НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«Київський політехнічний інститут імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра програмного забезпечення комп’ютерних систем

**КУРСОВА РОБОТА**

***з дисципліни "Основи веб-програмування"***

Виконав: Янкевич Станіслав Дмитрович

Група: КП-82

Допущено до захисту

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1 семестр 2019/2020

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«Київський політехнічний інститут імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра програмного забезпечення комп’ютерних систем

|  |  |
| --- | --- |
| Узгоджено  Керівник роботи    \_\_\_\_\_\_\_\_\_\_/Гадиняк Р.А./ | ЗАХИЩЕНА "\_\_"\_\_\_\_\_\_\_\_\_2019р.  з оцінкою\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_/Гадиняк Р.А./ |

***Веб-додаток серв*іс електронно**[**ї**](https://docs.google.com/document/d/1Pyu7h_eC0-5JLFzSFGsLgHEKiHbeejLY0Z_0LbVcp5E/edit#heading=h.8ru3y4l0f6pl) ***пошти* із зашифрованими повідомленнями**

Виконавець роботи

Янкевич Станіслав Дмитрович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_2019р.

Зміст

**1. Аналіз предметної галузі 4**

1.1. Аналіз вимог до функціональності програмних засобів 4

**2. Аналіз мов програмування та технологій розроблення 8**

2.1. Мова програмування JavaScript 8

2.2. Фреймворк NodeJS 8

2.3. Фреймворк ExpressJS 8

2.4. База даних MongoDB 8

2.5. Файловий хостинг Cloudinary 9

2.6. Веб-хостинг Heroku 9

2.7. Бібліотека Vue.js 9

2.8. Бібліотека Cryptico 10

**3. Опис розроблених програмних засобів 11**

3.1. Загальна структура системи 11

3.2. Структура бази даних 13

3.3. Модуль сховища даних 15

**4. Аналіз розроблених програмних засобів 19**

4.1. Особливості реалізації 19

4.2. Дизайн та вміст вікон 19

**Висновки 26**

# 

# 1. Аналіз предметної галузі

## 1.1. Аналіз вимог до функціональності програмних засобів

В процесі аналізу вимог до системи було виділено такі функціональні вимоги

1. Наявний веб-інтерфейс користувача. Графічні елементи використані відповідно до їхнього основного призначення.
   1. Наявно декілька сторінок або станів односторінкового сайту і веб-форм.
   2. Дотримується єдиний стиль для всіх веб-сторінок. Всі елементи стилізовано. Кнопки створення, редагування та видалення виділені спеціальними стилями (кольорами чи іншим способом).
   3. Домашня сторінка містить логотип, опис і найважливіші посилання. На сайті наявні гіперпосилання між сторінками\розділами\сутностями.
   4. Наявна навігація по сайту у вигляді меню чи навігаційної панелі.
   5. Наявна сторінка з детальним описом сайту та контактами розробника
2. Можливість створення, редагування та видалення даних.
   1. На формі створення обов’язкові поля мають значення за замовчуванням.
   2. Наявна форма редагування даних має мати вже заповнені поля із попередніми значеннями. Є можливість відмінити оновлення даних.
   3. Можна видаляти дані (з підтвердженням цієї дії).
3. Використання графічних списків та таблиць для відображення колекцій даних.
   1. Дані в таблиці не є лише відображенням однієї таблиці (колекції) БД, а містять склеєні чи змінені дані.
   2. Дані у списках та таблицях пагінуються.
4. Можливість пошуку сутностей.
   1. Сайт дозволяє шукати сутності по одному чи декількох полях.
   2. Пошук повинен відбуватися по порядку з ігноруванням регістру символів.
   3. Показувати по яких полях і що шукається, загальну кількість знайдених результатів.
   4. При невдалому пошуку виводити повідомлення про відсутність шуканих елементів.
5. Програма взаємодіє з базою даних. Запити до бази винесено у спеціальний окремий модуль (вид сховища).
   1. База даних містить мінімум 4 таблиці (колекції).  
       Реляційні таблиці в БД приведені до 1-ї та 2-ї нормальних форм.
   2. Між таблицями (колекціями) бази даних є one-to-many та many-to-many зв'язки
6. Програма має можливість завантаження файлів з файлової системи користувача, їх збереження у сховище даних.
   1. Завантажені користувачем зображення можна переглядати на сторінках сайту.
   2. Файли зберігаються у базі даних або на спеціальному віддаленому файловому хостингу.
7. Наявна реєстрація користувачів у системі. Паролі користувачів не зберігаються у відкритому виді, а хешуються обраним алгоритмом хешування рядків.
   1. Наявна форма аутентифікації. Пароль користувача при введені у форму прихований (використано спеціальний елемент). Сторінка цієї форми має сповіщення про некоректність введених користувачем даних.
   2. Програма авторизує доступу до даних по користувачах. У базі є дані, що належать конкретному користувачу (напр, авторство сутностей).
   3. Є мінімум 2 ролі користувачів. Обов’язкова роль адміністратора, що може керувати іншими користувачами.
   4. Користувач може вийти з сайту (logout). Наявна персональна сторінка користувача.
8. У сервера є можливість доступу до даних через API.
   1. Частина API шляхів (endpoints) надають доступ лише після авторизації.
   2. На сайті є веб-сторінка із коротким описом API.
9. Всі вхідні дані від користувача та з файлів перевіряються, помилки обробляються, користувачу відображається інформація про помилки.
10. SPA сайт із використанням Front-end фреймворка Vue.
11. Використання JWT

# 

# 2. Аналіз мов програмування та технологій розроблення

## 2.1. Мова програмування JavaScript

Розробка серверу проводилась на мові JavaScript, де використовувалися асинхронні функції для ефективної роботи програми. Також вона застосовувалася при виконанні скриптів в браузері, для виконання асинхронних запитів за допомогою Fetch API до сервера, обробки отриманих даних та зберігання їх у localStorage.

## 2.2. Фреймворк NodeJS

За допомогою цього середовища організували проект розбивши виконання програми на модулі, з якими платформа надає змогу працювати та загалом створити серверну частину на мові JavaScript. Крім того були використані модулі для роботи з базами даних, файлами и т.д.

