Projekt VaTra Gruppe 8

**Autoren**  
Michael Räss, Tobias Schmoker, David Wiedmer

**Auftraggeber**  
Joachim Wolfgang Kaltz

Berner Fachhochschule – BTI703 - Projekt 1   
20. Januar 2016

Inhalt

[1. Einleitung 4](#_Toc441248580)

[2. Projektziele 4](#_Toc441248581)

[Hauptziel 4](#_Toc441248582)

[Teilziele 4](#_Toc441248583)

[3. System und Systemkontext 5](#_Toc441248584)

[4. System Akteure und Elemente 5](#_Toc441248585)

[5. Rahmenbedingungen 5](#_Toc441248586)

[6. Anforderungen 6](#_Toc441248587)

[Funktionale Anforderungen 6](#_Toc441248588)

[Detailbeschrieb 8](#_Toc441248589)

[Nicht-Funktionale Anforderungen 11](#_Toc441248590)

[7. UML-Klassendiagramm (Gesamtbild) 12](#_Toc441248591)

[8. Erklärung Klassendiagramme 13](#_Toc441248592)

[Aufbau Konfiguration Spring Security 13](#_Toc441248593)

[Aufbau Request Handling 14](#_Toc441248594)

[Übersicht Entitäten (ORM) 15](#_Toc441248595)

[Übersicht Algorithmen 17](#_Toc441248596)

[9. UML-Sequenzdiagramm („Geo“ Algorithmus) 18](#_Toc441248597)

[10. Mockups 19](#_Toc441248599)

[Login & Registrierung 19](#_Toc441248600)

[Dashboard 19](#_Toc441248601)

[Schema 20](#_Toc441248602)

[Algorithmen 20](#_Toc441248603)

[Algorithmen 21](#_Toc441248604)

[Statistik 21](#_Toc441248605)

[API 22](#_Toc441248606)

[11. Verwendete Hilfsmittel (Software) 23](#_Toc441248607)

[12. Eingesetzte Technologien 23](#_Toc441248608)

[13. Ausarbeitungsmöglichkeiten 24](#_Toc441248609)

[14. Journal 25](#_Toc441248610)

[Termine 25](#_Toc441248611)

[23.09.2015 - Meeting 1 mit Herr Kaltz 25](#_Toc441248612)

[01.10.2015 - Workshop im Projektteam 25](#_Toc441248613)

[05.10.2015 - Meeting 2 mit Herr Kaltz 25](#_Toc441248614)

[19.10.2015 - Meeting 3 mit Herr Kaltz 25](#_Toc441248615)

[02.11.2015 - Meeting 4 mit Herr Kaltz 25](#_Toc441248616)

[18.11.2015 - Meeting 5 mit Herr Kaltz 25](#_Toc441248617)

[08.12.2016 - Meeting 6 mit Herr Kaltz 25](#_Toc441248618)

[06.01.2016 - Meeting 7 mit Herr Kaltz 25](#_Toc441248619)

[13.01.2016 - Meeting 8 mit Herr Kaltz 26](#_Toc441248620)

[22.01.2016 – Abgabe 26](#_Toc441248621)

[Arbeiten 26](#_Toc441248622)

[23.09.2015 – 05.10.2015 26](#_Toc441248623)

[06.10.2015 – 19.10.2015 26](#_Toc441248624)

[20.10.2015 – 02.11.2015 26](#_Toc441248625)

[03.10.2015 – 18.11.2015 26](#_Toc441248626)

[19.11.2015 – 08.12.2015 26](#_Toc441248627)

[09.12.2015 – 06.01.2016 26](#_Toc441248628)

[07.01.2016 – 22.01.2016 27](#_Toc441248629)

[22.01.2016+ 27](#_Toc441248630)

[15. Erkenntnisse 28](#_Toc441248631)

[Problem beim Geo-Algorithmus 28](#_Toc441248632)

[Weiterleitung von Formulardaten an Drittsystem 28](#_Toc441248633)

[16. Glossar 29](#_Toc441248634)

# Einleitung

Bei der Durchführung einer Onlinezahlung besteht immer ein Risiko. Etwa durch harmlose Fehleingabe, bedingter Manipulation, Verwendung von gestohlenen Zahlungsinformationen oder auch Phishing Seiten.

„VaTra“ übernimmt die Validierung der Transaktionen, indem sie die Zahlungsinformationen überprüft und mit ausgeklügelten Algorithmen überprüft. Die Empfindlichkeit der Validierung lässt sich pro Kunde einzeln konfigurieren.

# Projektziele

## Hauptziel

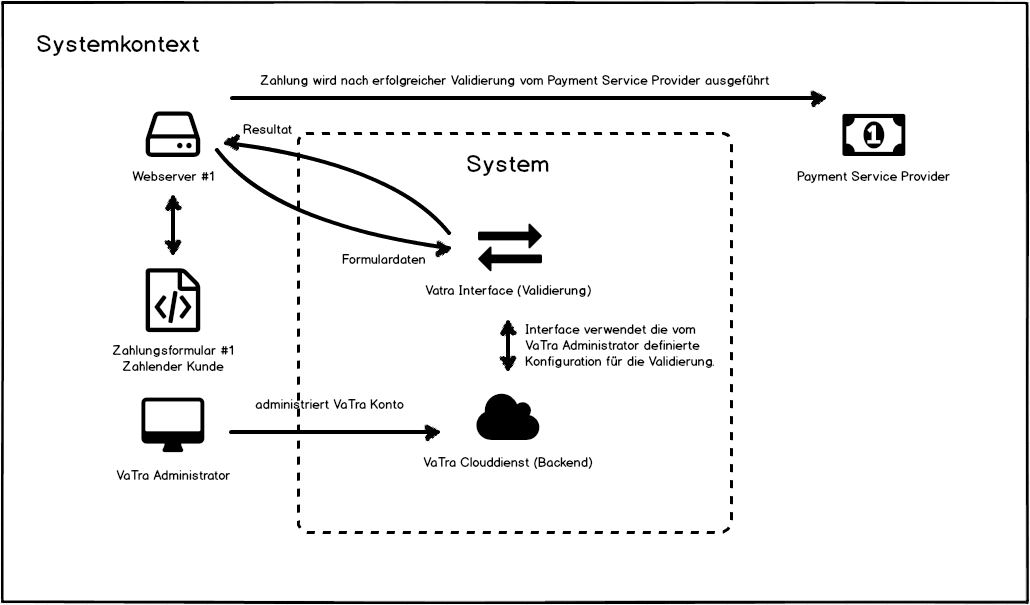
Das Hauptziel der Lösung ist die Erstellung eines Prototyps eines Webdienstes, der die Durchführung von Zahlungsvorgängen für Drittsysteme sicherer macht.

