

-Spam-Detektor

Programmierung 2

Prof. Dr. Bradley Richards



Abbildung 1 Deckblatt

Mai 2023

WIVZ 1.51

Philipp Dunkel

Hanja Heimann

Ece Kaya

Inhaltsverzeichnis

INHALTSVERZEICHNIS	- 2 -
1. EINLEITUNG	- 4 -
2. THEORETISCHE GRUNDLAGEN	- 5 -
3. PRAKTISCHE IMPLEMENTATIONSKONZEPT	- 8 -
4. ERGEBNISSE	- 13 -
4.1 ZWECK	- 13 -
4.2 Programmieren	- 13 -
4.3 Einschränkungen	- 14 -
5 SCHLUSSFOLGERUNGEN	- 15 -
5.1 weiterführende Untersuchungsmöglichkeiten	- 15 -
5.2 Wichtigste Erkenntnisse	- 16 -
6 QUELLENVERZEICHNIS	- 17 -
6.1 Java API	- 17 -
7 LITERATURVERZEICHNIS	- 17 -
8 ABBILDUNGSVERZEICHNIS	- 17 -
9 GLOSSAR	- 18 -
9.1 Spam-E-Mail	- 18 -
9.2 Phising E-Mails	- 18 -
9.3 Malware	- 18 -

9.4	Feature Selection Techniken	- 18 -
9.5	Overfitting	- 18 -
9.6	Machine Learning	- 19 -
9.7	Algorithmen	- 19 -
9.8	CSV- File	- 19 -
9.9	EML	- 19 -
9.10	HTML	- 20 -
9.11	CSS	- 20 -
9.12	Instanvariable	- 20 -
9.13	Konstruktor	- 20 -
9.14	GUI	- 20 -
9.15	JAR	- 21 -
10	ANHANG	- 22 -
10.1	Erste Logik Ideen	- 22 -
10.2	Erster Entwurf	- 22 -
11	AUSFÜHREN ALS JAR DATEI	- 24 -

1. Einleitung

Jeder Mensch, welcher E-Mails als Kommunikationsmittel nutzt, kennt den Spam Ordner. Die E-Mail-App zeigt eine neu eingegangene E-Mail an, die Unsicherheit, ob man den Link in der E-Mail bedenkenlos anklicken kann oder den Anhang öffnen soll, kennen viele Menschen. Die Spam E-Mails werden immer gerissener und damit schwerer erkennbar.

Der Spam-Detektor soll eine Hilfe für den Alltag sein- mit dem Spam-Detektor sollen Spam E-Mails schneller erkannt und eliminiert werden und somit keine Gefahr für unsere elektronische Geräte darstellen.

Die Bedienung ist einfach: Die Spam-Detektor Applikation ausführen, unter «Add E-Mail», die gewünschten E-Mails auswählen, die E-Mails müssen im vorhinein als eml-Datei abgespeichert werden.

Die angewählten E-Mails werden nun in der Tabelle angezeigt und mit dem Klick auf «Check Spam» werden die E-Mails auf Spam gescheckt. So bald Begriffe, die in unserer Blacklist eingetragen sind in der E-Mail erkannt werden, werden diese als Spam eingestuft. Der Spam-Detektor teilt die Bedrohungen der E-Mails in true und false ein. Das Resultat wird in der Spalte Spam ganz rechts angezeigt.

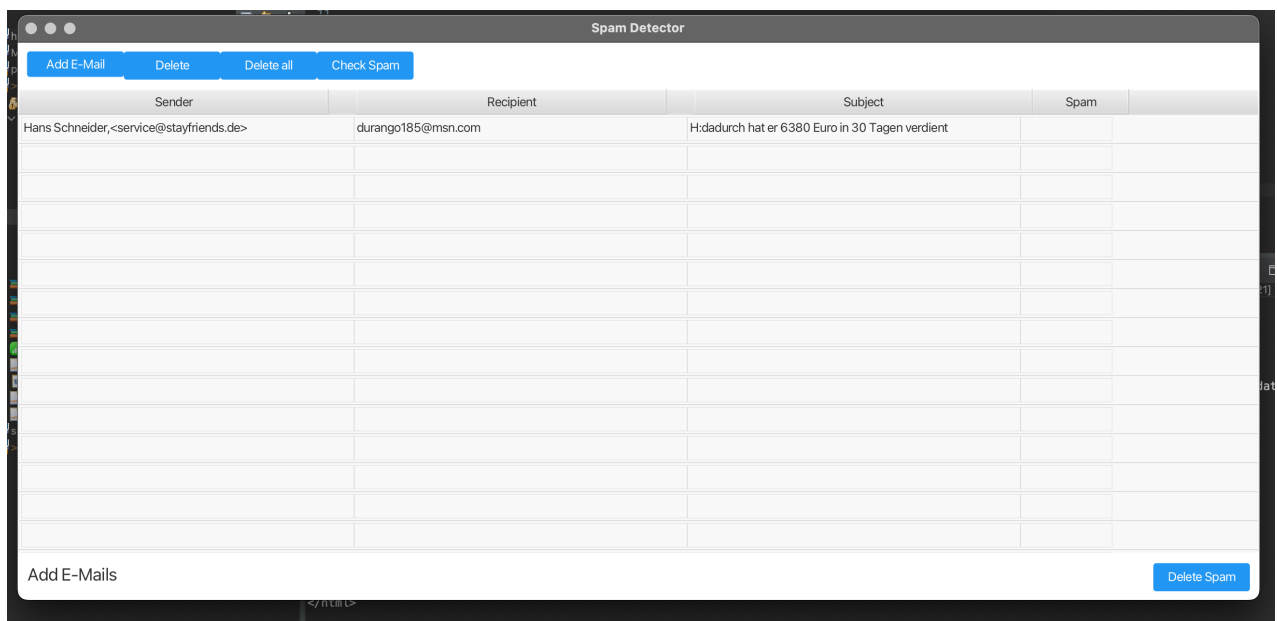


Abbildung 2 Übersicht Anwendung

2. Theoretische Grundlagen

Für das Requirements Engineering (RE) wurde die klassische reverse Methode (hierbei wird eine Aufgabenstellung analysiert, die Anforderungen identifiziert und dokumentiert) gewählt, um ein vorgegebenes Ziel innerhalb einer bestimmten Zeit zu erreichen. Die theoretischen Grundlagen des Requirements Engineering (RE) basieren auf der Annahme, dass es unerlässlich ist, die Bedürfnisse und Anforderungen der Stakeholder zu verstehen und zu dokumentieren, um eine erfolgreiche Softwareentwicklung zu gewährleisten.