## 2.3. Фреймворк ExpressJS

За допомогою цього фреймворку отримали змогу більш зручніше працювати з логікою сервера, а саме обробляти запити на вищому рівні абстракції та організувати систему шляхів для того, щоб більш модульно працювати с кодом.

## 2.4. База даних MongoDB

Була обрана база даних MongoDB для більш зручної роботи з даними у звичному для вебу форматі - документів.

## 2.5. Файловий хостинг Cloudinary

Сервіс був використанний для зберігання фото користувачів, як зручний інструмент для зберігання фотографій і швидкого доступу до них, без необхідності їх змінення.

## 2.6. Веб-хостинг Heroku

Був використаний цей хостинг, як бескоштовна можливість розташування сервісу в мережі для загального користування

## 2.7. Бібліотека Vue.js

Застосування цього фреймворку надало можливість для зручної роботи з частиною клієнта, а саме реалізації SPA.

## 2.8. Бібліотека Cryptico

За допомогою цієї бібліотеки було реалізовано е2е шифрування. Функціонал цієї бібліотеки надає змогу генерування ключів на основі даних, що було використано для реалізації генерації публічних ключів, які посилалися на сервер для шифрування даних іншими користувачами та приватних ключів на стороні клієнта для розшифрування даних цих даних лише для нього.

# 

# 3. Опис розроблених програмних засобів

## 3.1. Загальна структура системи

Програмні засоби реалізовані у вигляді web-додатку. Структурна схема системи зображена на рис.

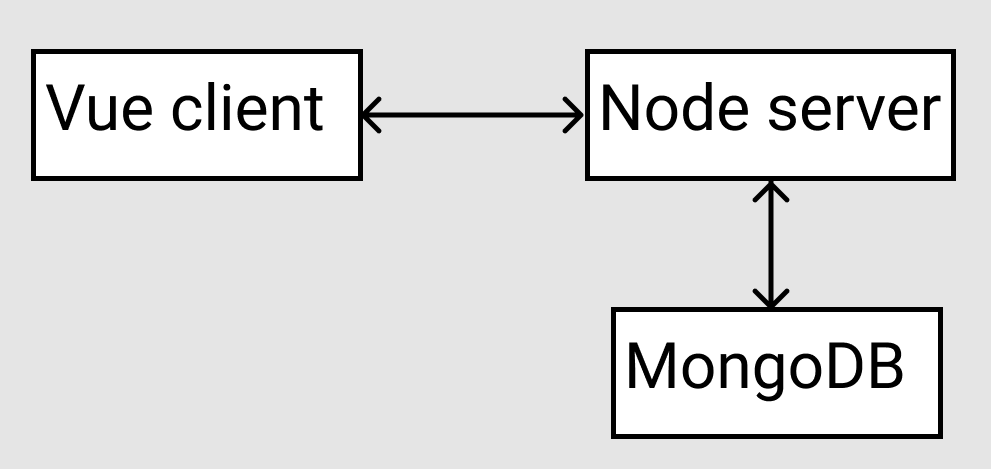


Рис. Структурна схема системи

Сервер системи обробляє запити від користувачів та виконує такі функції:

1. Відправлення обраних даних із бази даних
2. Створення нових сутностей в базі даних
3. Змінення обраних сутностей в базі даних
4. Видалення обраних сутностей з бази даних

Проект розбитий на такі модулі:

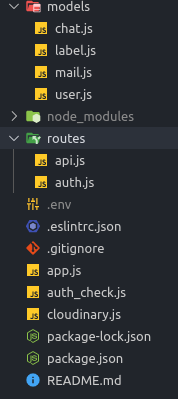
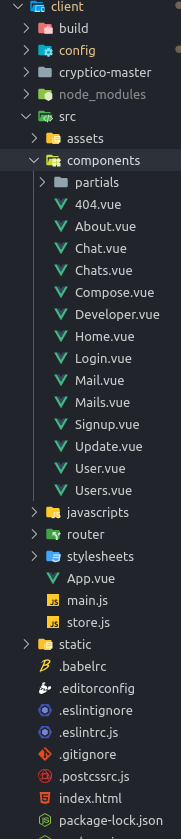
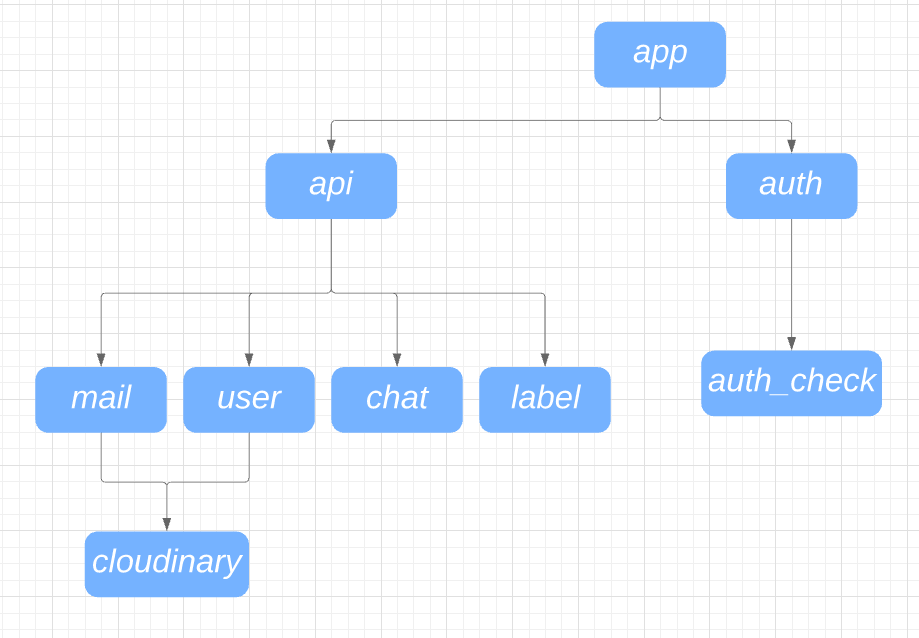


Рис. Модулі проекту

Діаграма залежностей модулів:



## 3.2. Структура бази даних

На рис. представлена ERD-діаграма структури бази даних системи.