## Teilziele

|  |  |
| --- | --- |
| Nr. | Unterziel |
| 1 | Dritte können mithilfe des Systems die **Sicherheit** ihrer Zahlungsformulare verbessern. |
| 2 | Das System verfügt über eine intuitive **Benutzeroberfläche**, mit welcher ein Laien-Benutzer zurechtkommt. |
| 3 | Der **Datenschutz** wird eingehalten, indem keine Formulardaten in der Datenbank gespeichert werden. |

# System und Systemkontext

Das System beinhaltet den VaTra Clouddienst sowie das VaTra Interface. Als Quellen dienen Eingabemasken von existierenden Softwaresystemen. In nachfolgender Abbildung wird der Systemkontext mit allen Akteuren grafisch dargestellt.



# System Akteure und Elemente

|  |  |
| --- | --- |
| Akteur / Element | Beschreibung |
| Zahlender Kunde | Der zahlende Kunde ist die Person, welche irgendein durch VaTra sichergestelltes Zahlungsformular verwendet. |
| VaTra Administrator | Der VaTra Administrator ist die Person, welche den VaTra Clouddienst verwendet und über dessen Backend Sicherheitskonfigurationen vornimmt. |
| Webserver | Empfängt Zahlungsdaten und kommuniziert mit VaTra Interface. |
| VaTra Interface | Prüft, ob eine Zahlung sicher ist. |
| VaTra Clouddienst | Stellt ein cloudbasiertes Backend für Sicherheitskonfigurationen zur Verfügung. |

# Rahmenbedingungen

* Das Projekt finden im Rahmen des Modules BTI7301 an der Berner Fachhochschule statt und muss bis am 22.01.2016 abgeschlossen sein.
* Das Projekt muss in einem Team realisiert werden.

# Anforderungen

## Funktionale Anforderungen

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Nr. | Bezeichnung | P[[1]](#footnote-1) | K[[2]](#footnote-2) | V[[3]](#footnote-3) | R[[4]](#footnote-4) | Quelle | Ziel |
| A 1 | **Systemaufbau** |  |  |  |  |  |  |
| A.1.1 | **Modularer Aufbau** | 2 | 1 | 1 | 1.2 | Team | 1 |
| A 1.1.1 | Einfache Erweiterung | 2 | 1 | 1 | 1.2 | Team | 1 |
| A 1.2 | **Schnittstellen** | 3 | 2 | 2 | 2.2 | Team | 1 |
| A 1.2.1 | Alle Requests werden empfangen und verarbeitet | 3 | 2 | 2 | 2.2 | Team | 1 |
| A 1.1 | Das für die Schnittstelle verwendete Paradigma soll REST sein | 1 | 1 | 2 | 1.3 | Team | 1 |
| A 2 | **Authentifizierung** |  |  |  |  |  |  |
| A 2.1 | **Accounts** | 3 | 3 | 2 | 2.7 | Team | 2 |
| A 2.1.1 | Login in die Verwaltung | 3 | 3 | 2 | 2.7 | Team | 2 |
| A 2.1.2 | Registrierung eines neuen Accounts | 3 | 2 | 2 | 2.2 | Team | 2 |
| A 3 | **Verwaltung** |  |  |  |  |  |  |
| A 3.1 | **Apps** | 3 | 2 | 3 | 2.5 | Team | 2 |
| A 3.1.1 | Dashboard einsehen | 3 | 1 | 1 | 1.4 | Team | 2 |
| A 3.1.2 | Hinzufügen von Apps | 3 | 2 | 1 | 2 | Team | 2 |
| A 3.1.3 | Entfernen von Apps | 3 | 1 | 1 | 1.4 | Team | 2 |
| A 3.1.4 | App ist wählbar | 3 | 2 | 3 | 2.5 | Team | 2 |
| A 3.2 | **Schemas** | 2 | 2 | 2 | 2 | Team | 2 |
| A 3.2.1 | Bearbeiten des Schemas einer App | 2 | 2 | 2 | 2 | Team | 2 |
| A 3.3 | **Statistiken** | 1 | 1 | 2 | 1.4 | Team | 2 |
| A 3.3.1 | Statistik über Requests ist einsehbar | 1 | 1 | 2 | 1.4 | Team | 2 |
| A 3.3.2 | Statistik zu einzelnen Algorithmen ist einsehbar | 1 | 1 | 2 | 1.4 | Team | 2 |
| A 3.4 | **API Zugang** | 1 | 2 | 2 | 1.9 | Team | 1 |
| A 3.4.1 | API Key ist einsehbar und kann regeneriert werden | 1 | 1 | 1 | 1 | Team | 1 |
| A 3.4.2 | Bearbeitung der Server-Whitelist | 1 | 2 | 2 | 1.9 | Team | 1 |
| A 3.5 | **Algorithmen** | 3 | 2 | 2 | 2.2 | Team | 1 |
| A 3.5.1 | Mindestens 3 Algorithmen vorhanden | 3 | 2 | 2 | 2.2 | Team | 1 |
| A 3.5.2 | Ein- und Ausschalten von Algorithmen | 2 | 2 | 1 | 1.7 | Team | 1 |
| A 3.5.3 | Bearbeitung der Algorithmen-Tolleranz | 2 | 2 | 1 | 1.7 | Team | 1 |
| A 4 | **Verfügbarkeit** |  |  |  |  |  |  |
| A 4.1 | **Zeitnahe Verfügbarkeit** | 1 | 2 | 3 | 2.2 | Team | 1 |
| A 4.1.1 | Die Algorithmen müssen innert vernünftiger Zeit eine Antwort liefern | 1 | 2 | 3 | 2.2 | Team | 1 |
| A 4.2 | **Datenansicht** | 3 | 2 | 2 | 2.2 | Team | 2 |
| A 4.2.1 | Nur Account zugehörige Daten können eingesehen werden | 3 | 2 | 2 | 2.2 | Team | 2 |