Die Aufgabenstellung wurde analysiert und die Anforderungen in must/can-Anforderungen unterteilt. Zusätzlich wurden eigene Anforderungen hinzugefügt, die im Rahmen der Möglichkeiten und der vorgegebenen Zeit umsetzbar waren.

Im nächsten Schritt wurde ein mögliches GUI skizziert, um die Aufgabe und die Anforderungen besser zu visualisieren. Die möglichen Funktionen (Buttons) wurden dargestellt sowie die allgemeine Darstellung geplant, um einen Anfangspunkt für das eigentliche Programmieren zu haben.

Als erster praktischer Schritt wurde ein Git-Repository für das Projekt erstellt und mit Eclipse verknüpft. Das MVC-Prinzip wurde gewählt, mit einer main-class, welche die verschiedenen Objekte zusammenführt.

```
1 package SpamDetectorV2;
2
3 import javafx.application.Application;
4
5 //PD Mainclass Spamdetector
6 public class SpamMVC extends Application{
7     private SpamView view;
8     private SpamModel model;
9     private SpamController controller;
10
11 //PD start method
12 @Override
13 public void start(Stage stage) throws Exception {
14     model = new SpamModel();
15     view = new SpamView (stage, model);
16     controller = new SpamController(view, model);
17     view.start();
18 }
19
20 //PD Main method
21 public static void main (String[] args) {
22     launch(args);
23 }
24
25 }
26
27
28 }
```

Abbildung 3 SpamMVC main class

Das MVC-Prinzip ist eine Klassen-Architektur, die eine klare Trennung der Anwendungslogik von der Benutzeroberfläche ermöglicht. Dadurch wird der Code übersichtlicher, leichter wartbar und es ist einfacher, Änderungen an der Anwendung vorzunehmen. Ausserdem kann so parallel an verschiedenen Teilen der Anwendung gearbeitet werden, da die Module unabhängig voneinander sind.

Ein weiterer wichtiger Aspekt ist die Verwendung von externen Datenquellen, wie in unserem Beispiel die Verwendung einer CSV-Datei als Blacklist.

Die MVC-Architektur besteht aus drei Komponenten: dem Model, der View und dem Controller.

- Das Model beinhaltet die Anwendungslogik und die Daten, auf die die Anwendung zugreift und ist somit für die Datenverarbeitung und die Geschäftslogik verantwortlich.
- Die View stellt die Benutzeroberfläche dar und ist für die Darstellung der Daten verantwortlich, welche das Model bereitstellt.
- Der Controller ist für die Verarbeitung von Benutzerinteraktionen zuständig und steuert die Interaktion zwischen Model und View.

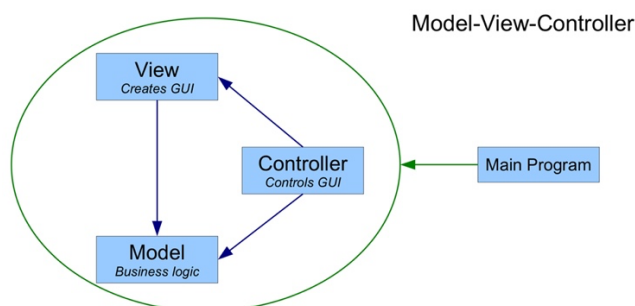


Abbildung 4 Model-View-Controller Pattern (Vorlesung 2, Folien Richards Bradley)

Die GUI wurde mit einem CSS (Spam.css) gestaltet. In der SpamController-Klasse wurden die Buttons mit ActionEvents besetzt. Im SpamModel wurden die Methoden für die ActionEvents implementiert.

Um E-Mails hinzuzufügen, wurde eine eigene E-Mail-Klasse erstellt. E-Mails wurden von eml in html umgewandelt und ein E-Mail-Objekt wurde erstellt. Die Umwandlung von E-Mails ins html-Format ermöglicht es, dass diese in einem Java-Programm angezeigt und weiterverarbeitet werden können.

Eine eigene Klasse für E-Mails ist hilfreich, da sie die E-Mail-Objekte im Programm strukturiert und kapselt. Dadurch können bestimmte Eigenschaften einer E-Mail wie z.B. Absender, Betreff oder Inhalt auf einfache Weise abgerufen und bearbeitet werden. Ausserdem erleichtert eine eigene E-Mail-Klasse das Verarbeiten von E-Mails, da sie spezifische Methoden und Attribute bereitstellen kann, die für das Programm notwendig sind. Zudem ermöglicht sie eine bessere Lesbarkeit und Wartbarkeit des Codes und erleichtert somit die Entwicklung und das Debugging.

Die Verwendung einer eigenen Klasse für E-Mails kann ebenfalls dazu beitragen, den Code besser zu strukturieren und zu kapseln. Eine eigene Klasse für E-Mails ermöglicht es, bestimmte Eigenschaften einer E-Mail wie Absender, Betreff oder Inhalt auf einfache Weise abzurufen und zu bearbeiten. Dies dient der besseren Lesbarkeit und Wartbarkeit des Codes. Zudem erleichtert es die Entwicklung und das Debugging.