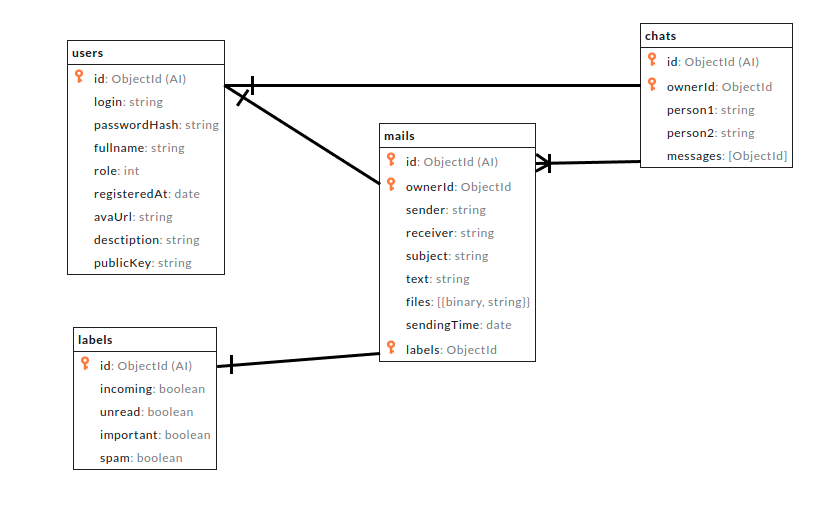


Рис. Структура бази даних

1. В колекції користувачів зберігаються логін, ім’я, дата створення акаунту, посилання на фото в cloudinary, опис, пароль в захешованому вигляді, роль користувача для обмеження доступу до ресурсів, та публічний ключ для створення листа для користувача.
2. В колекції листів зберігаються id власника(того чиїм ключем зашифровано лист), адреса відправника та отримувача, тема, та сам текст у зашифрованому вигляді, час відправки, id міток та массив об’єктів файлів, що складаються з самих даних файла і формату даних.
3. В колекції міток зберігаються булеві значення для вхідного, непрочитаного, важливого та спам листів.
4. В колекції чатів зберігаються id власника(того чиїм ключем зашифровано листи), адреси осіб між якими відбувається переписка та массив id листів в цій переписці.

## 3.3. Модуль сховища даних

|  |
| --- |
| **user.js** |
| static getById(id) {  return user.findById(id).exec();  };  static getByLogin(login){  return user.findOne({login: login}).exec();  }  static async insert(userParams) {  userParams.passwordHash = await User.hashPassword(userParams.password);  const newUser = new User(userParams);  let promiseUser = await new user(newUser).save();  return promiseUser.\_id;  }  static update(id, userParams) {  return user.findByIdAndUpdate(id, userParams, {new: true}).exec();  }  static deleteById(id) {  return user.deleteOne({\_id: id}).exec();  } |

Модуль призначений для роботи з колекціями користувачів, за його допомогою можна отримати, створити та видалити користувача, а також змінити деякі його дані.

|  |
| --- |
| mail.js |
| static getSubjectAndIdByLogin(login, userId){  return mail.find({ $and: [  {$or: [{ sender: login }, { receiver: login }]},  {ownerId: userId}  ]}, {subject:1});  }  static async getById(id) {  return mail.findById(id);  }  static async insert(mailParams, isSender) {  if(isSender){  mailParams.labels = await Label.insert({incoming: false, unread: false});  } else {  mailParams.labels = await Label.insert({incoming: true, unread: true});  }  const newMail = new Mail(mailParams);  let promiseMail = await new mail(newMail).save();  Chat.createMail(promiseMail.ownerId, promiseMail.sender, promiseMail.receiver, promiseMail.\_id);  return promiseMail.\_id;  }  static update(id, newMail) {  return mail.findByIdAndUpdate(id, newMail, {new: true}).exec();  }  static async delete(id, ownerId) {  const currMail = await Mail.getById(id);  await Label.deleteById(currMail.labels);  await Chat.deleteMail(ownerId, currMail.sender, currMail.receiver, currMail.id);  return mail.deleteOne({\_id: id}).exec();  } |

Модуль призначений для роботи з колекціями листів, за його допомогою можна отримати частичні дані кожного листа користувача, отримати певний лист, створити новий, змінити мітки в обраному та видалити бажаний.

|  |
| --- |
| label.js |
| static getById(id) {  return label.findById(id).exec();  }  static async insert(labelParams) {  const newLabel = new Label(labelParams);  let promiseLabel = await new label(newLabel).save();  return promiseLabel.\_id;  }  static update(id, newLabel) {  return label.findByIdAndUpdate(id, newLabel, { new: true }).exec();  }  static deleteById(id) {  return label.deleteOne({ \_id: id }).exec();  } |

Модуль призначений для роботи з колекціями міток, за його допомогою можна отримати, створити, змінити та видалити мітки обраного листа.

|  |
| --- |
| chat.js |
| static getAllByOwnerId(ownerId) {  return chat.find({ ownerId: ownerId});  }  static getById(id) {  return chat.findById(id).exec();  }  static getByIdAndPersons(ownerId, personX, personY)  {  return chat.findOne({ $and: [  { ownerId: ownerId },  { person1: { $in: [personX, personY] } },  { person2: { $in: [personX, personY] } }  ] }).exec();  }  static async insert(chatParams) {  const newChat = new Chat(chatParams);  let promiseChat = await new chat(newChat).save();  return promiseChat.\_id;  }  static update(id, newChat) {  return chat.findByIdAndUpdate(id, newChat, { new: true }).exec();  }  static deleteById(id) {  return chat.deleteOne({ \_id: id }).exec();  } |

Модуль призначений для роботи з колекціями чатів, за його допомогою можна отримати чат за його id або id власника листа та адреси іншого учасника переписки. Також можна створити, змінити та видалити обраний чат

# 

# 4. Аналіз розроблених програмних засобів

## 4.1. Особливості реалізації

Для реєстрації та входу використовується логін, який є email адресою користувача та пароль на основі якого створюються ключі для шифрування, через це пароль не можливо змінити або відновити.