|  |  |  |  |
| --- | --- | --- | --- |
| Detailbeschrieb | | | |
| A Nr. | | | **Beschrieb** |
| 1 |  |  | **Systemaufbau** |
|  | **1.1** |  | **Modularer Aufbau** |
|  |  | 1.1.1 | Das System soll objektorientiert aufgesetzt werden, um eine spätere Erweiterung zu vereinfachen.  Besonders Modular müssen die Algorithmen umgesetzt werden. Diese sollen unabhängig voneinander agieren. Das implementieren eines Algorithmus soll ohne Systemkenntnisse als Java Programmierer möglich sein. |
|  | **1.2** |  | **Schnittstelle** |
|  |  | 1.2.1 | Jeder Request, ob gültig oder ungültig, soll in der H2 Datenbank vermerkt und protokolliert werden.  Gespeichert werden sollen - Client Informationen - zugehörige App  Formulareingaben dürfen nicht gespeichert werden. |
|  |  | 1.2.2 | Das für die Schnittstelle verwendete Paradigma soll REST sein. |
| 2 |  |  | **Authentifizierung** |
|  | **2.1** |  | **Accounts** |
|  |  | 2.1.1 | Mit einer im System vorhandenen E-Mail-Adresse und dazugehörigem Passwort soll sich ein Anwender in der Verwaltung anmelden können.  Ein angemeldeter Anwender soll sich per Knopfdruck wieder vom System abmelden können. |
|  |  | 2.1.2 | Ein Anwender soll mit einer E-Mail-Adresse und durch doppelte Eingabe des Passworts einen Account erstellen können. Das Passwort soll dabei aus mindestens 8 Zeichen bestehen. |
| 3 |  |  | **Verwaltung** |
|  | **3.1** |  | **Apps** |
|  |  | 3.1.1 | Auf dem Dashboard soll ein Anwender eine Übersicht über alle seine Apps finden. |
|  |  | 3.1.2 | Der Anwender soll seinem Account Apps hinzufügen können. Der Anwender soll eine neue App hinzufügen können, indem er einen Domainnamen eingibt und die Eingabe bestätigt. |
|  |  | 3.1.3 | Apps sollen mit all ihren Einstellungen entfernt werden. |
|  |  | 3.1.4 | Der Anwender soll überall auf der Plattform, solange er mit einem Account angemeldet ist und mindestens eine App hinzugefügt hat, zu den Einstellungen einer App springen können. |
|  | **3.2** |  | **Schema** |
|  |  | 3.2.1 | Der Anwender soll dem System mitteilen, wie ein Formular zu einem spezifischen App aussieht. Der Aufbau des Formulars (Schema) kann eingesehen und bearbeitet werden. Das Schema wird direkt im JSON Datenformat bearbeitet. |
|  | **3.3** |  | **Statistiken** |
|  |  | 3.3.1 | Der Anwender soll folgende Informationen zur ausgewählten App einsehen können:   * Anzahl Requests * Anzahl gültiger Requests * Anzahl ungültiger Requests   + Ungültig infolge Algorithmus   + Ungültig infolge Standardvalidierung |
|  |  | 3.3.2 | Der Anwender soll folgende Statistik Informationen zu den einzelnen Algorithmen erhalten:   * Anzahl gütliger Requests pro Algorithmus * Anzahl ungültiger Requests pro Algorithmus |
|  | **3.4** |  | **API Zugang** |
|  |  | 3.4.1 | Der Anwender soll pro App einen API-Key einsehen und diesen per Knopfdruck regenerieren können. |
|  |  | 3.4.2 | Der Anwender soll die Whitelist in einem Textarea bearbeiten können. Jede Zeile ist ein einzelner Eintrag. Wenn keine Einträge in der Whitelist vorhanden sind, werden sämtliche Requests mit dem API-Key akzeptiert. |
|  | **3.5** |  | **Algorithmen** |
|  |  | 3.5.1 | Es sollen 3 Algorithmen implementiert werden:   1. Blacklist Algorithmus   Überprüft die Zahlungsteilnehmer anhand einer Blacklist.   1. Zahlungsintervall Überprüfung   Schätzt das Risikio anhand der Regelmässigkeit der Zahlungsaktionen.   1. Kreditkartenüberprüfung   Überprüfung der Kreditkartenangaben. |
|  |  | 3.5.2 | Der Anwender soll pro App auswählen können, welche Algorithmen er zur Validierung seines Formulars verwenden möchte. |
|  |  | 3.5.3 | Der Anwender soll pro App, mithilfe eines Sliders, die Toleranz der Algorithmen bearbeiten können. |
| 4 |  |  | **Verfügbarkeit** |
|  | **4.1** |  | **Zeitnahe Verfügbarkeit** |
|  |  | 4.1.1 | Bei einem Request auf die API soll das System innert 2 Sekunden ein Ergebnis liefern. |
|  | **4.2** |  | **Datenansicht** |
|  |  | 4.2.1 | Das System soll Zugriffberechtigungen zu angeforderten Daten überprüfen und soll diese nur zurückgeben, wenn entsprechende Berechtigung vorhanden ist. |