Für die Blacklist wurde eine Excel-Datei als CSV-Datei verwendet, um Spambegriffe von dem Code zu trennen. Eine Blacklist-Klasse griff auf die CSV-Datei zu, las sie ein und speicherte die einzelnen Wörter in einer String-ArrayList. Die CSV-Datei wurde in der Working Directory des Projekts gespeichert, um das Programm auf verschiedenen Computern ausführen zu können und Probleme mit den Zugriffsrechten innerhalb des Packages zu vermeiden. Durch das Erstellen einer eigenen Klasse für das Einlesen eines CSV-Files wurde die Wiederverwendbarkeit des Codes verbessert und die Logik des CSV-einlesens besser gekapselt.

Zusammenfassend lässt sich sagen, dass die theoretischen Grundlagen der Softwareentwicklung auf der Annahme basieren, dass es unerlässlich ist, die Bedürfnisse und Anforderungen der Stakeholder zu verstehen und zu dokumentieren. Um dies zu erreichen, können verschiedene Methoden eingesetzt werden, wie zum Beispiel die Reverse Engineering Methode. Darüber hinaus ist es wichtig, eine klare Trennung von Daten, Anwendungslogik und Benutzeroberfläche zu ermöglichen, wie es beim MVC Design Pattern der Fall ist.

Die SpamMVC Klasse ist die Hauptklasse, diese initialisiert und startet die Anwendung. Die SpamView-Klasse ist dafür verantwortlich, das grafische Benutzerinterface (GUI) der Anwendung zu erstellen und zu verwalten. Die Klasse enthält verschiedene Elemente wie eine TableView, Buttons, Labels, etc., die alle als Instanzvariablen definiert sind. Diese Elemente werden im Konstruktor initialisiert und konfiguriert, indem sie in der GUI-Hierarchie positioniert und formatiert werden. Es gibt auch eine start()-Methode, die das GUI anzeigt, sobald es bereit ist.

Die SpamView-Klasse verwendet auch das SpamModel-Objekt, um die E-Mails anzuzeigen, die der Benutzer hinzufügt und löscht. Die TableView wird so konfiguriert, dass sie die E-Mails aus dem SpamModel anzeigt, und die Buttons werden verwendet, um die E-Mails hinzuzufügen, zu löschen, alle E-Mails zusammen zu löschen und zu überprüfen, ob eine E-Mail als Spam klassifiziert werden soll.

Insgesamt bietet die SpamView-Klasse eine benutzerfreundliche Schnittstelle, um die vom SpamModel bereitgestellten Funktionen zu nutzen und die E-Mails des Benutzers anzuzeigen und zu verwalten.

Die Klasse «SpamModel» erhält eine E-Mail-Liste als «ObservableList» und eine «SpamView» als View.

Die Klasse hat eine Methode «chooseEmFile()», die einen Dateiauswahldialog öffnet und die ausgewählte Datei als «selectedFile» speichert.

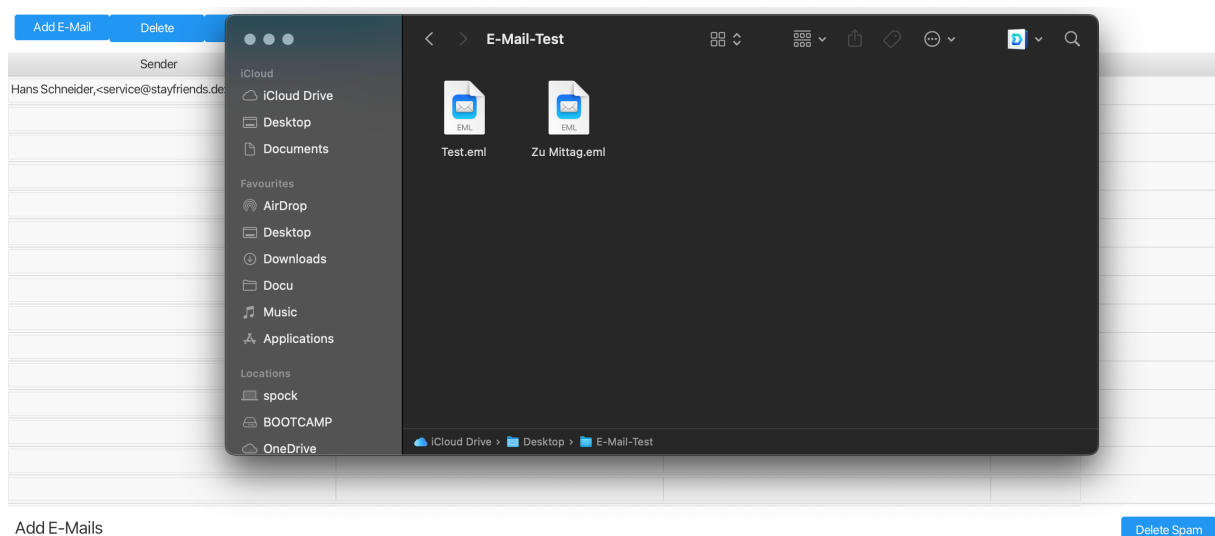


Abbildung 7 Datenauswahldialog

Die Methode «add()» liest die ausgewählte E-Mail-Datei aus und konvertiert sie in eine Textdatei, die dann als «textFileName» gespeichert wird. Die Methode erstellt eine E-Mail-Instanz aus der Textdatei und fügt sie der E-Mail-Liste hinzu.