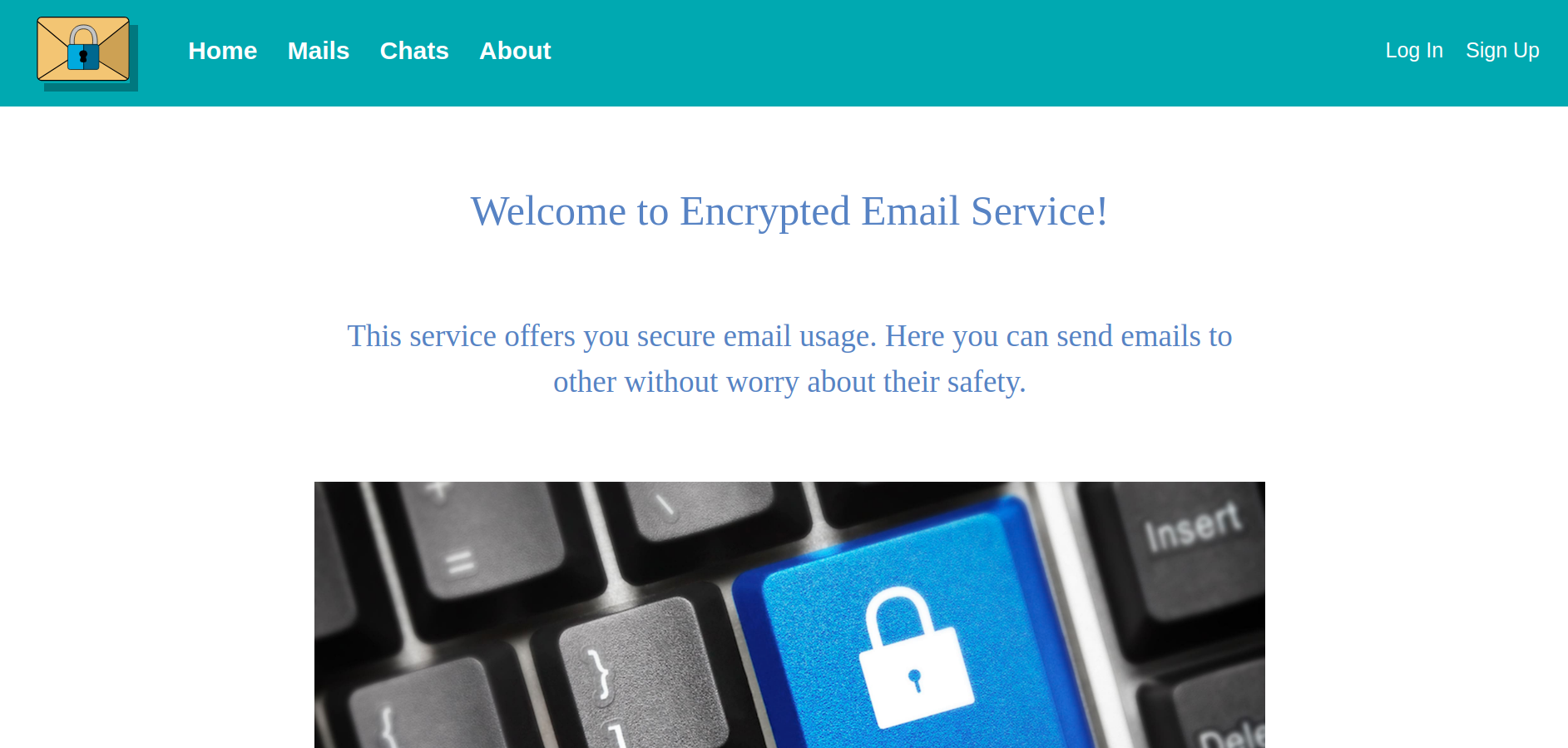
Існує дві ролі користувачів, крім звичайних користувачів є адміністратори, котрі можуть переглядати сторінки користувачів та робити їх адміністраторами.

Всі дії відбуваються за допомогою асинхронних функцій, що надає змогу користуватися ресурсом з більшою продуктивністю.

