершилося в 2016 році.

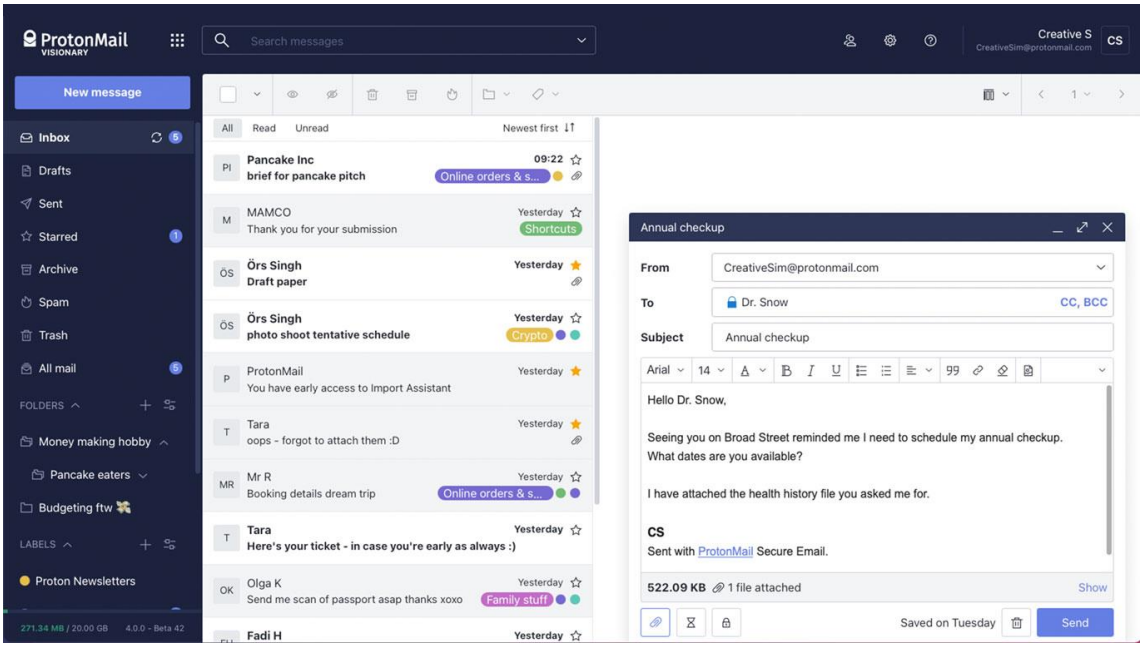

Рисунок 1.1 - Зовнішній вигляд поштового сервісу ProtonMail

З переваг сервісу варто виділити:

- наскрізне шифрування, що забезпечує безпеку листування (дані не можуть бути розшифровані посторонньою особою);
- відкритий вихідний код, що дозволяє pen-тестерам повністю перевірити продукт на безпечність та стресостійкість;
- Ephemeral messaging, що дозволяє зробити повідомлення зникаючим після прочитання.

Недоліками сервісу є:

- сховище 500 МБ на користувача, що змусить часто чистити сервіс від листів для підтримки життєздатності акаунту;
- цільова аудиторія використання сервісу, через що частіше за все листи, надіслані з ProtonMail, потрапляють до папки “Спам” та унеможливлюють безперешкодно комунікувати з потрібними контактами;
- відсутність можливості створення електронної пошти на унікальному домені.

1.1.2 TutaNota

Далі ми розглянемо сервіс TutaNota [2], що був розроблений в Німеччині в 2011 році. Він працює за популярною моделлю freemium, тобто користувачі вже мають базові функції, проте за розширення функціоналу потрібно доплатити.

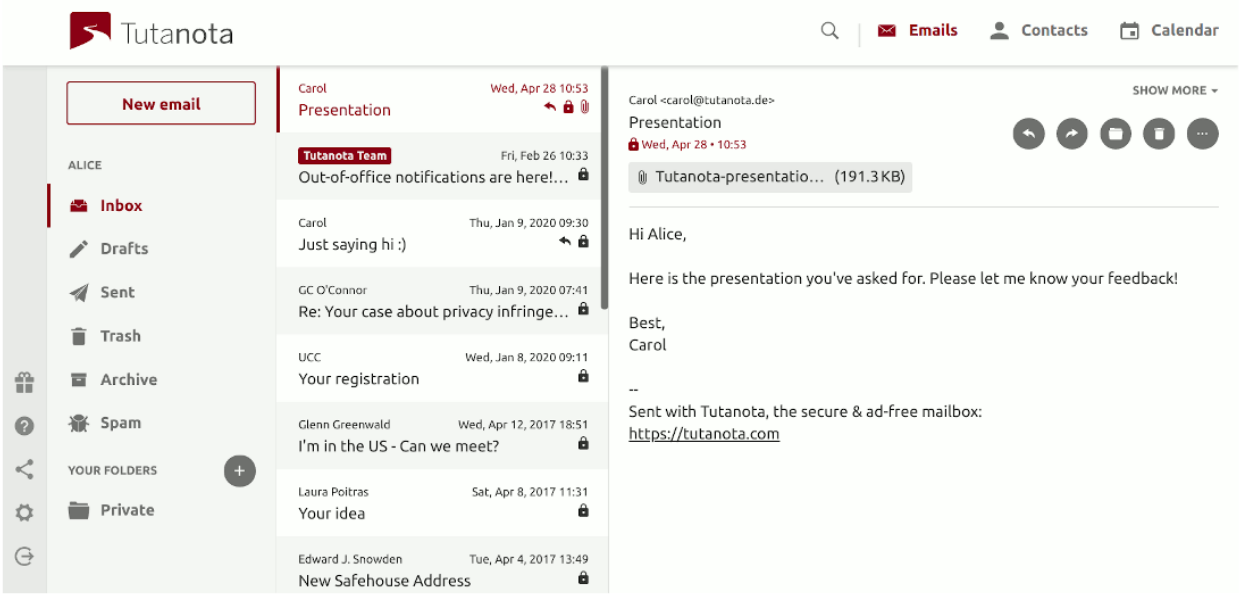

Рисунок 1.2 - Зовнішній вигляд поштового сервісу TutaNota

Переважно цей сервіс повторює функціонал та безпекову складову ProtonMail, але має такі переваги:

- розмір сховища для листів 1 Гігабайт, що вдвічі більше ніж у ProtonMail;
- можливість створювати електронну скриньку з власним доменним іменем;
- для користувачів інших поштових сервісів після надсилання повідомлення надається посилання на тимчасовий акаунт, що дозволить переглянути лист з TutaNota;

Основним недоліком сервісу можна виділити його маленьку спільноту, що може спричинити труднощі при технічних проблемах на сервісі, адже складно буде знайти відповідь на потрібне питання.

1.1.3 MailFence

Останнім сервісом, який ми розглянемо, буде MailFence [3], що на відміну від вже розглянутих аналогів найстарший з усіх, адже розроблений ще у 1999 році.

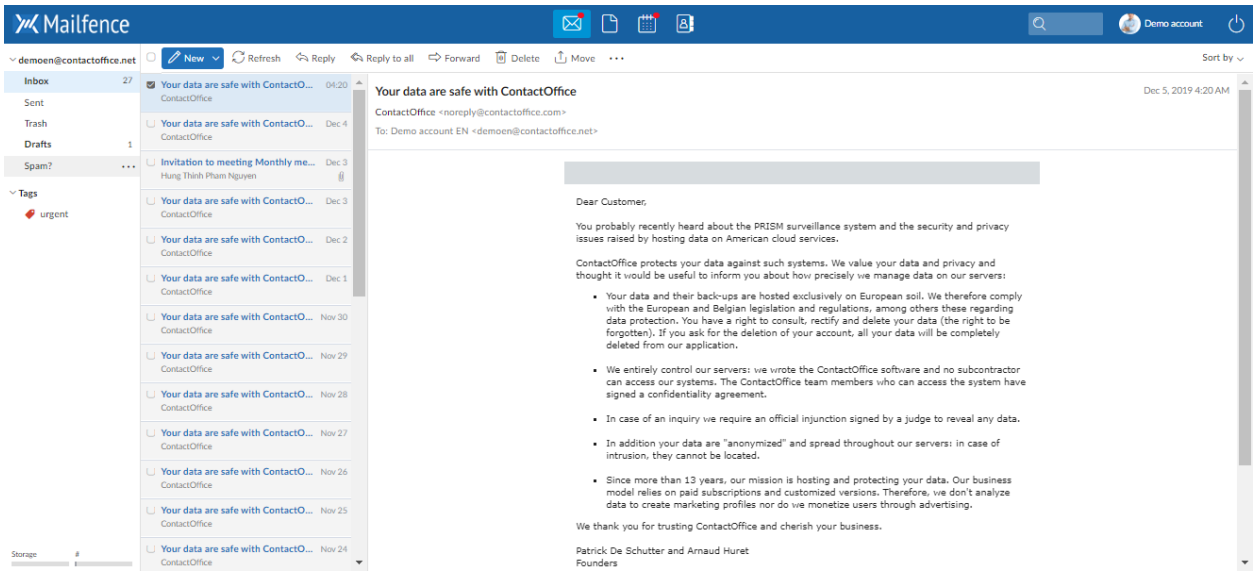

Рисунок 1.3 - Зовнішній вигляд поштового сервісу MailFence

Завдяки тому, що даний сервіс є найстаршим серед усіх і досі підтримується, він вже перевірений часом та користувачами. Також має шифрування, зручний інтерфейс. Проте варто виділити недоліки:

- сховище у 250 МБ для пошти, що менше за розглянуті аналоги;
- відсутність мобільної версії;
- закритий вихідний код, що унеможливлює перевірку на безпекову складову пентестерами.