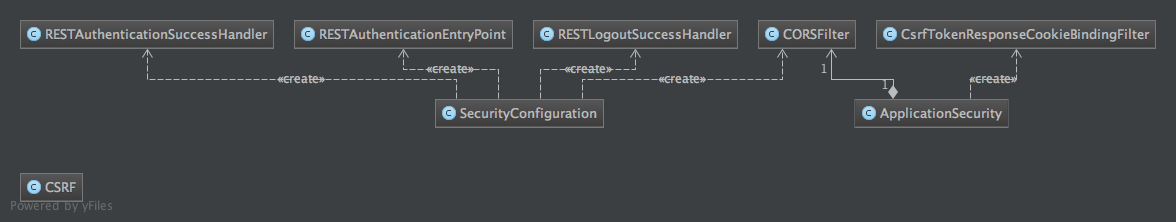
## Nicht-Funktionale Anforderungen

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Nr. | Bezeichnung | P | K | V | R | Quelle | Ziel |
| A 1 | **Systemaufbau** |  |  |  |  |  |  |
| A 1.1 | Einfache Wartbarkeit | 3 | 2 | 1 | 1.8 | Team | 1 |
| A 2 | **Schnittstelle** |  |  |  |  |  |  |
| A 2.2 | Unkomplizierte Integration in bestehende Systeme | 2 | 1 | 3 | 1.8 | Team | 1 |
| A 3 | **Verwaltung** |  |  |  |  |  |  |
| A 3.1 | Logischer und einfacher Aufbau | 3 | 2 | 2 | 2.2 | Team | 2 |
| A 3.2 | Eingabefelder erklärt, wenn nötig | 3 | 1 | 1 | 1.3 | Team | 2 |
| A 3.3 | Erkennung von Fehleingaben | 1 | 1 | 1 | 1 | Team | 2 |
| A 3.4 | Einheitliche Oberfläche | 3 | 1 | 1 | 1.3 | Team | 2 |
| A 3.5 | Benutzerfreundliche Darstellung der Informationen | 3 | 1 | 1 | 1.3 | Team | 2 |
| A 3.6 | Die Verwaltung muss in der jeweils neusten Version der Browser Internet Explorer, Firefox sowie Chrome funktionieren. | 3 | 1 | 1 | 1.3 | Team | 2 |

# VaTra Class Diagram (small)UML-Klassendiagramm (Gesamtbild)

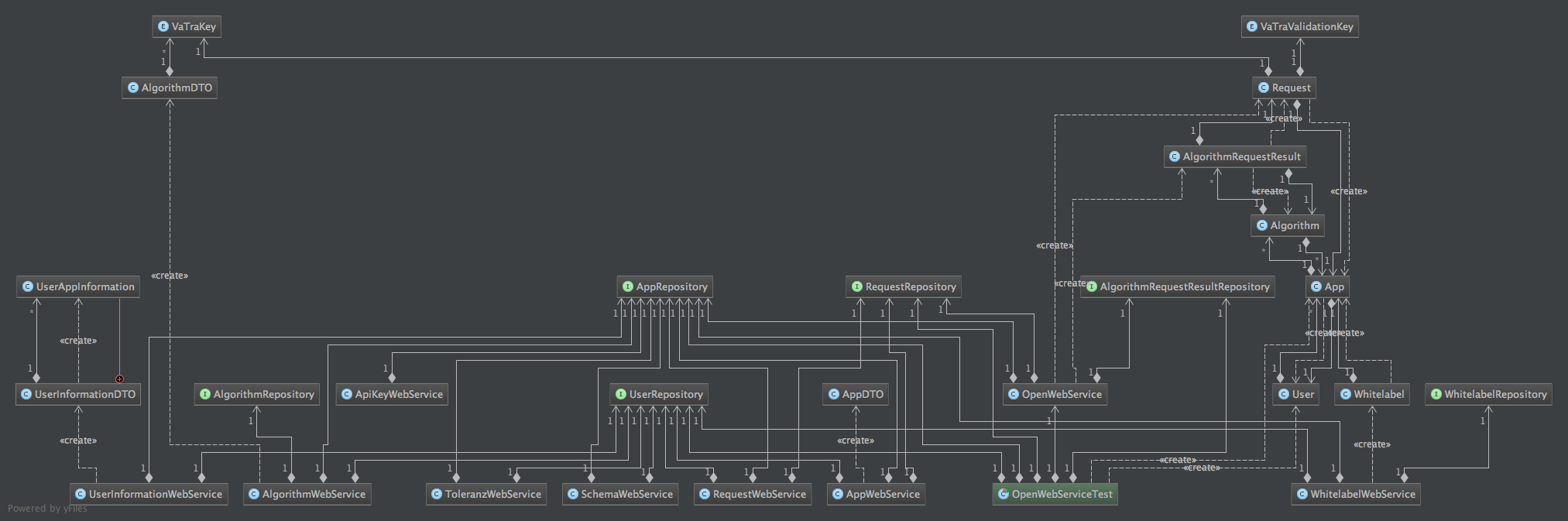
# Erklärung Klassendiagramme

## Aufbau Konfiguration Spring Security

Aufbau der Konfiguration für Spring Security.