Die Methode «checkSpam()» überprüft die E-Mail-Liste auf Spam, indem sie eine «Blacklist» mit Schlüsselwörtern liest, die in einer CSV-Datei gespeichert sind. Die Methode sucht dann in jeder E-Mail nach den Schlüsselwörtern und gibt eine Spam-Bewertung zurück, die auf der Anzahl der gefundenen Schlüsselwörter basiert. Die Spam-Bewertung wird als «SpamScore» in jeder E-Mail gespeichert, und wenn der Wert grösser als 0 ist, wird die E-Mail als Spam gekennzeichnet.

```
//PD Check if Spam and set a spamScore
public void checkSpam(){
    Blacklist blacklist = new Blacklist("blacklist-spam-detector1.csv");
    ArrayList<String> keywords = blacklist.readFile();
    ArrayList<String> matchingWords = new ArrayList<String>();
    //Zur Kontrolle
    for(String s : keywords) {
        System.out.print(s + " ");
    } System.out.println();

    for (Email m : mails) {
        int spamScore = 0;
        ArrayList<String> content = m.getContent();
        //Zur Kontrolle
        for(String s : content) {
            System.out.print(s + " ");
            System.out.println();
        }
        for (String keyword : keywords) {
            for (String s : content) {
                if (s.equals(keyword)) {
                    spamScore++;
                    matchingWords.add(s);
                }
            }
        }
        m.setSpamScore(spamScore);
        m.setSpam(spamScore > 0);
        //Zur Kontrolle
        System.out.println(m.getSpamScore());
        System.out.println(m.isSpam());
        for (String s : matchingWords) {
            System.out.print(s + " ");
        }
    }
}
```

Abbildung 8 checkSpam Methode

Die Methode «getMailList()» gibt die Liste der E-Mails als «ObservableList» zurück.

Die Klasse SpamController ist die Schnittstelle zwischen der Benutzeroberfläche (SpamView) und der Anwendungslogik (SpamModel) bildet. Die wichtigsten Ideen der Implementierung sind:

- Der Konstruktor erstellt eine Instanz der Klasse und initialisiert die View und das Model.
- Die handle()-Methode behandelt alle Benutzeraktionen, die von der View ausgelöst werden, indem sie die Schaltflächenidentität abrufen und dann den entsprechenden Code für jede Schaltfläche ausführt. Es gibt fünf Schaltflächen:

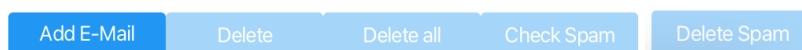


Abbildung 9 Ansicht von allen Buttons

- «Add E-Mail»: Fügt eine neue E-Mail zur Liste der E-Mails im Model hinzu und aktualisiert die View.
- «Delete»: Entfernt die ausgewählte E-Mail aus der Liste im Model und aktualisiert die View.

- «Delete All»: Entfernt alle E-Mails aus der Liste im Model und aktualisiert die View.
- «Check»: Überprüft, ob eine E-Mail Spam ist, und aktualisiert die View.
- «Delete Spam»: Entfernt alle Spam-E-Mails aus der Liste im Model und aktualisiert die View.
- Der Code aktualisiert auch den Status der Schaltflächen, indem er sie aktiviert oder deaktiviert, basierend darauf, ob es E-Mails in der Liste gibt oder nicht.
- Es gibt ein Eventhandler, der die gesamte E-Mail anzeigt, wenn der Benutzer auf eine bestimmte E-Mail in der Tabelle doppelklickt.

```
view.table.setOnMouseClicked(event -> {
    if (event.getClickCount() == 2) {
        Email selectedRow = view.table.getSelectionModel().getSelectedItem();
        if (selectedRow != null) {
            int index = view.table.getItems().indexOf(selectedRow);
            EmailView emailView = new EmailView(new Stage(), model, index);
            emailView.start();
        }
    }
});
```

Abbildung 10 Methode für den Doppelklick

Die Klasse E-Mail erzeugt ein E-Mail-Objekt mit verschiedenen Attributen wie ID, Sender, Empfänger, Betreff, Inhalt, Spam-Score und Spam-Flag. Der Konstruktor der Klasse initialisiert die Attribute ID, Sender, Empfänger, Betreff und Inhalt. Es gibt auch Getter- und Setter-Methoden für die Attribute Sender, Empfänger, Betreff, Spam-Score und Inhalt, sowie eine Methode zum Abrufen des Inhalts einer E-Mail als Liste von Wörtern, die aus dem Sender, Betreff und Inhalt extrahiert wurden.

Die E-MailView verwendet, um eine einzelne E-Mail anzuzeigen und dem Benutzer die Möglichkeit zu geben, den Inhalt der E-Mail zu einzusehen.

Die Ansicht besteht aus einem BorderPane und einem GridPane. Das GridPane wird verwendet, um die Details der E-Mail anzuzeigen, während das BorderPane verwendet wird, um das GridPane und den Inhalt der E-Mail (in Form eines WebView) anzuzeigen.

Das WebView wird verwendet, um den Inhalt der E-Mail anzuzeigen und wird in einem ScrollPane platziert, um das Scrollen des Inhalts zu ermöglichen. Das ScrollPane wird dann zusammen mit dem GridPane in einem VBox platziert, der als Zentrum des BorderPane fungiert.

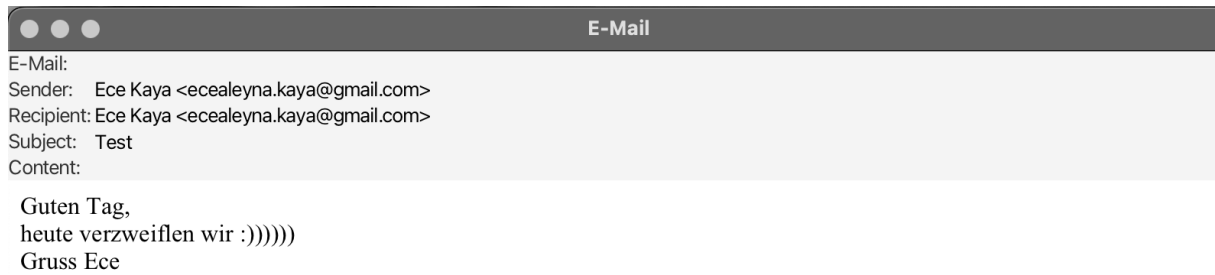


Abbildung 11 EMailView mit einer Beispiel-E-Mail

Die Blacklist-Klasse ist verantwortlich dafür CSV-Inhalte aus einer Datei zu lesen und in eine Array List zu speichern.

Die Idee hinter diesem Code ist, dass eine Liste mit bestimmten Wörtern oder Ausdrücken, die als Spam erkannt werden sollen, in einer CSV-Datei gespeichert wird. Dann kann die Blacklist-Klasse verwendet werden, um diese Liste aus der Datei zu lesen und in einer ArrayList zu speichern. Diese ArrayList kann dann von anderen Klassen verwendet werden, um E-Mails oder andere Textnachrichten zu durchsuchen und festzustellen, ob sie einen Spam-Text enthalten.