Отже, було розглянуто кожне з рішень і тепер створимо порівняльну таблицю за трьома критеріями: розмір сховища, наявність шифрування, відкритість вихідного коду, кросплатформеність.

Критерій	ProtonMail	TutaNota	MailFence
Розмір сховища	512 МБ	1024 МБ	256 МБ (листи) + 256 МБ (документи)
Наявність шифрування	+	+	+
Вихідний код у відкритому доступі	+	+	-
Кросплатформеність	+	+	-

Таблиця 1.1 Порівняння поштових сервісів за певними критеріями

Як бачимо, розглянуті поштові сервіси об’єднує наявність шифрування та частково відкритість вихідного коду, кросплатформеність. Розмір сховища для листів у кожного з сервісів є різним, проте це мова про безкоштовні версії.

При підготовці до розробки та безпосередньо під час даного процесу варто буде врахувати усі переваги та недоліки сервісів, які ми розглянули.

1.2 Постановка задачі

У результаті огляду та аналізу схожих рішень, визначено мету даної роботи – проектування та розробка поштового сервісу з індивідуальною системою імен електронної пошти без власного доменного імені. Даний сервіс повинен мати просту реєстрацію, зручний інтерфейс, достатній функціонал, що притаманний типовому поштовому сервісу. Для здобуття цієї мети необхідно завершити наступні завдання:

1) Огляд та вибір мови програмування для веб-застосунку;
2) Вибір допоміжних інструментів для розробки поштового сервісу;
3) Проектування взаємодії користувача з сервісом;
4) Проектування та розробка поштового сервісу;
5) Тестування сторонніми користувачами;

2 ВИБІР ПРОГРАМНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1 Огляд та вибір мови програмування для веб-застосунку

Розглянувши аналоги, потрібно підготуватися до розробки сервісу. Перш за все, потрібно вибрати мову програмування, з якою ми реалізуємо back-end власного сервісу.

- Першою мовою програмування, яку ми розглянемо, є Python. Дана мова програмування користується широкою популярністю та універсальністю, адже вона має застосування у різних сферах IT: автоматизація та роботизація, Data Science, web-програмування тощо. Python веде за собою чималу кількість розробників, відповідно на базі цієї мови ми маємо такі популярні фреймворки, як Django, Flask, FastAPI та aiohttp, що спрощують розробку веб-застосунків.
- Не менш популярною мовою програмування є JavaScript. Не дивлячись на всі дивні речі, що ми можемо спостерігати у цій мові, вона розвивається з року в рік. Якщо раніше ми могли говорити про JavaScript як про мову для front-end складової веб-застосунку, то відносно нещодавно вона поповнилася бібліотеками та фреймворками для back-end розробки. Наприклад, найяскравішим та найпопулярнішим фреймворком на JS є Node.JS, що також є складовою широко використовуваного стеку MERN (MongoDB, Express, React, Node).
- Перелік завершимо мовою програмування PHP. Історія даної мови програмування цікава тим, що вона не розпочинала свій шлях як мова програмування. Проте наразі майже більшість сервісів використовує саме PHP, а перелік бібліотек та фреймворків стає все ширшим. PHP навіть без використання фреймворків дозволяє розробити веб-застосунок, що працюватиме довгий час при мінімальній підтримці. До речі, варто також згадати про можливість підтримки веб-застосунків,

написаних цією мовою. З цим проблем майже не виникне, адже ринок перенасичений розробниками на PHP через її простоту та популярність.

Виходячи з вищезазначеного, проаналізованих мов програмування та власних сил, мною було обрано для розробки мову програмування PHP. Вона дозволить не лише розробити back-end мого сервісу, а також поєднати його з front-end складовою, тобто в даному випадку не доведеться витрачати зайві сили на створення зв'язку між back-end та front-end частинами сайту.

2.2 Вибір допоміжних інструментів для розробки поштового сервісу

Кожен продукт не може обійтись без MVP, мінімально життєздатного продукту (англ. Minimal Viable Product) - версії продукту, яка демонструє базовий або певний функціонал для кінцевого користувача [4]. MVP не є фінальною версією продукту, це так би мовити життєздатний ескіз, який має перспективи та плани для розвитку в майбутньому.

Рисунок 2.1 - Шлях від мінімально життєздатного продукту до більш складного

Для того, щоб продемонструвати функціонал мого продукту, ми також створимо MVP. Варто згадати, що для даної версії продукту докладається мінімум зусиль, тому для їх мінімізації оберемо та скористаємося допоміжними засобами.

В основному допоміжні засоби будуть для frontend частини продукту, проте за допомогою них закриємо деякі питання у backend частині.

За основу ми візьмемо панель керування веб-хостингом cPanel [5]. Дана панель управління є другою за популярністю системою для хостингу та має безліч плагінів та аддонів для розширення функціоналу хостингу. cPanel розроблена на Perl та без проблем розгортається і працює на ОС Linux.

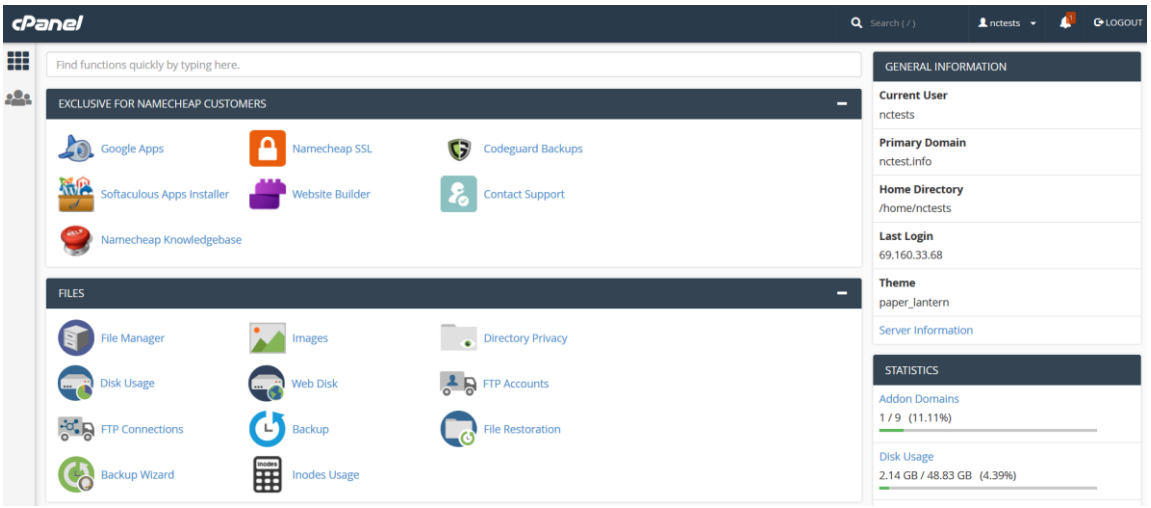

Рисунок 2.2 - Зовнішній вигляд cPanel

Для нашого поштового сервісу нам знадобиться відслідковувати навантаження на сервер, кількість зайнятої користувачами пам’яті, які користувачі є на сервісі. Безпосередньо, для розробки поштового сервісу ми скористаємося webMail та оберемо між Roundcube, Horde. Розглянемо обидва ці елементи окремо.

webMail є складовою cPanel та дозволяє забезпечити авторизацію до поштової скриньки, доступ до налаштувань скриньки, самої скриньки. Зазвичай webMail використовується у комбінації з клієнтами Roundcube або Horde. Розглянемо кожен з цих клієнтів окремо.

Roundcube є клієнтом для роботи з електронною поштою, що розроблений на мові PHP з використанням JS, HTML, CSS та технології AJAX (Asynchronous Javascript and XML) [6]. Даний клієнт є універсальним, тобто

має можливість для встановлення на більшість серверів, конфігурацію з популярними СУБД.

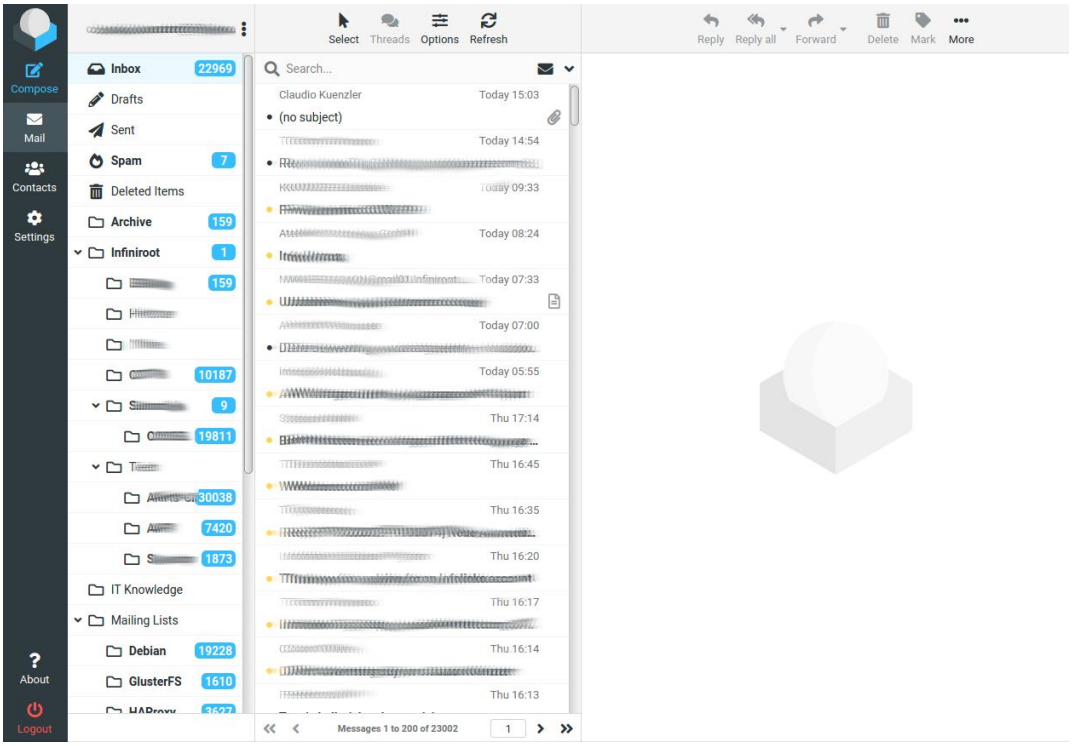

Рисунок 2.3 - Зовнішній вигляд клієнту Roundcube

Roundcube випускається під ліцензією GPL та має відкритий вихідний код, що забезпечує вільне користування будь-ким. Перекладений більшістю мов світу та має зручне налаштування інтерфейсу, що розширює цільову аудиторію користувачів.