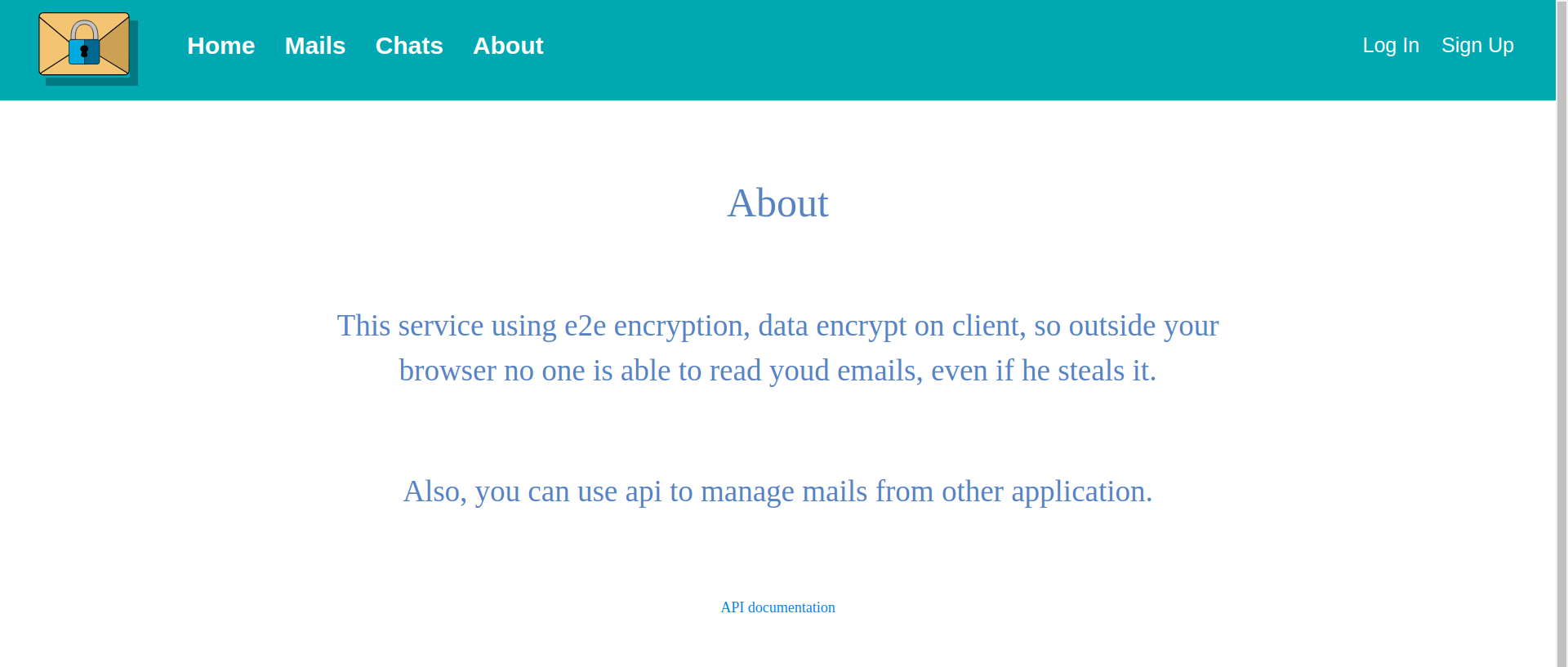
## 4.2. Дизайн та вміст вікон

Для відображення даних система має графічний web-інтерфейс користувача, що складається з:

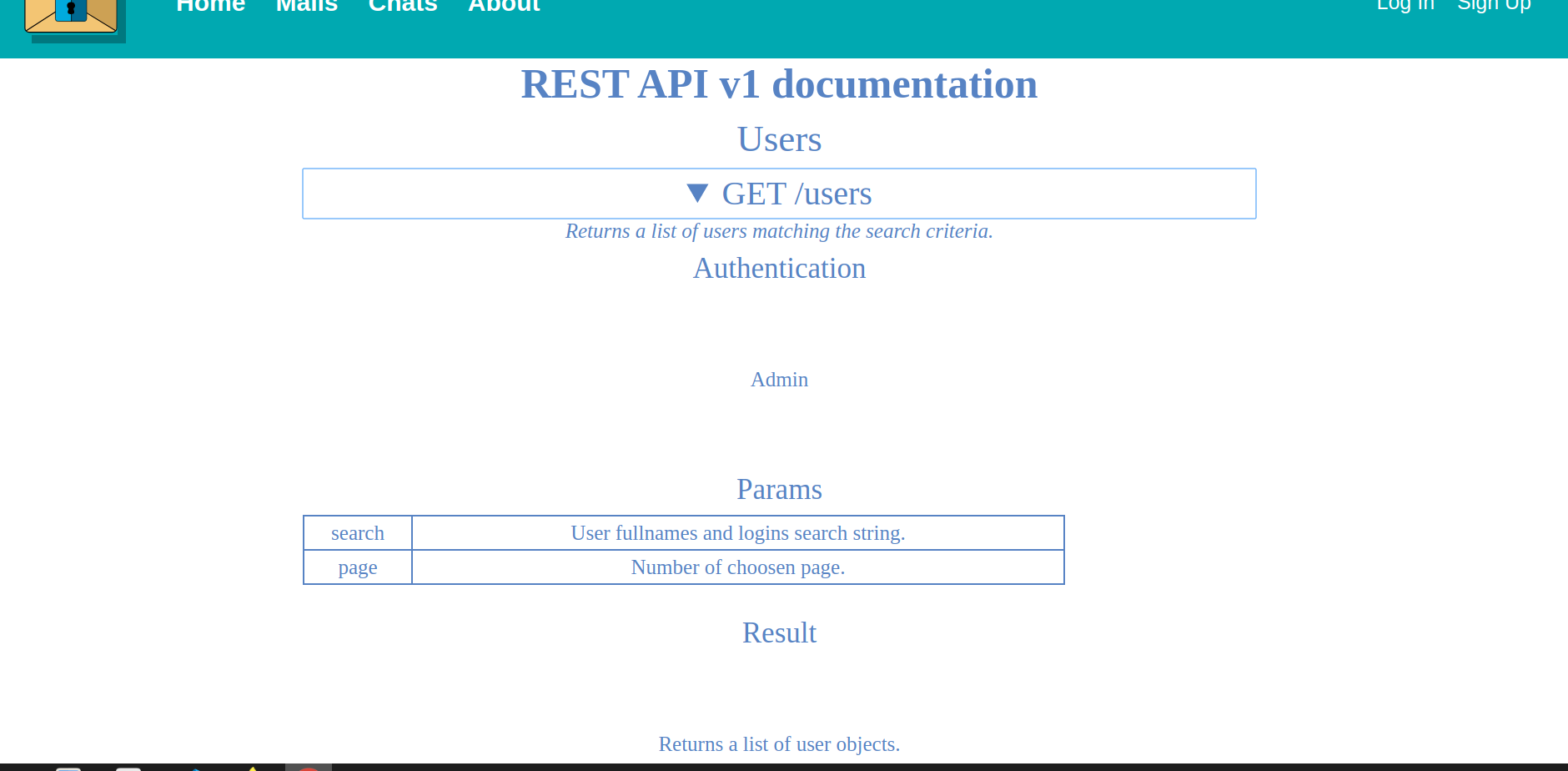
1. Домашня сторінка
2. Сторінка з детальною інформацією про сайт
3. Сторінка реєстрації
4. Сторінка входу
5. Сторінка листів
6. Сторінка листа
7. Сторінка створення листа
8. Сторінка чатів
9. Сторінка чата
10. Сторінка користувачів
11. Сторінка користувача
12. Сторінка зміни даних користувача
13. Сторінка з документацією API



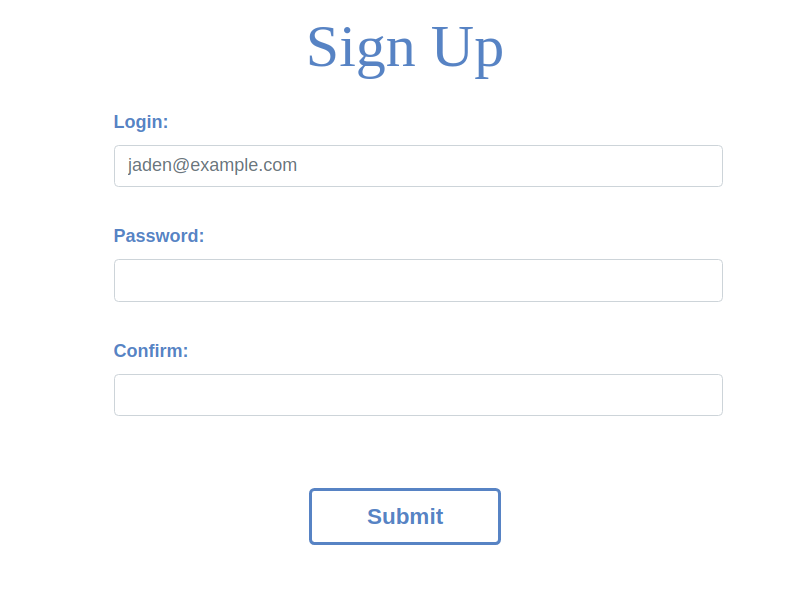
Домашня сторінка з описом сервісу та навігацією на основні ресурси