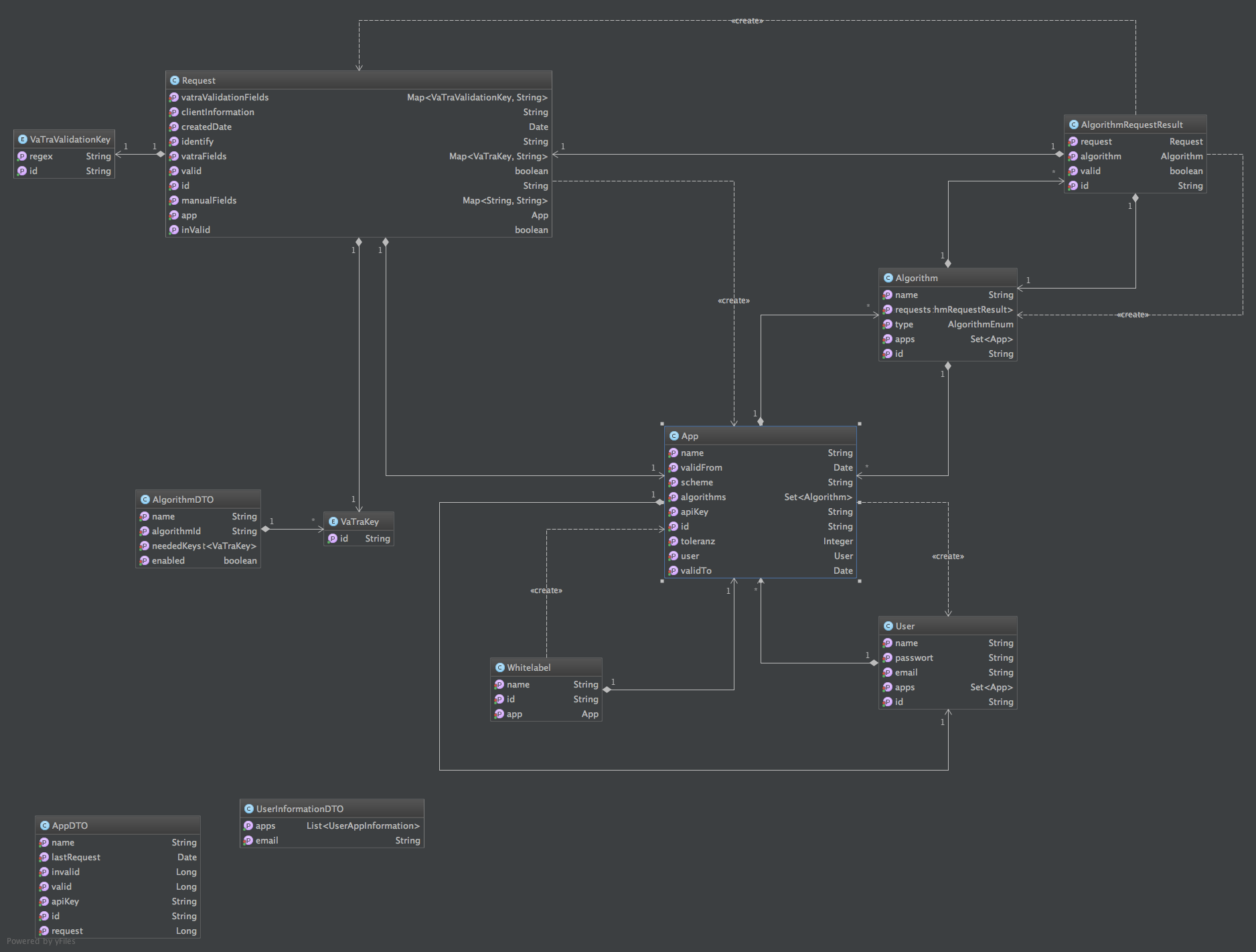
Aus diesem Diagramm ist ersichtlich wie die Konfiguration der Security aufgebaut ist. Es gibt die Security Configuration und Die Application Security. Während die Security Configuration verantwortlich ist für den Aufbau der RequestHandler sorgt sich die ApplicationSecurity um die Einbindung der definierten Filter.

## Aufbau Request Handling

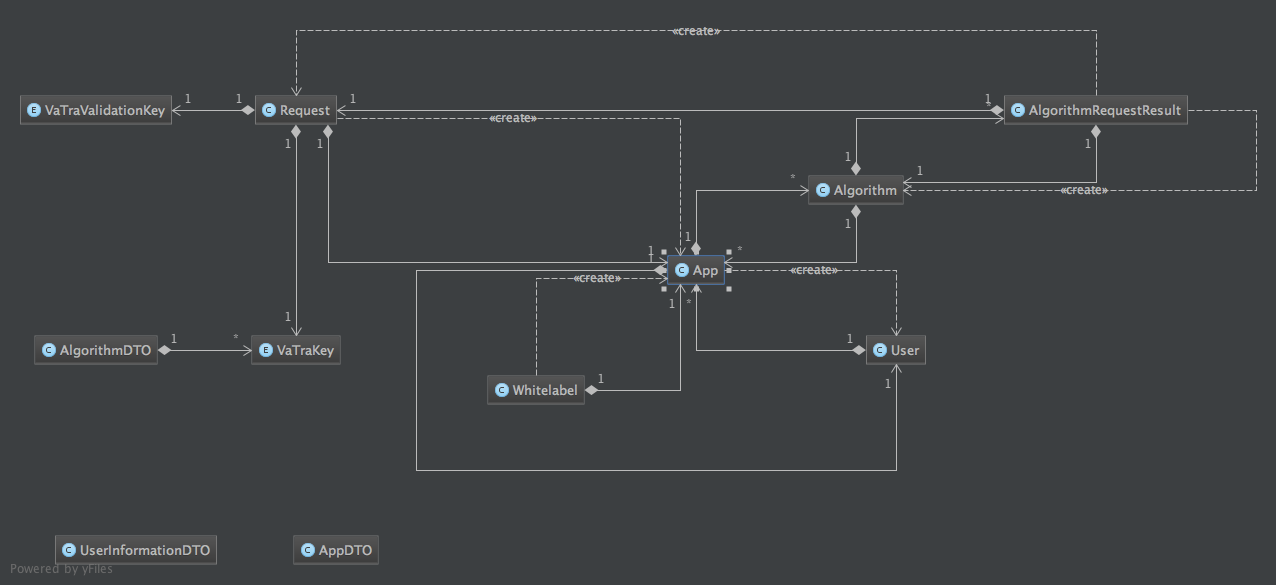


Aufbau des Request Handlings. Hier können wir aufzeigen wie die Abhängigkeiten der Controllen zu Ihren Datenzulieferern sind. Dabei geht auch hervor dass natrülich fast alle Services in gebraucht kommen um die Daten bei einem neuen FormularRequest aufgebraucht werden, während viele RestServices natürlich nur sehr beschränkt auf die Repositorys zugreiffen müssen. Der UserRepository-Service wird natürlich öffters gebraucht, da auch immer wieder der eingeloggte User erreicht werden muss.

## Übersicht Entitäten (ORM)

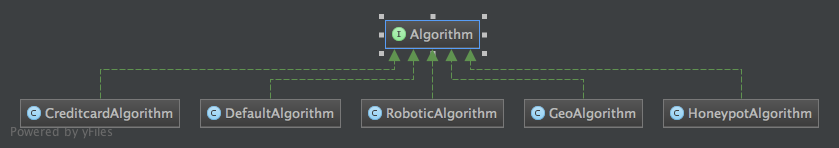


Hier sind die Entitäten mit Ihren Abhängigkeiten aufgeteilt. Dies Entspricht dem ERM der Datenbank. Die DTO sind hier auchdabei und zeigen keine direkte Verknüpfungen mit den Entitäten auf. (Ausser das AlgorithmDTO, dieses teilt sich für die Übertragung das Enum VaTraKey)



Hier ist dieselbe Grafik ohne Attribute zu finden.