Тепер розглянемо Horde. Даний клієнт також написано на PHP, проте через меншу популярність не так широко використовується в розробці [7].

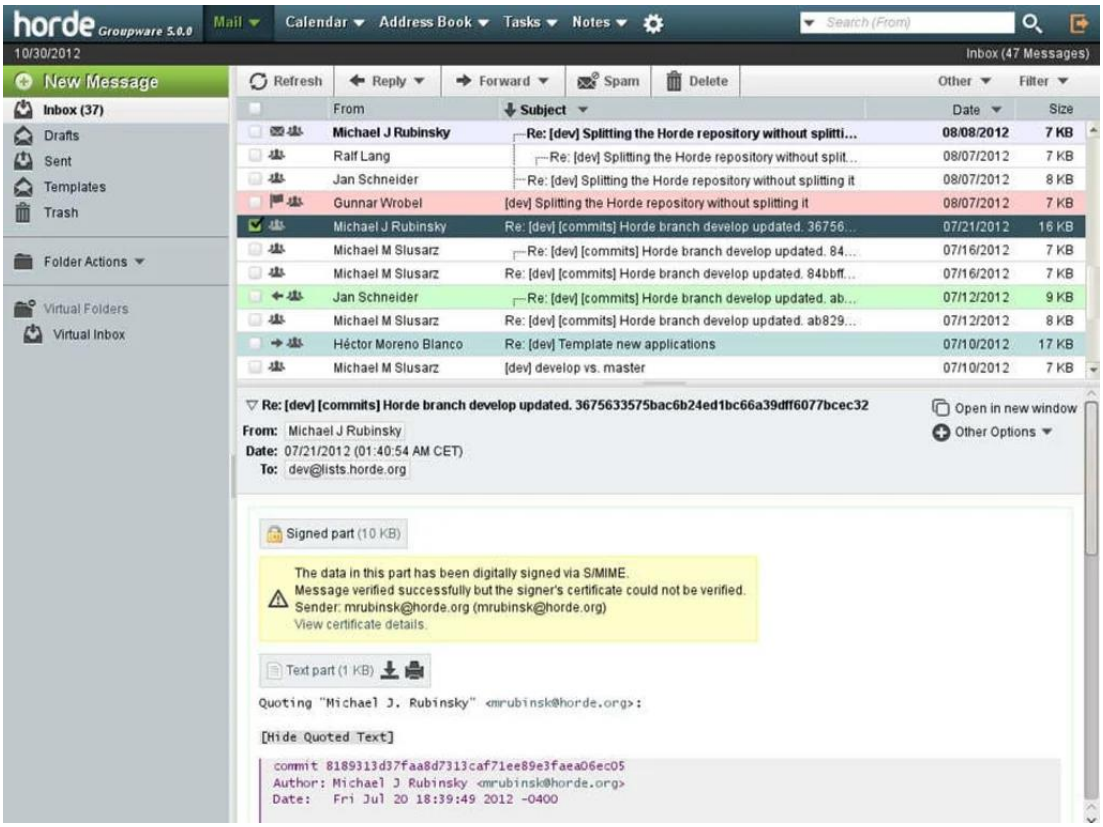

Рисунок 2.4 - Зовнішній вигляд Horde

Також не вдалося знайти інформацію про підтримувані СУБД клієнтом Horde, а інтерфейс сайту проекту та безпосередньо сервісу є незручними.

Отже, провівши поверхневий аналіз, було обрано клієнт RoundCube через його популярність та більшу спільноту розробників, що з ним працюють.

2.3 Проектування поштового сервісу

Після огляду, аналізу та обрання усіх компонентів для розробки, ми можемо перейти до проектування взаємодії користувач-сервіс та безпосередньо самого поштового сервісу.

Розглянемо першу взаємодію користувача із сервісом.

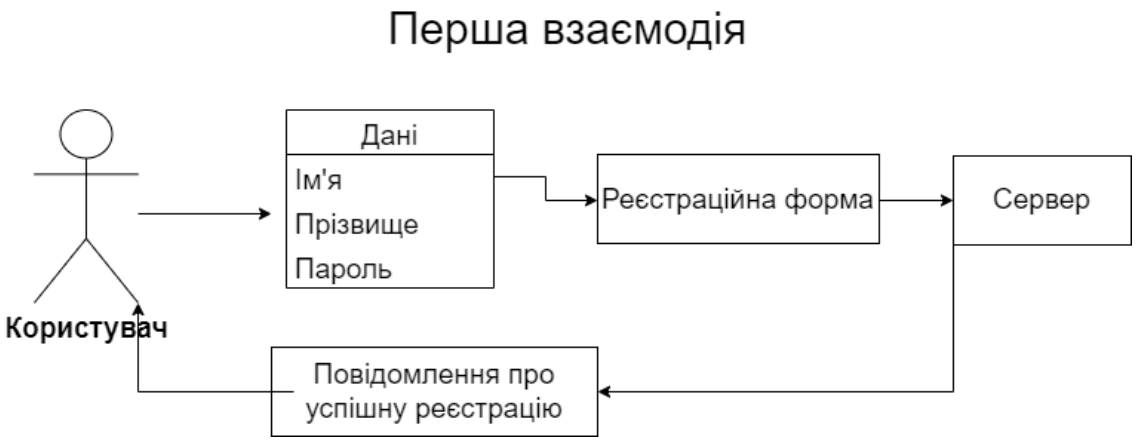

Рисунок 2.5 - Перша взаємодія користувача із сервісом

В даному випадку, користувач відвідує наш сервіс та надає мінімальний набір даних: ім’я, прізвище та пароль. Заповнивши реєстраційну форму, дані передаються на сервер, зберігаються у базі даних та користувач отримує повідомлення про успішну реєстрацію.

Розглянемо тепер повторну взаємодію, яка відбувається після реєстрації користувача в системі.

Рисунок 2.6 - Повторна взаємодія користувача з сервісом

Як ми бачимо, користувачу достатньо ввести власний e-mail та пароль, що були створені до цього і він потрапить до своєї поштової скриньки. Варто зазначити, що e-mail виглядатиме таким чином: firstname@lastname.2.email

Візьмемо, наприклад, користувача John Doe. Його e-mail виглядатиме як john@doe.2.email. Унікальне доменне ім’я, ще й безкоштовно - приємно та зручно, чи не так?

Розглянемо тепер те, заради чого створюється поштовий сервіс - процес надсилання листа.

Рисунок 2.7 - Процес “Надсилання листа”

Користувач формує лист, в якому вказує відправника, отримувача, тему листа та його тіло. За бажанням він також може вкласти файл та додати підпис до листа, проте це не є обов’язковими процедурами. Лист йде на сервер, з якого він зберігається у базі даних поштового сервісу та надсилається повідомлення про успішне надсилання. З нашого серверу він також транспортується до серверу отримувача та доходить до адресату.

Варто зазначити, що у даному випадку використовуватиметься звичайне TLS-шифрування, проте в перспективі можливо замінити та покращити тип шифрування, а також разом з цим додати функції, які відрізнятимуть наш сервіс від інших.

Тепер розглянемо проектування безпосередньо самого сервісу. Враховуючи, що ми використовуємо cPanel, webMail та RoundCube, проектування та розробка будуть дещо спрощеними.

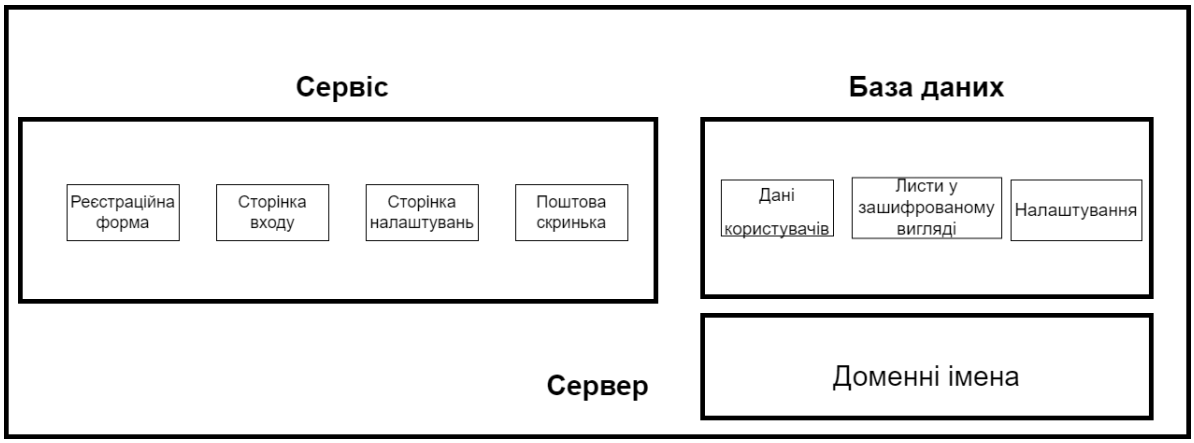

Рисунок 2.8 - Спрощена діаграма поштового сервісу

Ми маємо сервер, на якому розташовані безпосередньо сам сервіс, база даних та доменні імена кожного користувача. У самому сервісі ми маємо такі основні сторінки, як реєстраційна форма, сторінка входу (для авторизованих користувачів), сторінку налаштувань та поштову скриньку. Кожна з цих сторінок має ще декілька додаткових сторінок, адже cPanel + Roundcube надають чимало функціоналу нашому сервісу.