4. Ergebnisse

4.1 Zweck

Der Spam Detector ist an erster Stelle, ein Spam Detector. Heisst die Applikation erkennt, ob die eingelesenen E-Mails Spam enthalten oder nicht. Falls die E-Mail Spam enthält, kann sie mit dem «Delete Spam» gelöscht werden. E-Mails, die keinen Spam enthalten können, problemlos geöffnet und gelesen werden.

4.2 Programmieren

In unserer Programmiergruppe haben wir erfahren, dass es manchmal anstrengend sein kann, den Code von anderen Programmierern zu verstehen, insbesondere wenn sie verschiedene Technologien und Frameworks verwenden oder komplexe Algorithmen implementieren. Um diese Herausforderungen zu bewältigen, ist es von entscheidender Bedeutung, dass alle Programmierer dokumentieren, was sie getan haben und wie der Code funktioniert. Durch eine klare Dokumentation kann die Zusammenarbeit erleichtert werden. Ein weiteres Problem, dem wir begegnet sind, ist die Sprachbarriere oder kulturelle Unterschiede zwischen den Programmierern. Es ist wichtig, klare und prägnante Kommentare zu schreiben, um Missverständnisse zu vermeiden und sicherzustellen, dass alle Teammitglieder auf derselben Seite sind.

Im Endeffekt erfordert das gemeinsame Programmieren Zusammenarbeit und offene Kommunikation, um sicherzustellen, dass alle Teammitglieder effektiv und effizient am Code arbeiten können. Eine gute Dokumentation und gute Kommentare sind notwendig, um die Zusammenarbeit zu erleichtern und sicherzustellen, dass der Code reibungslos implementiert wird.

Obwohl das gemeinsame Programmieren seine Herausforderungen mit sich bringt, hat es auch viele Vorteile für unsere Programmiergruppe gebracht. Vorteile sind die Möglichkeit, Ideen auszutauschen und gemeinsam an Lösungen zu arbeiten. Durch das gemeinsame Programmieren konnten wir effektivere Lösungen entwickeln, indem wir das Wissen und die Fähigkeiten jedes Teammitglieds einbezogen haben.

Darüber hinaus hat das gemeinsame Programmieren unsere Zusammenarbeit und Kommunikation verbessert, indem wir unsere Arbeitsprozesse koordiniert und besser organisiert haben. Wir haben Methoden entwickelt, um sicherzustellen, dass der Code von jedem Mitglied des Teams verständlich ist, und haben eine umfassende Dokumentation erstellt, um das Verständnis des Codes zu erleichtern.

Insgesamt hat das gemeinsame Programmieren unser Team enger zusammengeschweisst und uns geholfen, effektiver und effizienter zu arbeiten. Wir sind in der Lage,

Herausforderungen gemeinsam zu bewältigen und schnellere Lösungen zu finden, indem wir unser Wissen und unsere Fähigkeiten kombinieren.

4.3 Einschränkungen

Eine Einschränkung des Spam Detectors ist, dass er nicht in der Lage ist, alle Spam E-Mails zu erkennen. Manche E-Mails könnten fälschlicherweise als Spam klassifiziert werden (False Positives), während andere Spam E-Mails unentdeckt bleiben könnten (False Negatives).

Die Leistung des Spam Detectors hängt sehr von der Qualität der verwendeten Trainingsdaten ab.

Aktuell beschränkt sich die Erkennung von Spam auf die in der Excel-Datei enthaltenen Begriffe. Eine Erweiterung auf eine grössere Datenbank oder die Integration von Machine Learning Algorithmen könnte die Erkennungsrate verbessern.

Insgesamt ist der Spam Detector eine nützliche Applikation, die dazu beiträgt, den E-Mail-Verkehr sicherer und effizienter zu gestalten. Es gibt jedoch noch Raum für Verbesserungen und Erweiterungen, um die Leistung und Genauigkeit zu steigern.

5 Schlussfolgerungen

Das Erstellen des Spam Detektors war für uns ein komplexes und zeitaufwendiges Projekt. Es umfasste mehrere Schritte einschliesslich Datensammlung Vorverarbeitung, Modellauswahl und -training sowie Evaluierung und Optimierung. Zudem ist es sehr unreal, dass unser Spam Detektor zukünftig alle Spam-E-mails erkennt, da Spam-Betreiber ständig neue Techniken entwickeln, um ihre Nachrichten durch Spam-Filter zu bringen und es fordert ständige Anpassungen und Updates, um den Detektor effektiv zu halten.

5.1 weiterführende Untersuchungsmöglichkeiten

1. Verbesserung der Klassifizierungsgenauigkeit

Man könnte die verwendeten Merkmale oder Algorithmen optimieren. Beispielsweise mit verschiedenen Machine-Learning-Modellen oder Feature-Selection-Techniken.

Auch der Datensatz könnte man erweitern oder verfeinern, um mehr unterschiedliche Spam- und Nicht-Spam-Beispiele zu haben auf die das Modell trainiert werden kann.

2. Multilinguale Klassifizierung

Der Spam Detektor kann momentan nur deutsche und wenige englische Begriffe.

Dazu könnte man einen multilingualen Datensatz erstellen und verschiedenen

Transfer-Learning-Techniken ausprobieren, um zu sehen, wie gut sich das Modell auf neue Sprachen anpassen kann.

3. Erkennung von Phishing und Malware-E-mails

Der Spam Detektor ist momentan nur auf Spam-E-mails trainiert. Daher könnte man untersuchen, wie gut er sich auf andere Arten von bösartigen E-mails anwenden lässt, wie beispielsweise Phishing- oder Malware-E-mails. Dazu müsste man einen Datensatz mit diesen Arten von E-mails erstellen und das Modell darauf trainieren, sie zu erkennen.