## Übersicht Algorithmen

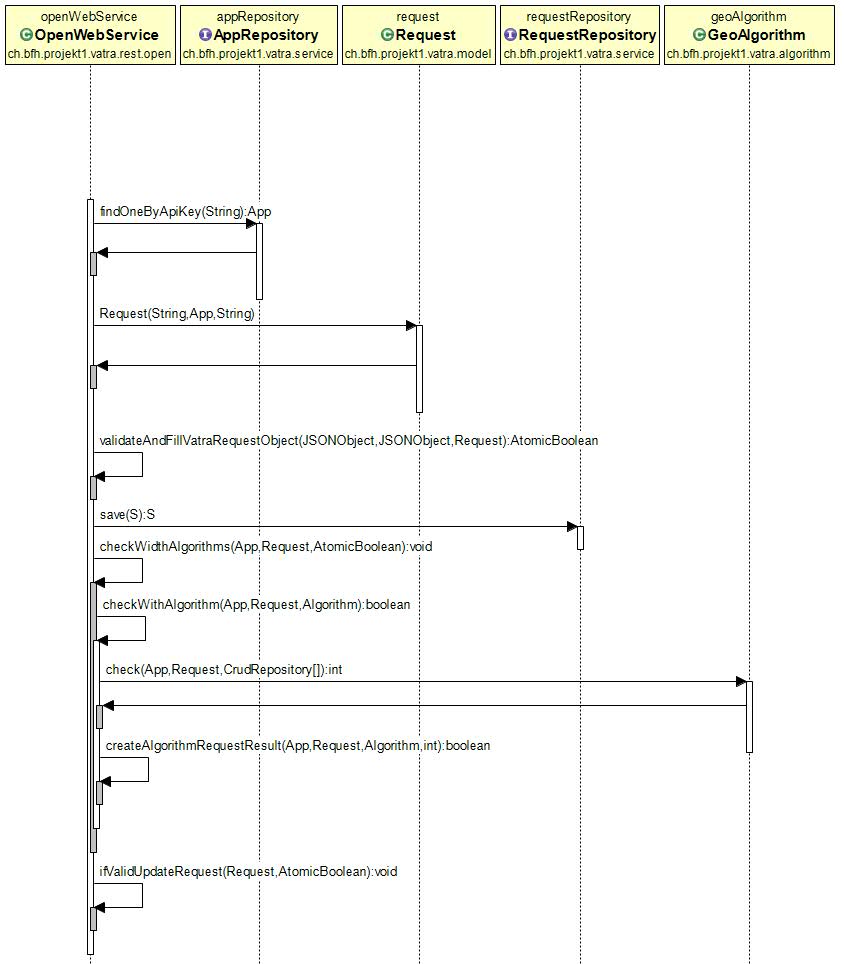


Diese Grafik zeigt auf, dass die Algorithmen alle das Interface Algorithm implementieren, welche 2 Methoden zu Verfügung stellt. Dies war nötig um eine Struktur in die Algorithmen zu bringen und diese etwas zu generischer zu gestalten.

# UML-Sequenzdiagramm („Geo“ Algorithmus)

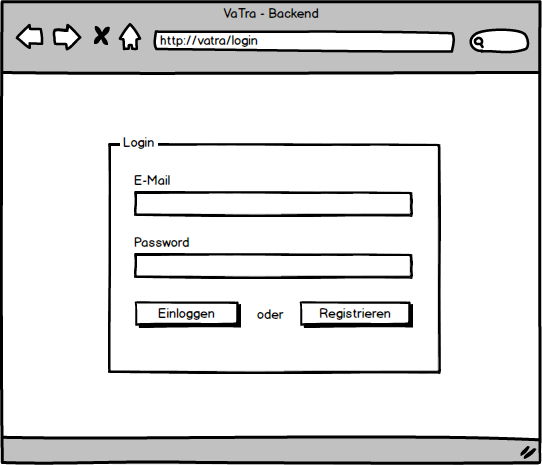
Folgendes Sequenzdiagramm zeigt den Ablauf einer Anfrage an das VaTra Interface für die Validierung einer „Zahlungsanfrage“. Dieses Sequenzdiagramm verwendet als Beispiel den Geo-Algorithmus. Im Normalfall werden alle aktivierten Algorithmen verwendet.