Також ми маємо базу даних, проте і вона спроектована з допомогою інструментів, які ми обрали. В ній міститимуться дані користувачів, листи (у зашифрованому вигляді) та налаштування для кожного користувача та безпосередньо для всього сервісу.

Отже, ми спроектували взаємодію користувач-сервіс, складові самого сервісу та його помічники (база даних, сховище доменних імен). Тепер ми можемо приступити до розробки та безпосередньо тестування нашого сервісу.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Розробка веб-застосунку

Переходимо до розробки нашого застосунку. У минулому розділі було обрано мову програмування PHP та такі програмні засоби, як CPanel та Roundcube.

Починаємо з функції для створення поштової скриньки, яка буде опрацьовуватися безпосередньо на головній сторінці застосунку.

При розробці цієї функції нам треба врахувати такі моменти:

1. Людина може випадково увімкнути capslock під час написання прізвища та імені, які сформують адресу електронної скриньки. В подальшому це може вплинути на отримання нею доступу до скриньки, адже до бази даних було передано ім’я JoHN Doe, а вона наступного разу вводитиме john doe.
2. Валідація e-mail. Людина може додавати спеціальні символи, які є неприйнятними для адреси поштової скриньки. Звичайно, її варіант можна допустити і не валідувати, проте в подальшому поштовий сервер може просто нестабільно працювати через зайві символи в адресі.

Враховуючи вищезазначені моменти, функція виглядатиме таким чином:

```
$fname = strtolower($_POST['fname']);
$lname = strtolower($_POST['lname']);
$passw = $_POST['passw'];

$email = $fname."@".$lname.".2.email";

function validateEmail($email) {
   if(filter_var($email, FILTER_VALIDATE_EMAIL)) {
```

```
            $a = null;
        }
        else {
            $a = 1;
        }
        return $a;
    }

    if(validateEmail($email)){
        echo "{$email}: Not a valid email"."<br>";
        echo "Your email can not be created. Do not use spaces or special signs.<br> Try one more time<br>";
        goto tryer;
    }else{
        echo "{$email}: A valid email and will be created"."<br>";
    }
```

Тобто, ми переводимо ім’я та прізвище у lowercase (всі літери – маленькі), щоб не допустити випадковий caps lock. Для недопущення зайвих символів, ми використовуємо validateEmail – вбудована функція PHP, яка відповідає за перевірку вхідної адреси електронної скриньки, яку надає користувач.

Тепер ми переходимо до інтеграції CPanel у наш застосунок. Для його інтеграції ми скористаємося API – опис способів, за допомогою яких один застосунок може взаємодіяти з іншим.

Ми створюємо клас cPanelAPI, в якому міститимуться такі складові:

1. Конструктор – прийматиме та зберігатиме ім’я користувача та пароль, а також адресу та порт, за якими можна потрапити до cPanel.

2. Функція для побудови посилання на сторінку cPanel – потрібна для того, щоб користувач коректно потрапив до потрібної панелі.
3. Функція для побудови шляху до поштового сервісу. Потрібна для того, щоб застосунок коректно зміг «спілкуватися» з поштовим сервером.
4. Функція для початку роботи cPanel. Відповідає за те, щоб cPanel розпочав роботу коректно.

Після визначеного переліку складових класу, який відповідатиме за cPanel, було розроблено такий код:

```
class cPanelApi {

    public function __construct($cpanelUrl, $cpaneluser, $cpanelPwd,
$cpanelPort = '2083') {
        $this->cPanelUser = $cpaneluser;
        $this->cPanelPwd = $cpanelPwd;
        $this->cPanelUrl = $cpanelUrl;
        $this->cPanelPort = $cpanelPort;
    }

    /////////////// CPANEL /////////////////

    public function folt($subdomain, $folder = '') {

    if($folder == '') {
        $folderdir = '%2Fpublic_html%2F'.$subdomain;
    } else {
        $folderdir = '%2Fpublic_html%2F'.$folder;
    }
```

```
        $func = "https://$this->cPanelUrl:$this->cPanelPort/execute/SubDomain/addsubdomain?domain=$subdomain&rootdomain=$this->cPanelUrl&dir=$folderdir&disallowdot=0";

        return $this->exe_cpanel($func);

        }

        public function another_email($fname, $passw, $lname) {

            $func = "https://uashared16.twinservers.net:2083/cpsess9772145410/execute/Email/add_pop?email=$fname&password=$passw&domain=$lname.2.email&quota=250&send_welcome_email=1";

            return $this->exe_cpanel($func);

            }

        /////////////// CPANEL /////////////////

        private function exe_cpanel($func = '') {
            $query = $func;
            $curl = curl_init();
            curl_setopt($curl, CURLOPT_SSL_VERIFYPEER,0);
            curl_setopt($curl, CURLOPT_SSL_VERIFYHOST,0);
            curl_setopt($curl, CURLOPT_HEADER,0);
            curl_setopt($curl, CURLOPT_RETURNTRANSFER,1);
```

```
        $header[0] = "Authorization: Basic " . base64_encode($this-
>cPanelUser.":".$this->cPanelPwd) . "\n\r";
        curl_setopt($curl, CURLOPT_HTTPHEADER, $header);
        curl_setopt($curl, CURLOPT_URL, $query);
        $result = curl_exec($curl);
        if ($result == false) {
            error_log("curl_exec threw error \"" . curl_error($curl) . "\" for
$query");
        }
        curl_close($curl);
        return $result;
    }

}
```

Інші функції, які присутні у нашому застосунку, можна переглянути у додатку.

Переходимо до розробки front-end частини. Так як cPanel та Roundcube вже мають графічний інтерфейс, то нам залишається розробити головну сторінку.

Починаємо розробку нашої сторінки з head-блоку. В ньому буде розміщено яке саме кодування потрібно для сторінки, як саме відображати сторінку (у мобільному або desktop-форматі), як називається сторінка, а також за допомогою цього блоку ми підключимо css-файли.

Head-блок виглядатиме таким чином:

```
<head>
    <meta charset="utf-8">
```

```
            <meta name="viewport" content="width=device-width, initial-
        scale=1.0, shrink-to-fit=no">
            <title>2.email.alpha</title>
            <link rel="stylesheet"
        href="assets/bootstrap/css/bootstrap.min.css">
            <link rel="stylesheet" href="assets/css/Registration-Form-with-
        Photo.css">
            <link rel="stylesheet" href="assets/css/styles.css">
        </head>
```

Розробимо вітальне повідомлення, яке буде зустрічати нових користувачів системи або пропонувати увійти вже існуючим. Варто зазначити, що під час розробки front-end частини буде використовуватись Bootstrap – набір інструментів, що містить шаблони для елементів сайтів, на кшталт кнопок, форм, типографіки, навігації тощо.

Так виглядатиме розмітка вітального повідомлення:

```
<div class="container" style="margin-top: 20px;">
    <div class="card">
        <div class="card-body">
            <h4 class="card-title">You can take part in closed Pre Alpha
testing of 2Email</h4>
            <h6 class="text-muted card-subtitle mb-2">We strongly
recommend not to save any important data in alpha,
                because all accounts will be deleted before the next version will
be uploaded<br></h6>
            <p class="card-text" style="margin-bottom: 8px;margin-top:
8px;">You can fill in the information to
                complete your registration. <br>Something may not work
properly, we apologize for that.<br></p>
            <h3>2Email is the ultimate email service for self-branding</h3>
```

```
                    <h5>As well you can <a
href="http://mail.2.email:2096"><em><span style="text-decoration:
underline;">log in
                                here</span></em></a></h5>
                    <h6>Or in case this page will not be visible during some time, you
can try to visit <a

href="http://mail.2.email:2096"><strong>mail.2.email</strong></a> to log
in</h6>
                </div>
            </div>
        </div>
```

Як можна помітити, окрім стандартних засобів HTML та CSS, було використано card-body, що притаманний Bootstrap.