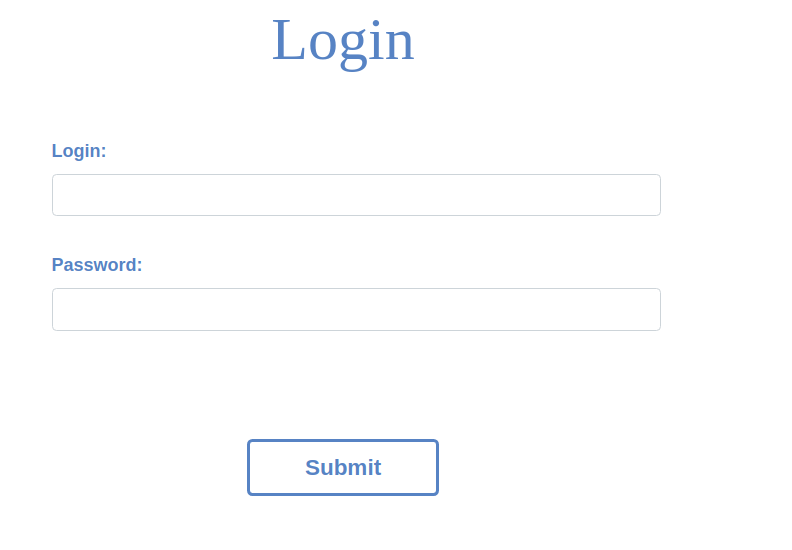
Сторінка з детальною інформацією про реалізацію можливостей роботи сервісу.



Сторінка з документацією API сервісу.

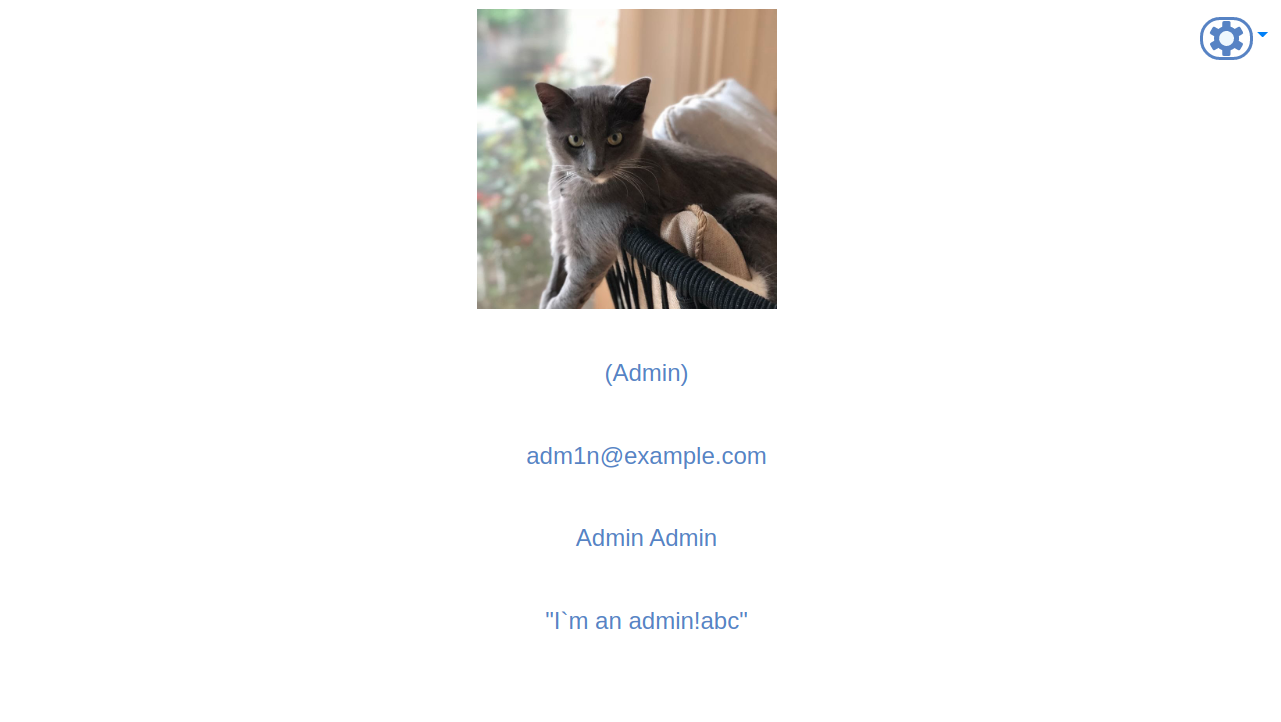


Сторінка реєстрації за унікальним логіном(адресою).