# 

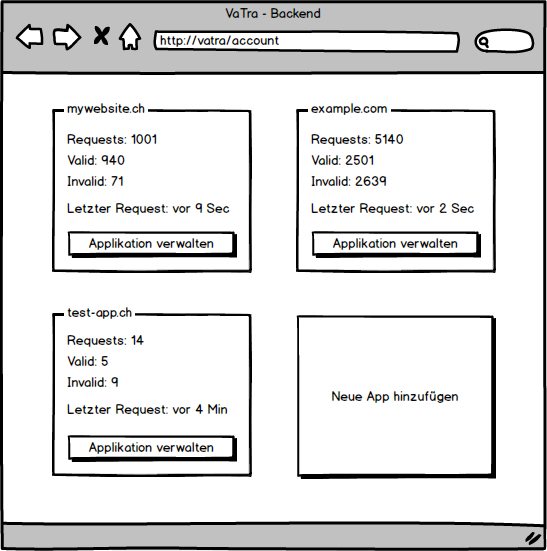


# Mockups

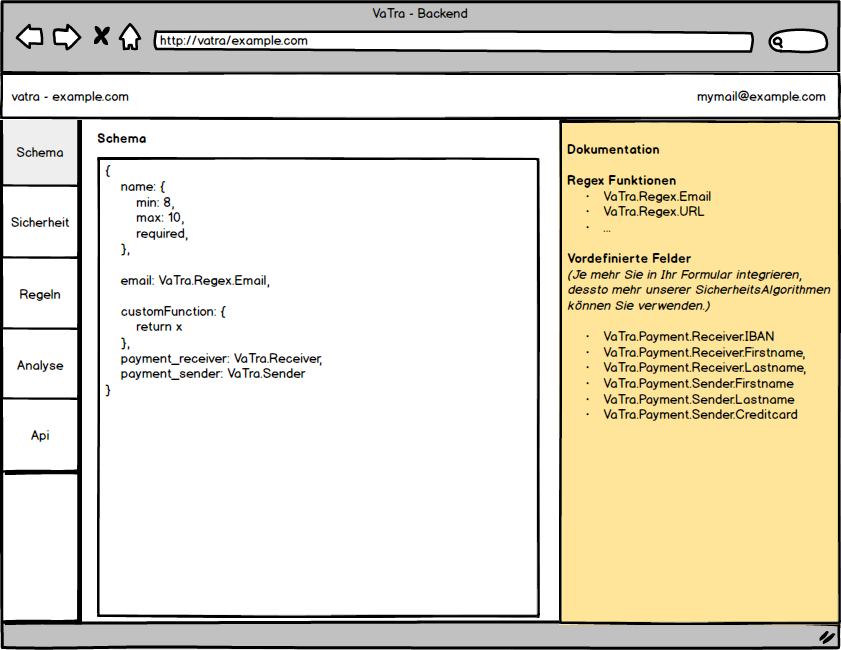
## Login & Registrierung



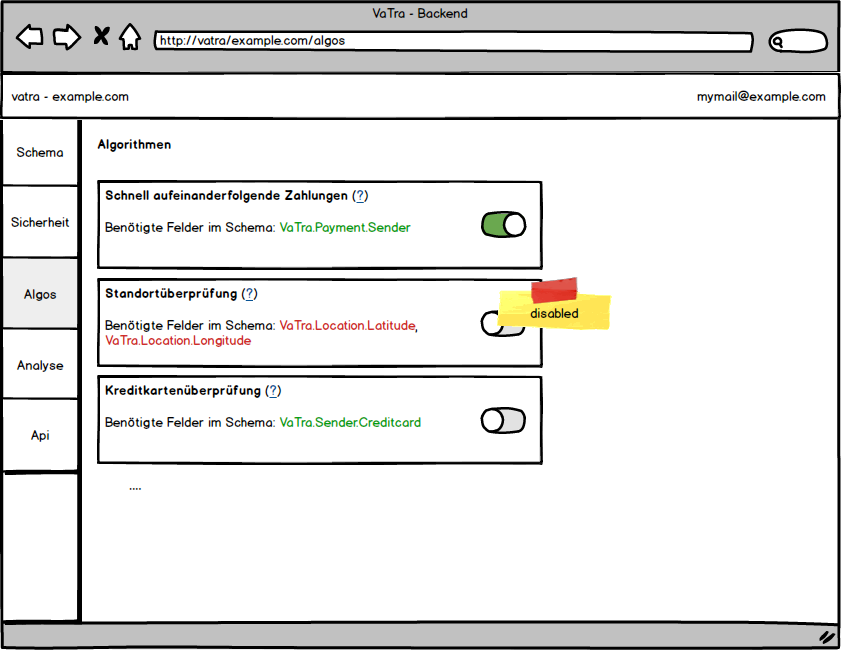
## Dashboard



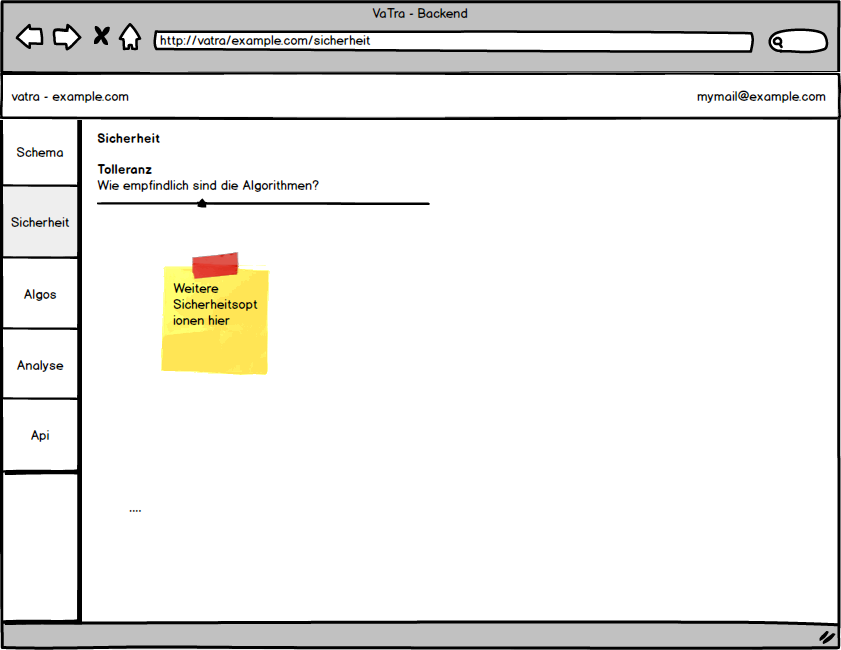