Також потрібно розробити розмітку форми, яка буде реєструвати поштову скриньку. Тут ми також використаємо Bootstrap, але окрім цього ми також в цьому шматку коду приєднаємо наш застосунок до PHP-коду. Так виглядатиме форма реєстрації поштової скриньки:

```
    <div class="container">
        <section class="register-photo">
            <div class="form-container">
                <form method="post" action="index.php">
                    <h2 class="text-center"><strong>Create</strong> an
account.</h2>
                    <div
                        class="d-sm-flex d-md-flex d-xxl-flex align-items-sm-center
justify-content-md-center justify-content-xxl-start mb-3">
                        <input class="form-control" type="text" name="fname"
placeholder="First Name">
```

```
                        <span class="d-xxl-flex align-items-xxl-center" style="font-size: 24px;height: 40px;">@</span>
                        <input class="form-control" type="text" name="lname" placeholder="First Name">
                        <span class="d-xxl-flex align-items-xxl-center"
                            style="font-size: 24px;height: 40px;">.2.email</span>
                    </div>
                    <div class="mb-3"><input class="form-control" type="password" name="passw" placeholder="Password">
                    </div>
                    <div class="mb-3">
                        <div class="form-check"><label class="form-check-label"><input class="form-check-input"
                                type="checkbox">I agree to share my opinion after, just kidding</label></div>
                    </div>
                    <div class="mb-3"><button class="btn btn-primary d-block w-100" type="submit">Sign Up</button></div>
                    <a class="already" href="#">You already have an account? Login here.</a>
                </form>
            </div>
        </section>
    </div>
```

Повну розмітку можна переглянути в додатку до дипломної роботи.

3.2 Тестування проекту

Після розробки сервісу, вихідний код якого можна буде знайти у додатку, та його розгортання на сервері, ми можемо перейти до його

тестування. Переходячи на сторінку, нас зустрічає проста форма для реєстрації, в якій повідомляється про альфа-тестування даного сервісу. Це нормально, адже перед тим, як сервіс буде повністю придатний для користування, потрібно перевірити його на працездатність та наявність критичних помилок.

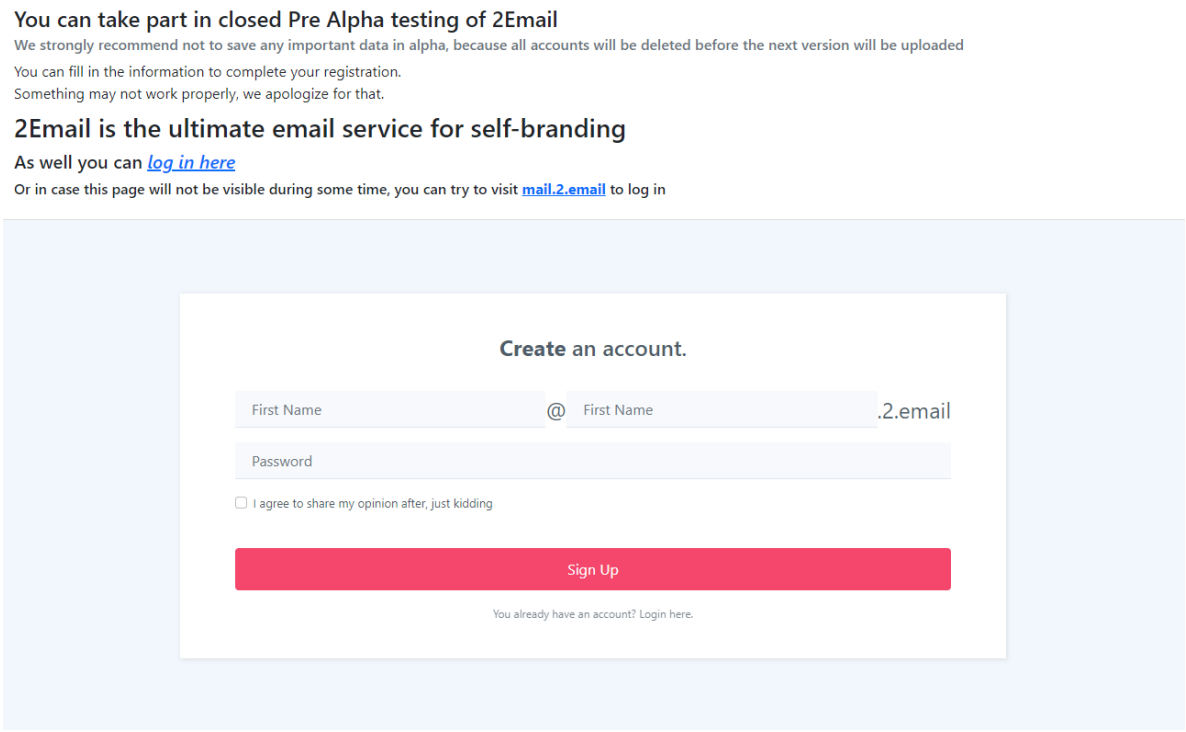

Рисунок 3.1 - Реєстраційна сторінка сервісу

Ми вводимо дані, що вимагаються від нас, а саме: прізвище, ім’я та пароль. Тобто, на відміну від більшості сервісів, мій не вимагає ні дати народження, ні інших персональних даних.

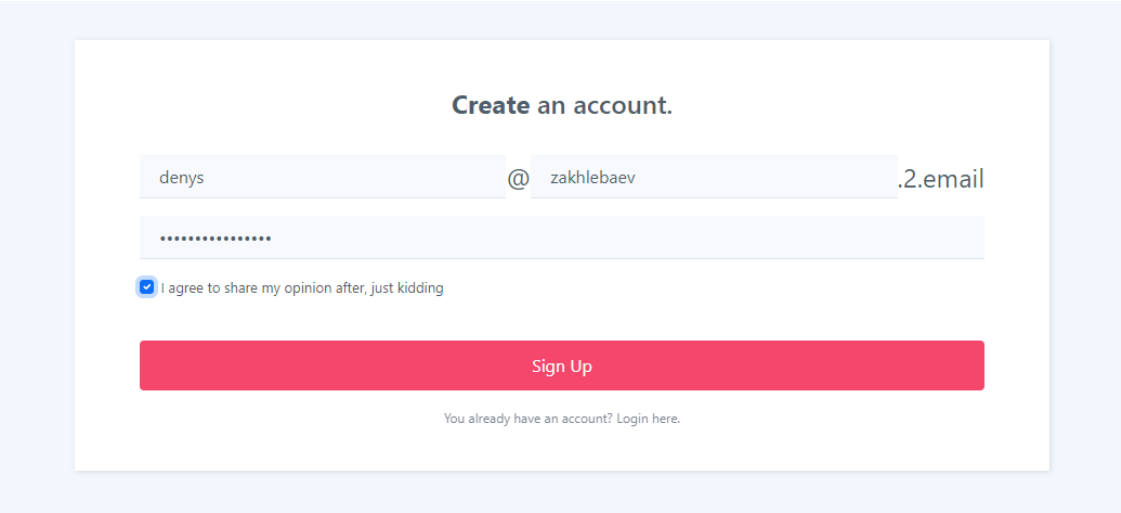

Рисунок 3.2 - Реєстраційна форма з заповненими даними

Натискаємо на Sign Up та отримуємо повідомлення про те, що нашу поштову скриньку зареєстровано.

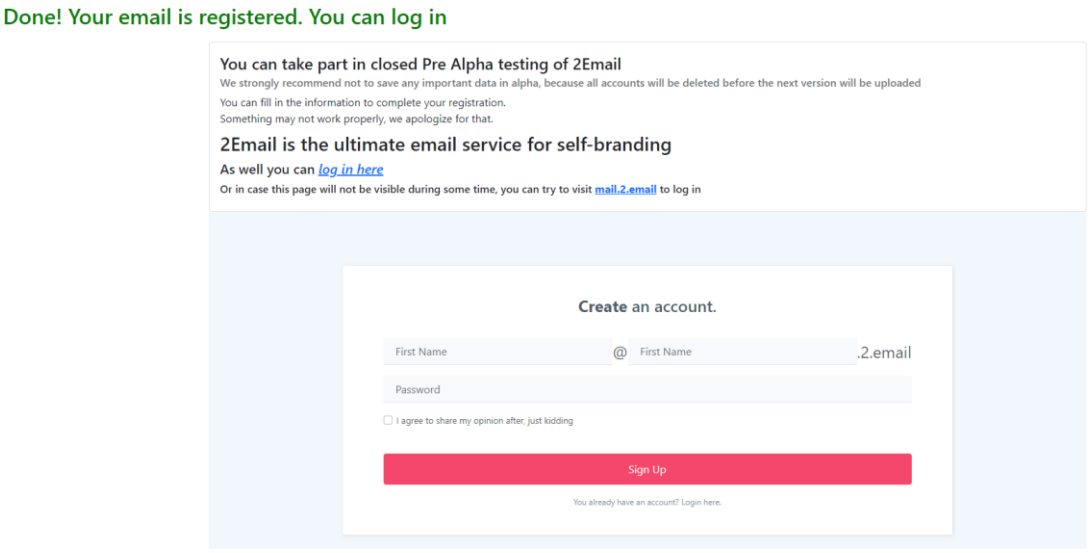

Рисунок 3.3 - Повідомлення про успішну реєстрацію

Тепер ми можемо увійти в нашу поштову скриньку та перевірити працездатність поштового сервісу. Для цього ми натискаємо на напис “log in here” та потрапляємо на сторінку для введення логіну та паролю.