4. Integration in Email-Clients

Praktisch wäre, wenn der Spam Detektor in bestehende Email-Clients integriert werden könnte, um Benutzern automatisch dabei zu helfen, Spam-E-mails zu erkennen und zu filtern. Dazu müsste man eine Schnittstelle oder eine Plugin-Schnittstelle entwickeln, die es dem Spam Detektor ermöglicht, auf die E-mails zuzugreifen und seine Klassifizierungsergebnisse direkt im Benutzer anzuzeigen.

5.2 Wichtigste Erkenntnisse

1. Der Spam Detektor kann dabei helfen, die E-Mail-Sicherheit zu erhöhen.
2. Entwicklung von Machine-Learning-Modellen: Das Spam-Detektor-Projekt kann auch dazu beitragen, die Entwicklung von Machine-Learning-Modellen zu verbessern, die in der Lage sind, Spam-Nachrichten automatisch zu erkennen und zu filtern.
3. Es ist anstrengend in einer Gruppe zu programmieren, die Denkweise des einzelnen sind gut erkennbar am jeweiligen Code. Andererseits ist es des öfteren schwieriger den Code fehlerfrei zu interpretieren. Umso wichtiger ist es Kommentare gut zu formulieren.

6 Quellenverzeichnis

6.1 Java API

<https://openjfx.io/javadoc/18/help-doc.html>, (7. Mai 2023)

7 Literaturverzeichnis

Java in 14 Wochen, Kaspar Riesen, 2020

8 Abbildungsverzeichnis

Abbildung 1 Deckblatt.....	- 1 -
Abbildung 2 Übersicht Anwendung.....	- 4 -
Abbildung 3 SpamMVC main class.....	- 5 -
Abbildung 4 Model-View-Controller Pattern (Vorlesung 2, Folien Richards Bradley)	- 6 -
Abbildung 5 UML Klassendiagramm.....	- 8 -
Abbildung 6 Übersicht Package Explorer	- 8 -
Abbildung 7 Datenauswahldialog	- 9 -
Abbildung 8 checkSpam Methode	- 10 -
Abbildung 9 Ansicht von allen Buttons.....	- 10 -
Abbildung 10 Methode für den Doppelklick	- 11 -
Abbildung 11 EMailView mit einer Beispiel-E-Mail.....	- 12 -
Abbildung 12 Oberflächen Aufbau Applikation Spam Detector.....	- 22 -
Abbildung 13 Detaillierte Bereiche der HBOX, im Top Bereich sind die Buttons zu sehen / Liste der eingelesenen E-Mails.....	- 23 -
Abbildung 14 Anzeige, ob E-Mail Spam hat oder nicht, Delete Spam Button	- 23 -

9 Glossar

9.1 Spam-E-Mail

Eine Spam-E-Mail ist eine unerwünschte Nachricht, die in der Regel in grossen Mengen und ohne Zustimmung des Empfängers versendet wird. Sie enthält oft Werbung, betrügerische Angebote, unerlaubte Nachrichten oder Links zu schädlichem Inhalt. Spam-E-Mails können lästig sein und die E-Mail-Kommunikation stören.

9.2 Phishing E-Mails

Phishing-E-Mails sind E-Mails, die dazu verwendet werden, den Empfänger dazu zu verleiten, auf einen gefälschten Link zu klicken oder vertrauliche Informationen wie Passwörter, Benutzernamen oder Kreditkartennummern preiszugeben. Der Link oder die Webseite, auf die der Empfänger geleitet wird, sieht oft täuschend echt aus und ist dazu gedacht, den Empfänger dazu zu bringen, persönliche Informationen preiszugeben.

9.3 Malware

Malware ist eine Abkürzung für «malicious software» und beschreibt Software, die mit der Absicht geschrieben wurde, Schaden zu verursachen oder unerlaubte Zugriffe auf Computer oder Netzwerke zu ermöglichen. Beispiele für Malware sind Viren, Trojaner, Spyware und Ransomware.

9.4 Feature Selection Techniken

Feature Selection Techniken beziehen sich auf Methoden, die verwendet werden, um die Anzahl der Merkmale (features) in einem Datensatz zu reduzieren, während gleichzeitig die Genauigkeit eines Modells beibehalten oder verbessert wird. Diese Techniken sind wichtig, um Overfitting zu vermeiden und die Leistung von Machine-Learning-Modellen zu optimieren.

9.5 Overfitting

Overfitting ist ein Phänomen im Machine Learning, bei dem ein Modell zu stark an die Trainingsdaten angepasst wird und dadurch schlechte Vorhersagen auf neuen Daten liefert. Es tritt auf, wenn das Modell zu komplex ist oder zu lange trainiert wird. Overfitting kann vermieden werden, indem man das Modell vereinfacht, mehr Trainingsdaten verwendet oder Regularisierungstechniken einsetzt. Es ist wichtig, Overfitting zu erkennen und zu vermeiden, um eine gute Vorhersageleistung auf neuen Daten zu erzielen.

9.6 Machine Learning

Machine Learning ist eine Methode der künstlichen Intelligenz, bei der Computer-Algorithmen darauf trainiert werden, Muster in Daten zu erkennen und Vorhersagen auf Basis dieser Muster zu treffen. Es gibt drei Hauptarten von Machine-Learning: überwachtes Lernen, unüberwachtes Lernen und verstärkendes Lernen.

9.7 Algorithmen

Algorithmen sind eine Reihe von Anweisungen oder Regeln, die von einem Computer ausgeführt werden, um eine bestimmte Aufgabe auszuführen. Im Kontext von Machine-Learning bezieht sich der Algorithmus auf den Satz von Regeln, die vom Computer befolgt werden, um Muster in Daten zu erkennen und Vorhersagen zu treffen. Ein Beispiel für einen Machine-Learning-Algorithmus ist der Random Forest-Algorithmus.