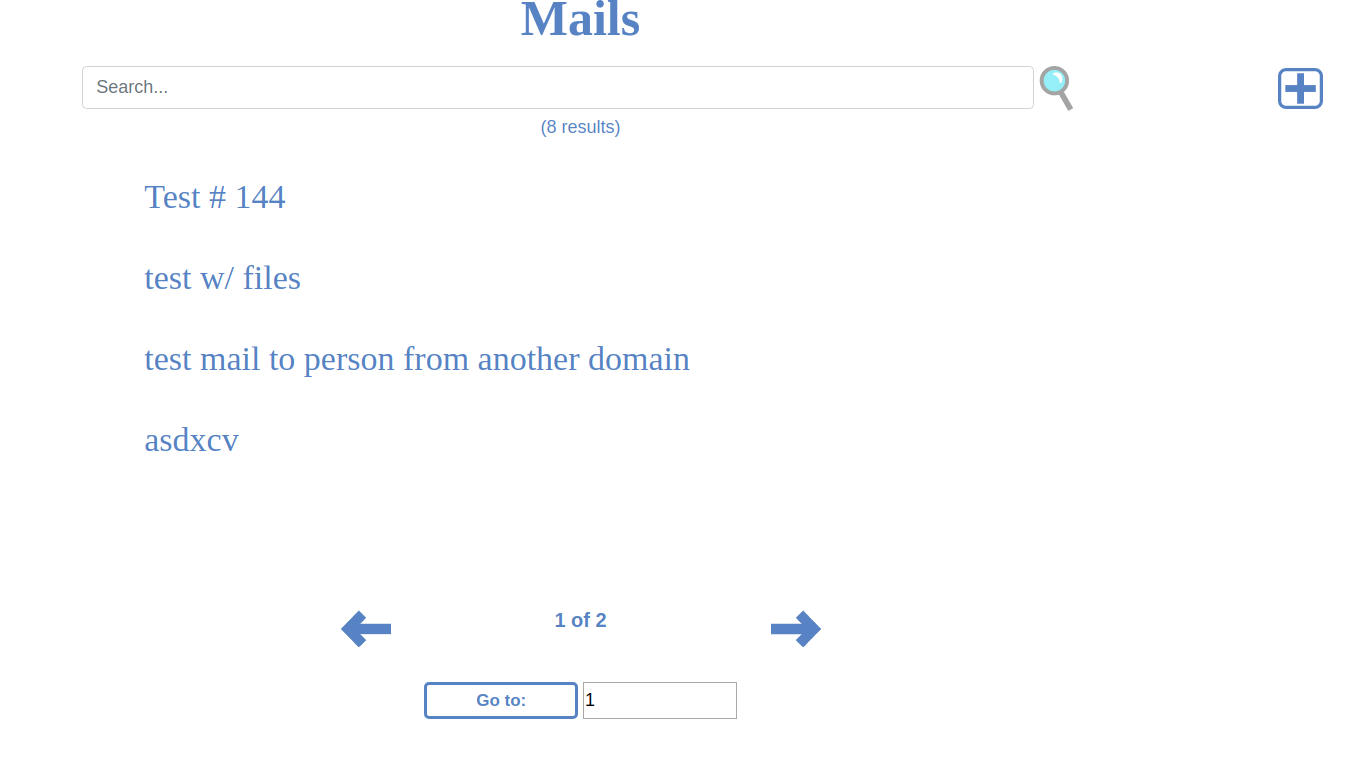


Сторінка входу.

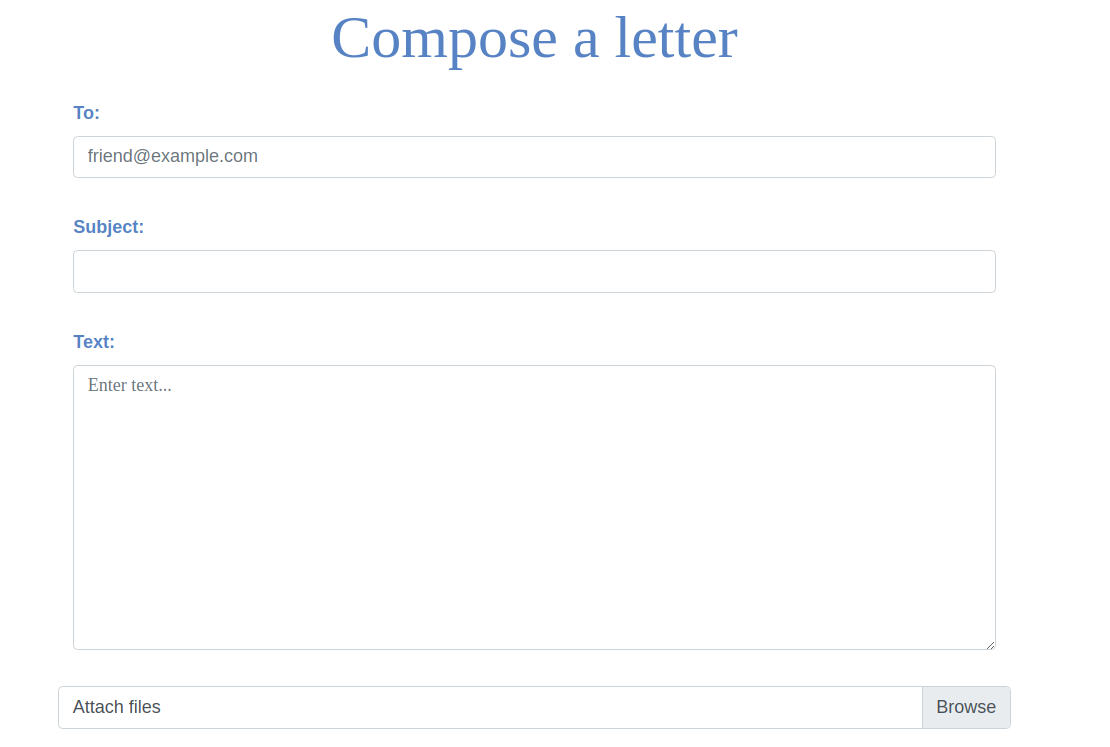
Сторінка користувачів, на котрій можна перейти до будь-якого користувача.