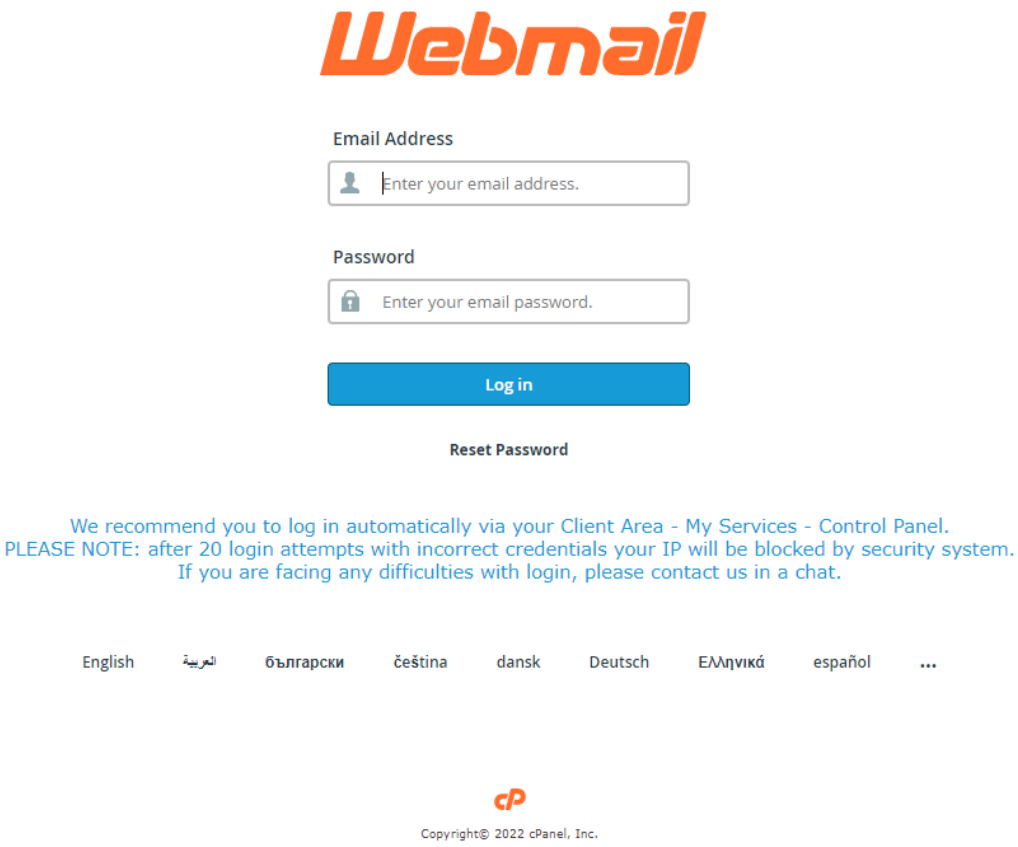

Рисунок 3.4 - Сторінка для авторизації до нашого поштового сервісу

Після введення логіну та паролю і натискання на кнопку “Log in”, ми потрапляємо на сторінку налаштувань нашої скриньки.

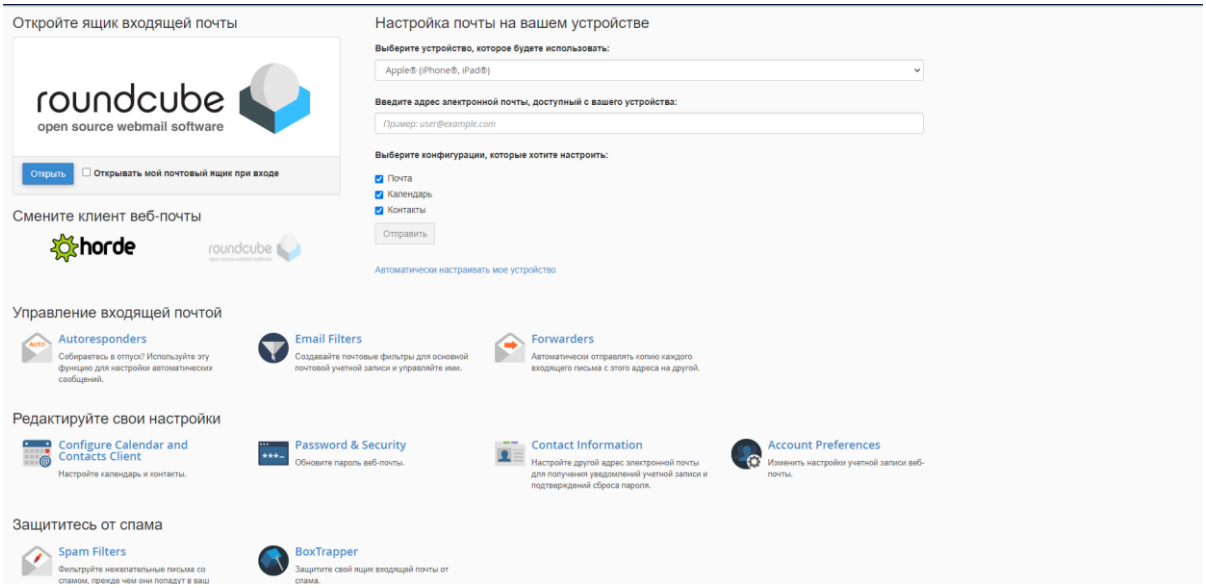

Рисунок 3.5 - Сторінка налаштувань нашої скриньки

За бажанням, можна провести додаткові налаштування скриньки: автоматичні відповіді, захист від спаму, додавання контактної інформації тощо. Проте наразі ціллю є перевірка працездатності нашого сервісу та наявності можливості надіслати листа на іншу пошту.

Переходимо безпосередньо до нашої поштової скриньки. Нас зустрічає сторінка з сучасним та зручним дизайном, що спрощує роботу для будь-якої категорії користувачів.

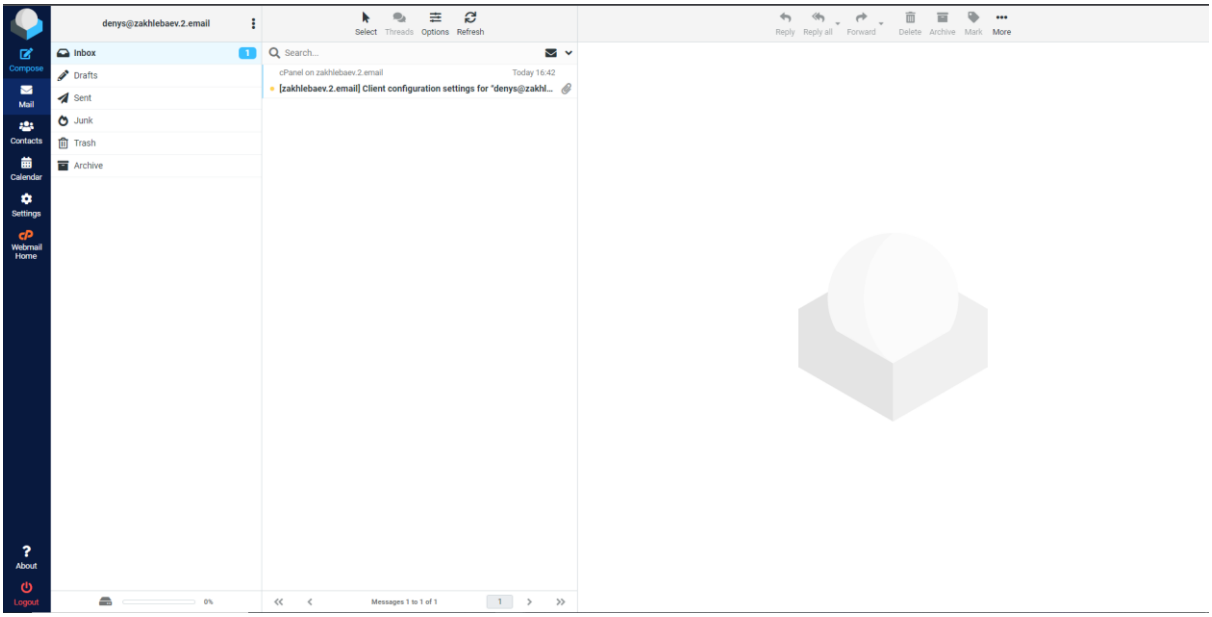

Рисунок 3.6 - Поштова скринька новоствореного користувача

Перевіримо, чи вдасться нам надіслати лист іншому користувачу, натиснувши на кнопку “Compose” в правому верхньому кутку.

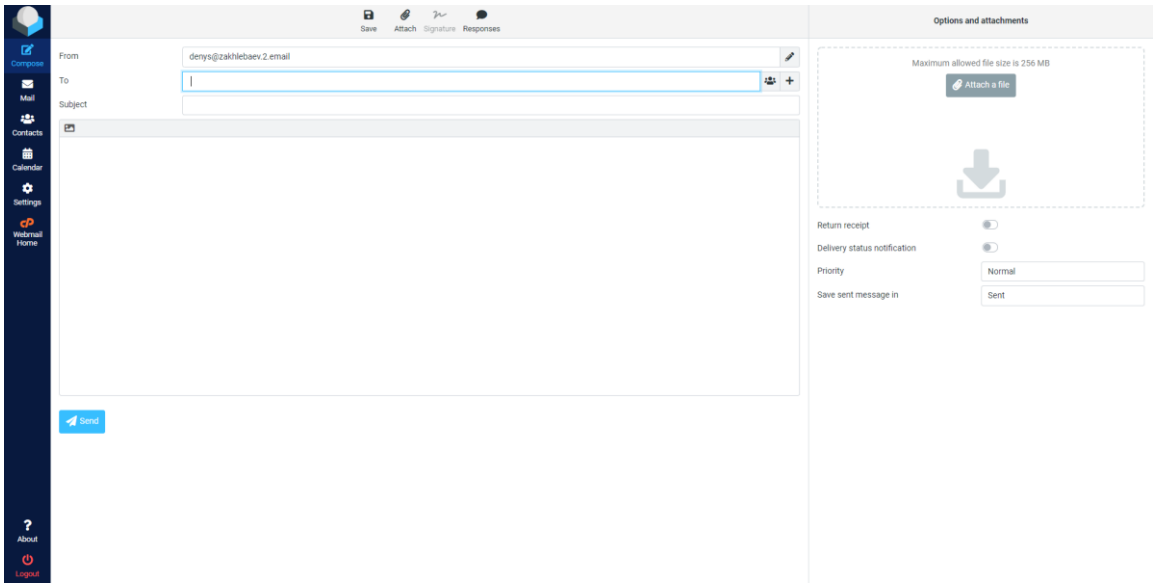

Рисунок 3.7 - Сторінка створення нового листа

Як ми бачимо, наш сервіс має достатній функціонал для повноцінної роботи під час створення листа, а саме: вказання відправника, отримувача, теми листа, додавання файлу або підпису тощо.