9.8 CSV- File

Eine CSV (Comma-Separated Values)-Datei ist ein Textdateiformat, das zur Speicherung und Übertragung von tabellarischen Daten verwendet wird. Wie der Name schon sagt, sind die Daten in der Datei durch Kommas getrennt. Jede Zeile in der Datei entspricht einem Datensatz, und die Spalten in der Zeile repräsentieren die Felder der Daten. CSV-Dateien sind plattformunabhängig und können in jeder Textverarbeitungs- oder Tabellenkalkulationssoftware geöffnet werden. Sie sind auch einfach zu generieren und zu verarbeiten, was sie zu einem beliebten Format für den Austausch von Daten zwischen Anwendungen macht.

9.9 EML

Eine EML-Datei ist ein Dateiformat für E-Mails, das von verschiedenen E-Mail-Clients wie Microsoft Outlook, Mozilla Thunderbird, Apple Mail und anderen verwendet wird. Eine EML-Datei enthält die vollständige Nachricht, einschliesslich Absender, Empfänger, Betreff, Text und gegebenenfalls Anhänge. EML-Dateien können einfach per E-Mail verschickt oder auf dem lokalen Computer gespeichert werden. Sie können auch von anderen E-Mail-Clients oder -Programmen geöffnet und gelesen werden, die EML-Dateien unterstützen. In der Regel wird eine EML-Datei verwendet, um eine einzelne E-Mail zu speichern, aber sie kann auch mehrere E-Mails enthalten, die als Anhang hinzugefügt wurden.

9.10 HTML

HTML steht für Hypertext Markup Language und ist eine Auszeichnungssprache, die zur Erstellung von Webseiten verwendet wird. Es dient dazu, den Inhalt einer Webseite strukturiert zu markieren und somit für den Webbrowser interpretierbar zu machen.

HTML besteht aus sogenannten Tags, die in spitzen Klammern angegeben werden und den Text innerhalb dieser Klammern mit einer Bedeutung versehen. Zum Beispiel wird der Text innerhalb des Tags **<h1>** als Überschrift erkannt und entsprechend dargestellt.

HTML ermöglicht es auch, auf andere Dateien wie Bilder, Videos und Stylesheets zu verweisen und sie in die Webseite einzubinden. Durch die Verwendung von CSS (Cascading Style Sheets) können HTML-Elemente auch gestaltet werden, um das Aussehen und das Layout der Webseite zu verbessern.

Insgesamt ist HTML eine grundlegende Technologie im World Wide Web und bildet zusammen mit CSS und JavaScript die Basis für die meisten modernen Webseiten.

9.11 CSS

CSS (Cascading Style Sheets) ist eine Sprache zur Gestaltung von Webseiten. Sie wird verwendet, um das Aussehen und das Layout einer Webseite zu definieren und zu kontrollieren. Mit CSS können Sie Elemente wie Schriftarten, Farben, Abstände, Ausrichtungen und vieles mehr auf einer Webseite formatieren und gestalten.

9.12 Instanzvariable

Eine Instanzvariable ist eine Variable, die in einer Klasse definiert ist und für jedes Objekt dieser Klasse unterschiedliche Werte haben kann. Sie wird typischerweise in einer Klasse definiert, um Daten zu speichern, die für alle Methoden dieser Klasse zugänglich sein sollen.

9.13 Konstruktor

Ein Konstruktor ist eine besondere Methode, die bei der Erstellung eines Objekts aufgerufen wird. Sie wird verwendet, um die Eigenschaften und den Zustand eines Objekts zu initialisieren und zu konfigurieren. Ein Konstruktor hat denselben Namen wie die Klasse und kann Parameter enthalten, die beim Erstellen eines Objekts übergeben werden.

9.14 GUI

Eine GUI (Graphical User Interface) ist eine grafische Benutzeroberfläche, die verwendet wird, um mit einem Computerprogramm zu interagieren. Es besteht aus einer Sammlung von visuellen Elementen wie Schaltflächen, Textfeldern, Menüs und Fenstern, die es dem Benutzer ermöglichen, Aktionen auszuführen und Informationen anzuzeigen. GUIs werden

typischerweise mit einer Programmiersprache und einer GUI-Bibliothek wie Java Swing oder Qt erstellt.

9.15 JAR

Eine JAR-Datei (Java Archive File) ist ein komprimiertes Archivformat, das dazu verwendet wird, eine oder mehrere Java-Klassen, Ressourcen, Bibliotheken und andere Dateien zu verpacken. Es ist ein Standardformat für die Verteilung von Java-Anwendungen und Bibliotheken, da es alle erforderlichen Klassen und Ressourcen in einer einzigen Datei enthält.

Eine JAR-Datei kann auf verschiedene Arten erstellt werden, z.B. mit dem JDK-Befehl "jar" oder mit integrierten Tools von Entwicklungsumgebungen wie Eclipse oder IntelliJ IDEA. Um eine JAR-Datei auszuführen, muss die Java Virtual Machine (JVM) auf dem Zielcomputer installiert sein. Die JAR-Datei kann dann ausgeführt werden, indem man das Kommando "java -jar <jar-datei>" in der Befehlszeile ausführt.

JAR-Dateien können auch als Bibliotheken verwendet werden, indem sie in das "classpath" des auszuführenden Programms aufgenommen werden. Das bedeutet, dass die Klassen und Ressourcen in der JAR-Datei von anderen Java-Anwendungen verwendet werden können, die auf die Bibliothek zugreifen müssen.

