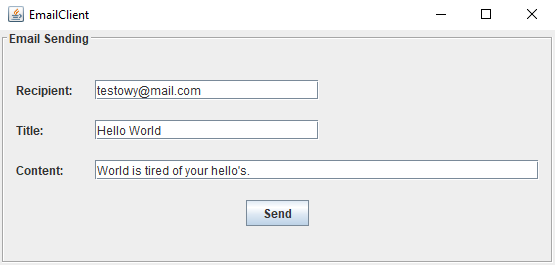
**Dokumentacja programu Klient Pocztowy**

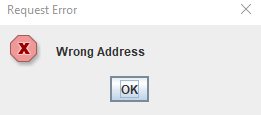
**Instrukcja Obsługi Klienta Poczty Email:**

Klient użytkownika usługi e-mail bazuje na prostej obsłudze i aktywacji.

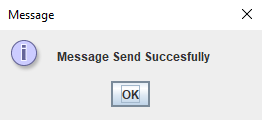


Po włączeniu aplikacji klienta wyskakuje okienko „EmailClient”, który pokazuje trzy pola wejściowe i przycisk Send. Do pola Recipient należy wpisać adres email, na który ma być wysłana wiadomość. Pole Title to tytuł wysyłanej wiadomości. W Content należy napisać pełną wiadomość jaką pragnie się wysłać. Przyciskiem „Send” wykonujemy czynność wysyłania emaila na podany adres.

Jeżeli email jest nieprawidłowy to wyskoczy okienko informujące o wpisanym złym adresie email.



Gdy wszystko zostało poprawnie wpisane do odpowiednich okienek i wysłane, wtedy wyskoczy okienko potwierdzające pomyślny przebieg operacji wysyłania emaila.



Aby zakończyć działanie klienta, wystarczy kliknąć na przycisk „X”, który zamknie proces.

**Opis Usług Serwera:**

Serwer składa się z 5 różnych klas obsługują zapisywanie emaili w pamięci i wysyłanie ich do pożądanego odbiorcy.

W klasie MainController znajduje się @RestController z obiektem LOGGER, który rejestruje logi wysyłane do aplikacji. W @RequestMapping wartość przyjmuje ścieżkę, wywołuje żądanie RequestMethod.POST i zwraca wartość w JSON’ie. ResponseEntity<?> tworzy plik, który, poprzez RequestBody, mapuje ciało HttpRequest do obiektu mail, klasy Mail, i deserializuje JSON’a do typu Java.

Jeżeli żądanie zostało przeprowadzone pomyślnie i w kliencie zostały wprowadzone właściwe parametry w podanych polach, zwracany jest status OK. (code: 200) przez ResponseEntity. Jeżeli nastąpił problem z wysyłaniem emaila, serwer zwróci kod odpowiedzi 400.

Klasa Send zawiera logowania pod public static void send, gdzie final String pass to hasło, a final String from to mail użytkownika. Hasło jest uwierzytelniane poprzez funkcję getPasswordAuthentication, które pozwala na przeprowadzenie wysyłania emaili. W warunku try, klasy Send, znajduje się pobieranie danych tekstowych, które zostały wprowadzone w kliencie, pobieranie nazwy email osoby wysyłającej, pobieranie adresu email do odbiorcy emaila i, ostatecznie, wysyłanie emaila pod wprowadzony adres odbiorcy. Jeżeli wszystko zostało przeprowadzone pomyślnie, serwer wypisze w konsoli wiadomość „Message sent successfully…”.

Klasa Main zawiera obiekty typu String: recipient, title i kontent, wraz z powiązanymi do nich konstruktorami. Znajdują tam się funkcje tzw. Getter’y i Setter’y, które zwracają odpowiedni obiekt String. Dla getTo będzie zwracany recipient. Dla getSubject będzie zwracany title, a dla getBody, będzie zwracany content. Jest tzw. „kapsułkowanie”, dzięki któremu zmienne i metody zostają „zwinięte” w jeden segment i klasy, które chcą korzystać ze zmiennych zawartych w klasie Main, będą musiały to zrobić poprzez funkcje Getter i Setter.

Klasa FileDTO otwiera plik „email\_backup.csv” i za pomocą metody addUserToFile pobiera parametry z klasy main, wiadomość z klasy Send i zapisuje je do pliku w formie nowego rejestru poprzez funkcję BufferedWriter. W metodzie getId(), każdy z emaili ma przypisany numer ID, który jest również zapisywany i zwracany do odpowiedniego użytkownika.

Klasa Application bazuje na funkcji „Spring-Boot”, dzięki której serwer jest aktywowany do pracy.

**Narzędzia Wykorzystane do Pracy:**

Server:  
1. Java (Środowisko InteliJ)  
2. Java (Spring Framework)  
3. Java (Maven)

Klient:  
1. Java (Środowisko InteliJ)  
2. Java Swing

**Tabela Podziału Pracy:**

|  |  |
| --- | --- |
| Przemysław Pyrda | Opracowanie GUI Klienta, Wysyłanie i zapisywanie Emailów. Opracowanie Try-Catch’y, Opracowanie struktury serwera, FileMetaData do klienta. Controller & Mapowanie do serweru. |
| Krzysztof Pikul | Opracowanie GUI Klienta, File DTO, Obsługa Wyjątków, Funkcja getID. Szablon projektu klienta pocztowego. |
| Szymon Bernady | Opracowanie klasy Mail serwera, klasy Main klienta, rewizje Mavena, Zwiększenie Czytelność Kodu, Dokumentacja, Opracowanie UserData klienta. |