Створюємо наш майбутній лист та для надсилання натискаємо на кнопку “Send”. Отримуємо сповіщення, що лист успішно надіслано.

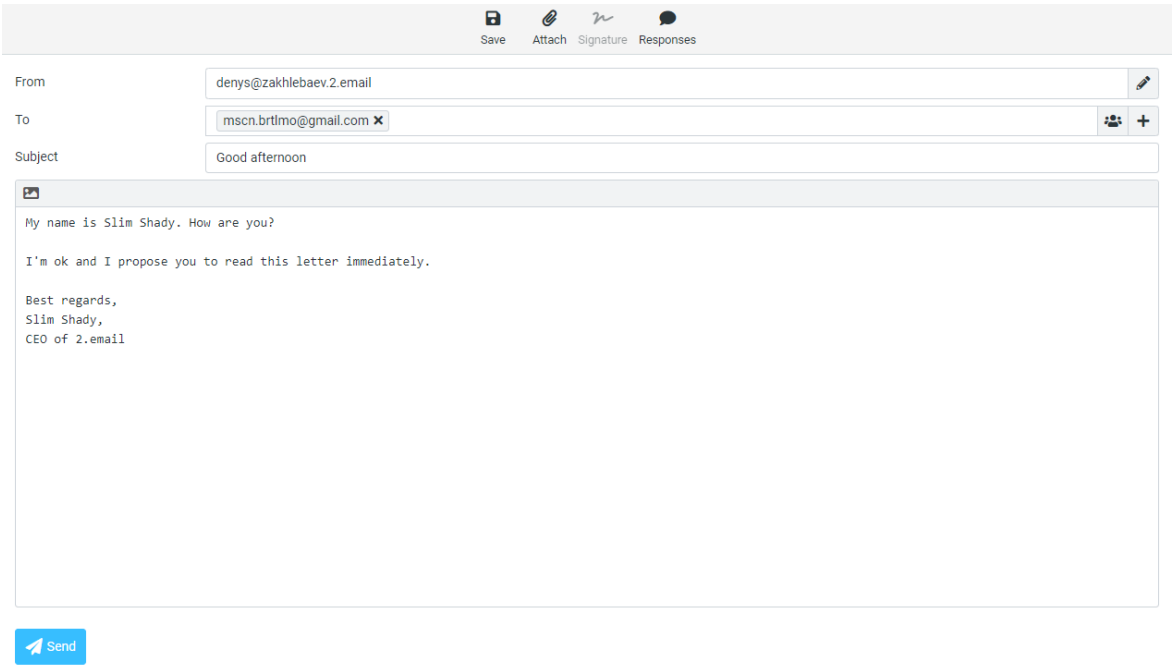

Рисунок 3.8 - Заповнені поля майбутнього листа

Тепер перевіримо на нашій поштовій скриньці, чи надійшов лист за допомогою нашого сервісу. Бачимо, що лист надійшов успішно, без потрапляння до папки “Спам” та повністю.

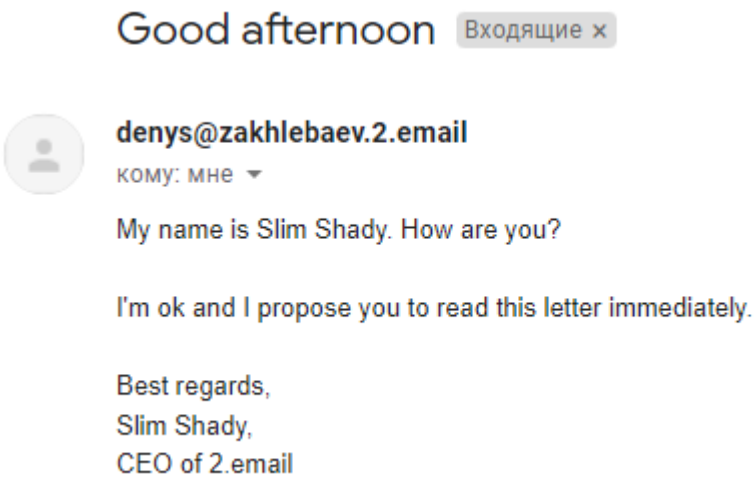

Рисунок 3.9 - Лист, отриманий з сервісу 2.email

Перевіряємо дані листа, де вказано від кого лист, до кого прийшов, о котрій годині та який спосіб шифрування. Бачимо, що наразі маємо стандартне TLS-шифрування, проте в перспективі змінити метод шифрування та зробити щось на кшталт як ProtonMail.

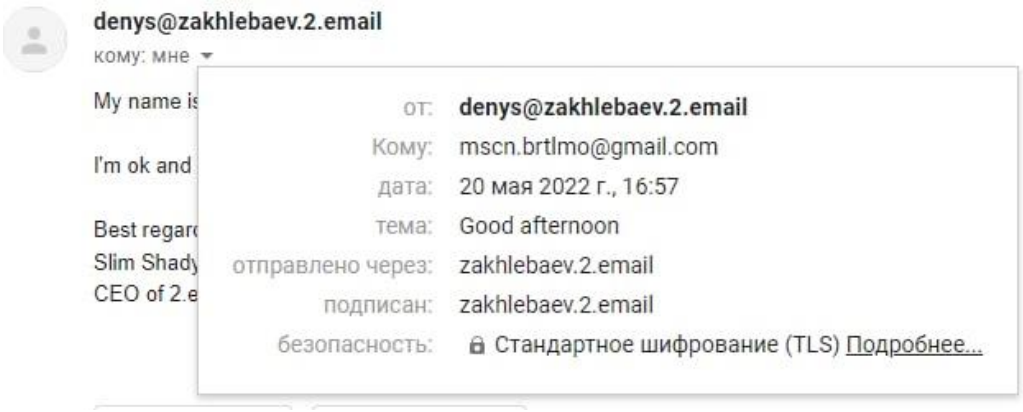

Рисунок - 3.10 Дані про лист

Виходячи з тестування, робимо висновок що сервіс працює справно, без зайвих проблем і ми можемо визначити перспективи та плани його розвитку.

3.3 Визначення планів та перспектив на майбутнє

Отже, основними планами та перспективами на майбутнє для поштового сервісу є:

1. Обрання більш досконалого методу шифрування наших листів.
2. Створення аналогічної ProtonMail функції зникаючих листів.
3. Створення аналогічної TutaNota функції листів, які можна переглянути з наданим тимчасовим акаунтом.
4. Розширення сховища листів для кожного з користувачів.
5. Захист від DDoS-атак сервісу (напр. за допомогою Cloudflare).
6. Розробка власного аналогу cPanel та Roundcube, що дозволить залишити лише найпотрібніше у нашому сервісі.
7. Перехід на модель freemium.

Ми вже маємо мінімально життєздатний продукт, проте ці плани здатні покращити його працездатність та закріпити позиції на ринку за допомогою вищезгаданих планів та перспектив, а також переходу на модель freemium.

ВИСНОВКИ

В ході виконання роботи були вирішені наступні задачі:

- розглянуті та проаналізовані аналоги поштових сервісів;
- проведено аналіз мов програмування та допоміжних інструментів;
- спроектовано взаємодію користувач-сервіс, безпосередньо сам сервіс;
- розроблено поштовий сервіс;
- протестовано сторонніми користувачами поштовий сервіс 2.email.

Результатом виконання роботи є розроблений поштовий сервіс “2.email” з індивідуальною системою імен електронної пошти без власного доменного імені, що буде гарним помічником для тих, кому потрібна лише пошта з унікальним іменем і нічого зайвого.