10 Anhang

10.1 Erste Logik Ideen

- disable addBtn wenn Tabelle voll
- disable deleteBtn, deleteAllBtn, checkBtn, deleteSpamBtn wenn Tabelle leer
- disable deleteSpamBtn wenn kein Spam gefunden
- SpamLbl soll anzeigen wie viel Spam Mails gefunden wurde, bei 0 Text grün, bei >0 Text rot z.B. „0 Spam E-Mails found“
- addBtn soll E-Mail aus einer Datenbank in table einfügen
- In Table unterscheiden zwischen First name, Last name, E-Mail Adress und subject
- iterieren durch table soll möglich sein, um einzelne Mails mit deleteBtn zu löschen
- deleteAllBtn soll table leeren
- checkBtn soll alle E-Mails in der Tabelle nach Spambergriffen durchsuchen -> Spambegriffe festlegen (z.B. eigene Klasse, um mit Setter neue Begriffe hinzuzufügen)
- table Inhalt rot wenn Spam, grün wenn kein Spam, in SpamCln anzeigen ob Spam
- deleteSpamBtn soll nur Spammails aus table löschen

10.2 Erster Entwurf

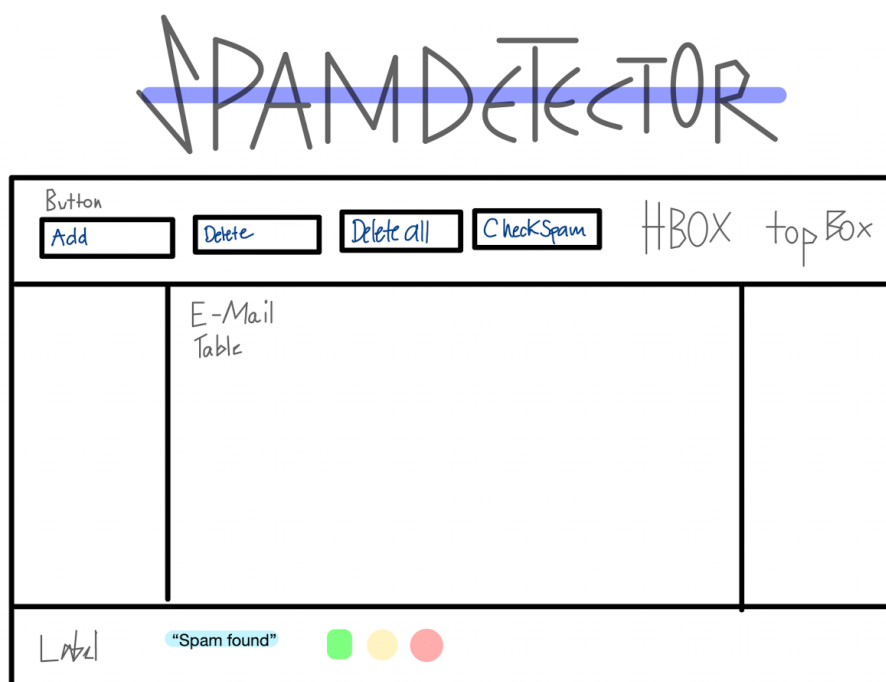
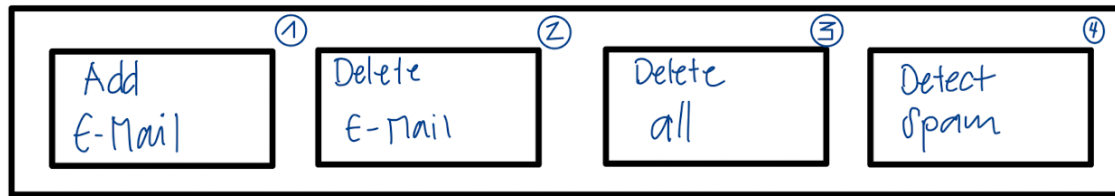


Abbildung 12 Oberflächen Aufbau Applikation Spam Detector

TOP BOX

HBOX



Center Table E-Mails

FName	LName	E-Mail	Subject	SPAM
✓	✓	✓	✓	NO SPAM
✓	✓	✓	✓	SPAM

Abbildung 13 Detaillierte Bereiche der HBOX, im Top Bereich sind die Buttons zu sehen / Liste der eingelesenen E-Mails

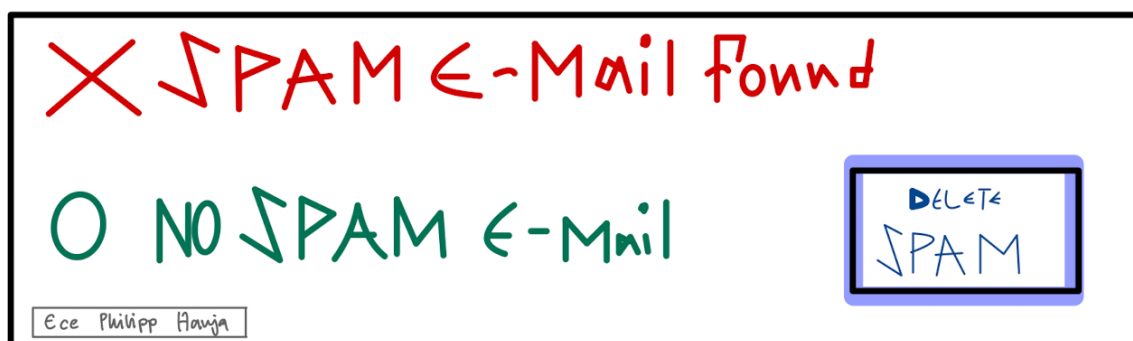
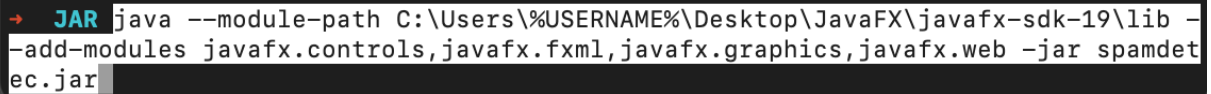


Abbildung 14 Anzeige, ob E-Mail Spam hat oder nicht, Delete Spam Button

1.1 Ausführen als JAR Datei

Um den Spam Detector als JAR Datei auszuführen, muss man im CMD / Terminal, die folgenden Arguments mitgegeben werden.

Bei Windows sieht das wie folgt aus (im Beispiel ist die JAR Datei auf dem Desktop gespeichert worden)



```
→ JAR java --module-path C:\Users\%USERNAME%\Desktop\JavaFX\javafx-sdk-19\lib --  
-add-modules javafx.controls,javafx.fxml,javafx.graphics,javafx.web -jar spamdet  
ec.jar
```