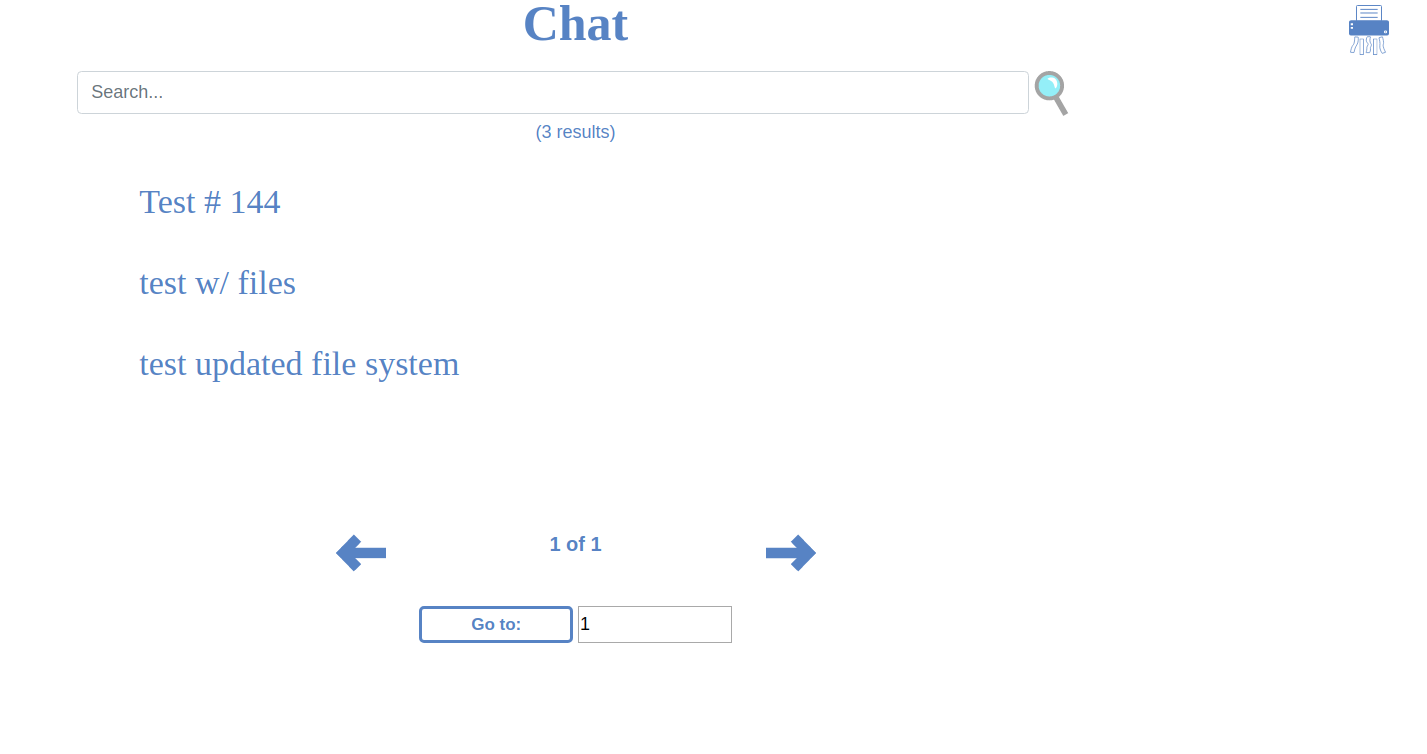


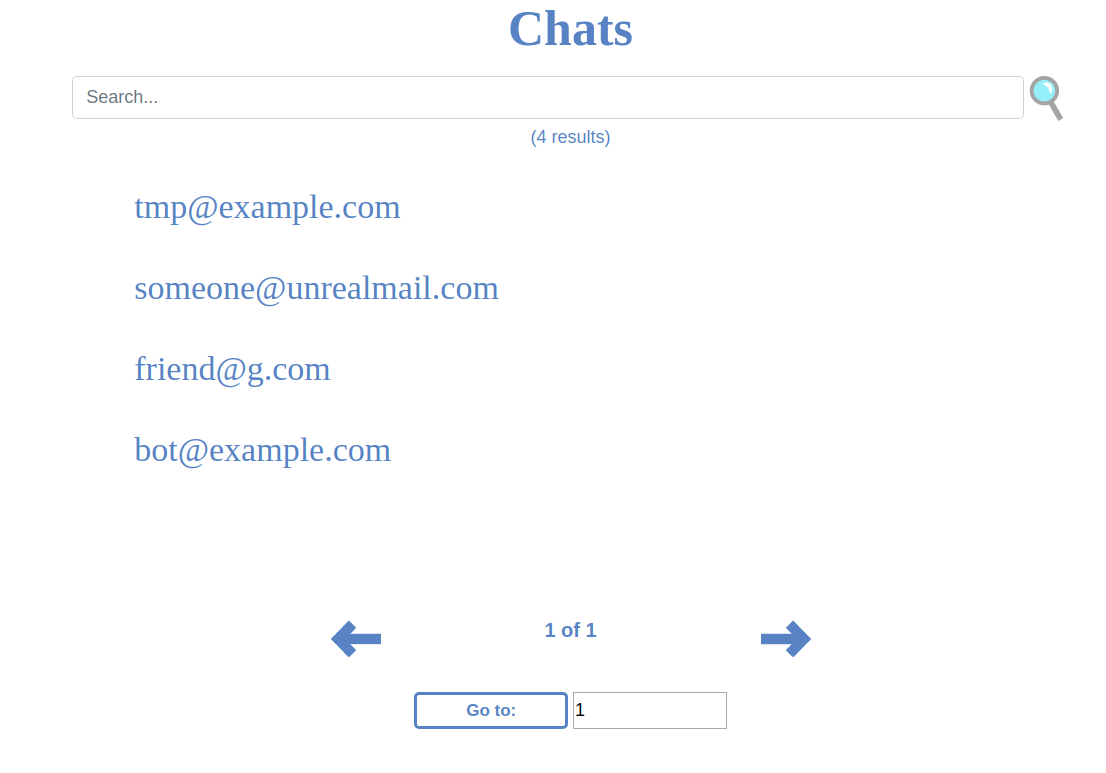
Сторінка користувача, на котрій він може змінити дані, або видалити себе.



Сторінка листів де можна перейти до будь-якого листа та перейти до створення нового.

Сторінка листа, де можна видалити лист чи змінити мітки.Сторінка створення листа .

Сторінка чата, де можна перейти до будь-якого листа, або видалити чат.

Сторінка чатів, де можна перейти до будь-якого чата.

# Висновки

Метою даного проекту було розроблення web-додатку для електроного листування з зашифрованими листами.

Розроблена система:

* забезпечує авторизований доступ до системи;
* дозволяє управляти листами, чатами та користувачами;
* дає можливість створювати зашифровані електронні листи.

Особливу увагу під час розроблення даного програмного продукту було приділено шифрування даних, для найкращого захисту даних.

Використання розробленої системи дозволить захищено листуватися користувачам даного сервісу.