СПИСОК ЛІТЕРАТУРИ

1. ProtonMail[Електронний ресурс] – Режим доступу: https://protonmail.com/
2. TutaNota [Електронний ресурс] – Режим доступу: https://tutanota.com/
3. MailFence [Електронний ресурс] – Режим доступу: https://mailfence.com/
4. MVP: що це таке і як працює? [Електронний ресурс] – Режим доступу: https://habr.com/en/company/productstar/blog/508892/
5. What is cPanel? [Електронний ресурс] – Режим доступу : https://www.wpbeginner.com/glossary/cpanel/
6. Roundcube Official website [Електронний ресурс] – Режим доступу : https://roundcube.net/
7. Horde Official website [Електронний ресурс] – Режим доступу: https://www.horde.org/apps/webmail
8. Налаштування поштового серверу “вдома” [Електронний ресурс] – Режим доступу : https://habr.com/en/post/539736/
9. Bootstrap. Official documentation [Електронний ресурс] – Режим доступу: https://getbootstrap.com/docs/4.1/getting-started/introduction/
10. Red Hat. What is REST API? [Електронний ресурс] – Режим доступу: https://www.redhat.com/en/topics/api/what-is-a-rest-api
11. Validate E-mail. PHP [Електронний ресурс] – Режим доступу: https://www.php.net/manual/en/filter.examples.validation.php
12. Guide to the Live API system - CPanel [Електронний ресурс] – Режим доступу: https://api.docs.cpanel.net/guides/guide-to-the-liveapi-system/
13. How to use PHP in HTML [Електронний ресурс] – Режим доступу: https://code.tutsplus.com/tutorials/how-to-use-php-in-html-code--cms-34378
14. Post Office Protocol [Електронний ресурс] – Режим доступу: https://www.javatpoint.com/pop-protocol
15. Introduction to cURL. PHP [Електронний ресурс] – Режим доступу: https://www.php.net/manual/ru/intro.curl.php

ДОДАТОК

Вихідний код сервісу

```
<?php

ini_set('display_errors', 1);
ini_set('display_startup_errors', 1);
error_reporting(E_ALL);

/*if($_COOKIE['go'] !== "pop"){
    exit();
}
*/

if(isset($_POST['fname'])||isset($_POST['lname'])||isse
t($_POST['passw'])){

    $fname = strtolower($_POST['fname']);
    $lname = strtolower($_POST['lname']);
    $passw = $_POST['passw'];

    $email = $fname."@".$lname.".2.email";

    function validateEmail($email) {
        if(filter_var($email, FILTER_VALIDATE_EMAIL)) {
            $a = null;
        }
        else {
            $a = 1;
```

```
        }
        return $a;
    }

    if(validateEmail($email)){
        echo "{$email}: Not a valid email"."<br>";
        echo "Your email can not be created. Do not use
spaces or special signs.<br> Try one more time<br>";
        goto tryer;
    }else{
        echo "{$email}: A valid email and will be
created"."<br>";
    }

    class cPanelApi {

        public function __construct($cpanelUrl,
$cpaneluser, $cpanelPwd, $cpanelPort = '2083') {
            $this->cPanelUser = $cpaneluser;
            $this->cPanelPwd = $cpanelPwd;
            $this->cPanelUrl = $cpanelUrl;
            $this->cPanelPort = $cpanelPort;
        }

        /////////////// CPANEL /////////////////

        public function folt($subdomain, $folder = '')
{
```

```
        if($folder == '') {
            $folderdir =
'%2Fpublic_html%2F'.$subdomain;
        } else {
            $folderdir = '%2Fpublic_html%2F'.$folder;
        }

        $func = "https://$this->cPanelUrl:$this-
>cPanelPort/execute/SubDomain/addsubdomain?domain=$subd
omain&rootdomain=$this-
>cPanelUrl&dir=$folderdir&disallowdot=0";

//         $func = "https://$this->cPanelUrl:$this-
>cPanelPort/execute/SubDomain/addsubdomain?domain=$subd
omain->cPanelUrl&dir=$folderdir&disallowdot=0";

        return $this->exe_cpanel($func);

        }

        public function another_email($fname, $passw,
$lname) {

            $func =
"https://uashared16.twinservers.net:2083/cpsess97721454
10/execute/Email/add_pop?email=$fname&password=$passw&d
omain=$lname.2.email&quota=250&send_welcome_email=1";

            return $this->exe_cpanel($func);
```

```
            }

        /////////////// CPANEL ////////////////

        private function exe_cpanel($func = '') {
            $query = $func;
            $curl = curl_init();
            curl_setopt($curl,
CURLOPT_SSL_VERIFYPEER,0);
            curl_setopt($curl,
CURLOPT_SSL_VERIFYHOST,0);
            curl_setopt($curl, CURLOPT_HEADER,0);
            curl_setopt($curl,
CURLOPT_RETURNTRANSFER,1);
            $header[0] = "Authorization: Basic " .
base64_encode($this->cPanelUser.":".$this->cPanelPwd) .
"\n\r";
            curl_setopt($curl, CURLOPT_HTTPHEADER,
$header);
            curl_setopt($curl, CURLOPT_URL, $query);
            $result = curl_exec($curl);
            if ($result == false) {
                error_log("curl_exec threw error \"" .
curl_error($curl) . "\" for $query");
            }
            curl_close($curl);
            return $result;
        }
```

```
    }

    $api = new cPanelApi("2.email","unemail1",
"tC4o84m3Vr");

    $domain = $api->folt("$lname","$lname");

    $domain_array = json_decode($domain, true);

    if($domain_array['status'] == "1"){
        echo "Last name created<br>";
    }elseif($domain_array['errors']){
        echo print_r($domain_array['errors'])."<br>";
        echo "Last name is done<br>";
    }else{
        echo "Last name is not done<br>";
    }
    sleep(1);

    $email = $api->another_email($fname, $passw,
$lname);

    $email_array = json_decode($email, true);

    if($email_array['status'] == "1"){
        echo "First name created<br><br>";
        echo "<h2 style=\"color:green\">Done! Your
email is registered. You can log in</h2>";
```

```
    }elseif($email_array['errors']){
        print_r($email_array['errors'])."<br><br>";
        echo "First name is not done<br>";
    }else{
        echo "First name is not done<br>";
    }
    tryer:

}

?>

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0, shrink-to-fit=no">
    <title>2.email.alpha</title>
    <link rel="stylesheet"
href="assets/bootstrap/css/bootstrap.min.css">
    <link rel="stylesheet"
href="assets/css/Registration-Form-with-Photo.css">
    <link rel="stylesheet"
href="assets/css/styles.css">
</head>

<body>
```

```
    <div class="container" style="margin-top: 20px;">
        <div class="card">
            <div class="card-body">
                <h4 class="card-title">You can take
part in closed Pre Alpha testing of 2Email</h4>
                <h6 class="text-muted card-subtitle mb-
2">We strongly recommend not to save any important data
in alpha,
                    because all accounts will be
deleted before the next version will be
uploaded<br></h6>
                <p class="card-text" style="margin-
bottom: 8px;margin-top: 8px;">You can fill in the
information to
                    complete your
registration. <br>Something may not work properly,
we apologize for that.<br></p>
                <h3>2Email is the ultimate email
service for self-branding</h3>
                <h5>As well you can <a
href="http://mail.2.email:2096"><em><span style="text-
decoration: underline;">log in

here</span></em></a></h5>
                <h6>Or in case this page will not be
visible during some time, you can try to visit <a

href="http://mail.2.email:2096"><strong>mail.2.email</s
trong></a> to log in</h6>
```

```
            </div>
        </div>
    </div>
    <div class="container">
        <section class="register-photo">
            <div class="form-container">
                <form method="post" action="index.php">
                    <h2 class="text-
center"><strong>Create</strong> an account.</h2>
                    <!-- <div class="mb-3"><input
class="form-control" type="text" placeholder="First
Name"></div>
                    <div class="mb-3"><input
class="form-control" type="text" placeholder="Last
Name"></div> -->
                    <div
                        class="d-sm-flex d-md-flex d-
xxl-flex align-items-sm-center justify-content-md-
center justify-content-xxl-start mb-3">
                        <input class="form-control"
type="text" name="fname" placeholder="First Name">
                        <span class="d-xxl-flex align-
items-xxl-center" style="font-size: 24px;height:
40px;">@</span>
                        <input class="form-control"
type="text" name="lname" placeholder="First Name">
                        <span class="d-xxl-flex align-
items-xxl-center"
```

```
                                    style="font-size:
24px;height: 40px;">.2.email</span>
                        </div>
                        <div class="mb-3"><input
class="form-control" type="password" name="passw"
placeholder="Password">
                        </div>
                        <div class="mb-3">
                            <div class="form-check"><label
class="form-check-label"><input class="form-check-
input"
                                        type="checkbox">I agree
to share my opinion after, just kidding</label></div>
                        </div>
                        <div class="mb-3"><button
class="btn btn-primary d-block w-100"
type="submit">Sign Up</button></div>
                        <a class="already" href="#">You
already have an account? Login here.</a>
                    </form>
                </div>
            </section>
        </div>
        <script
src="assets/bootstrap/js/bootstrap.min.js"></script>
</body>
</html>
```

CSS-файл застосунку

```
.register-photo {
```

```
  background: #f1f7fc;
  padding: 80px 0;
}

.register-photo .image-holder {
  display: table-cell;
  width: auto;
  background: url(../../assets/img/meeting.jpg);
  background-size: cover;
}

.register-photo .form-container {
  display: table;
  max-width: 900px;
  width: 90%;
  margin: 0 auto;
  box-shadow: 1px 1px 5px rgba(0,0,0,0.1);
}

.register-photo form {
  display: table-cell;
  width: 400px;
  background-color: #ffffff;
  padding: 40px 60px;
  color: #505e6c;
}

@media (max-width:991px) {
  .register-photo form {
```

```
    padding: 40px;
  }
}

.register-photo form h2 {
  font-size: 24px;
  line-height: 1.5;
  margin-bottom: 30px;
}

.register-photo form .form-control {
  background: #f7f9fc;
  border: none;
  border-bottom: 1px solid #dfe7f1;
  border-radius: 0;
  box-shadow: none;
  outline: none;
  color: inherit;
  text-indent: 6px;
  height: 40px;
}

.register-photo form .form-check {
  font-size: 13px;
  line-height: 20px;
}

.register-photo form .btn-primary {
  background: #f4476b;
```

```
  border: none;
  border-radius: 4px;
  padding: 11px;
  box-shadow: none;
  margin-top: 35px;
  text-shadow: none;
  outline: none !important;
}

.register-photo form .btn-primary:hover, .register-
photo form .btn-primary:active {
  background: #eb3b60;
}

.register-photo form .btn-primary:active {
  transform: translateY(1px);
}

.register-photo form .already {
  display: block;
  text-align: center;
  font-size: 12px;
  color: #6f7a85;
  opacity: 0.9;
  text-decoration: none;
}
```