## Schema



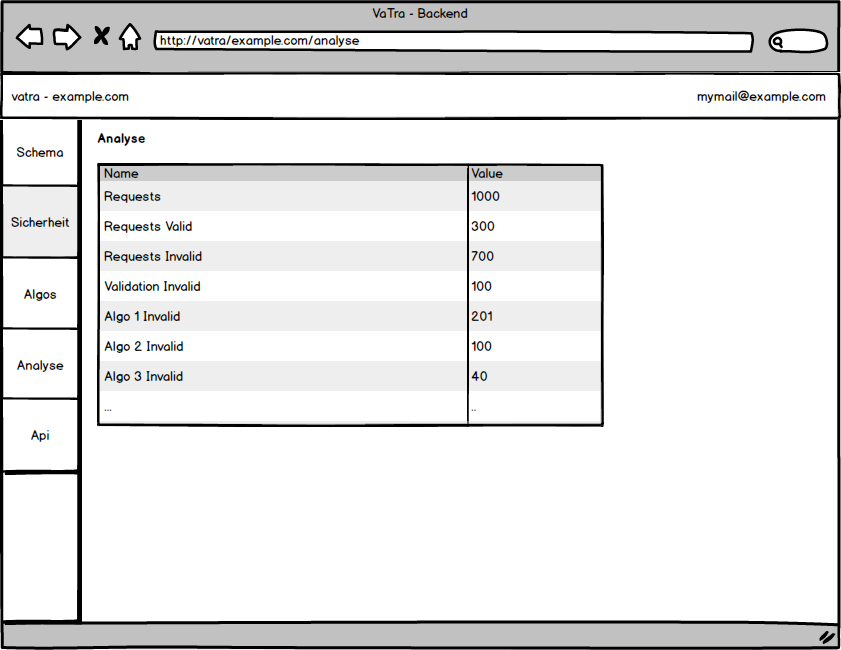
## Algorithmen



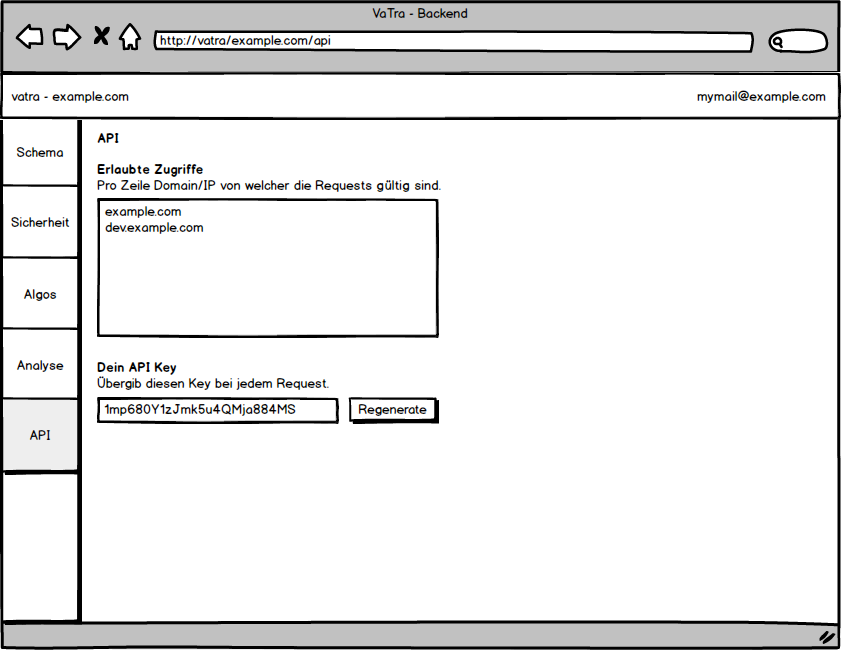
## Algorithmen



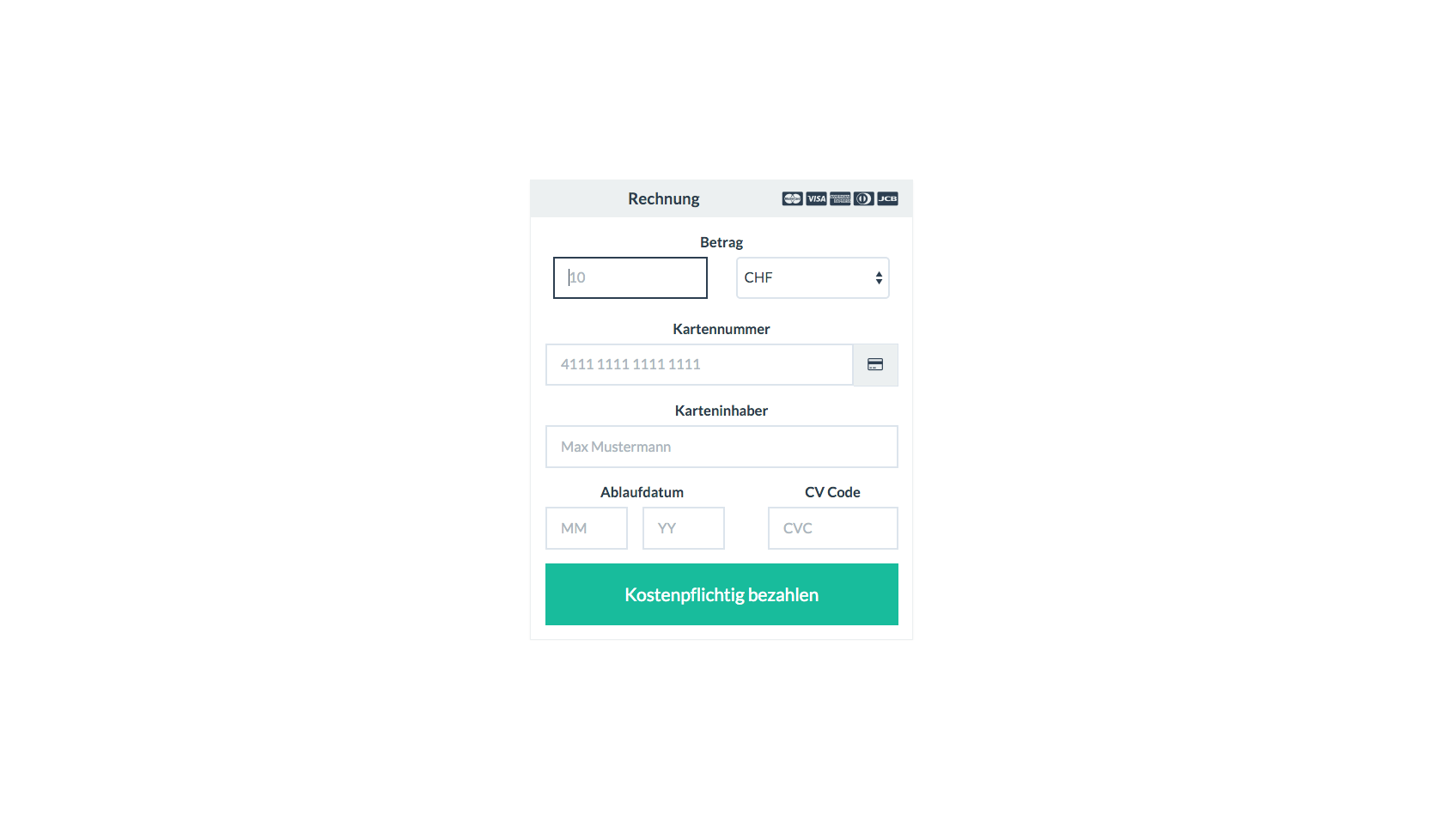
## Statistik

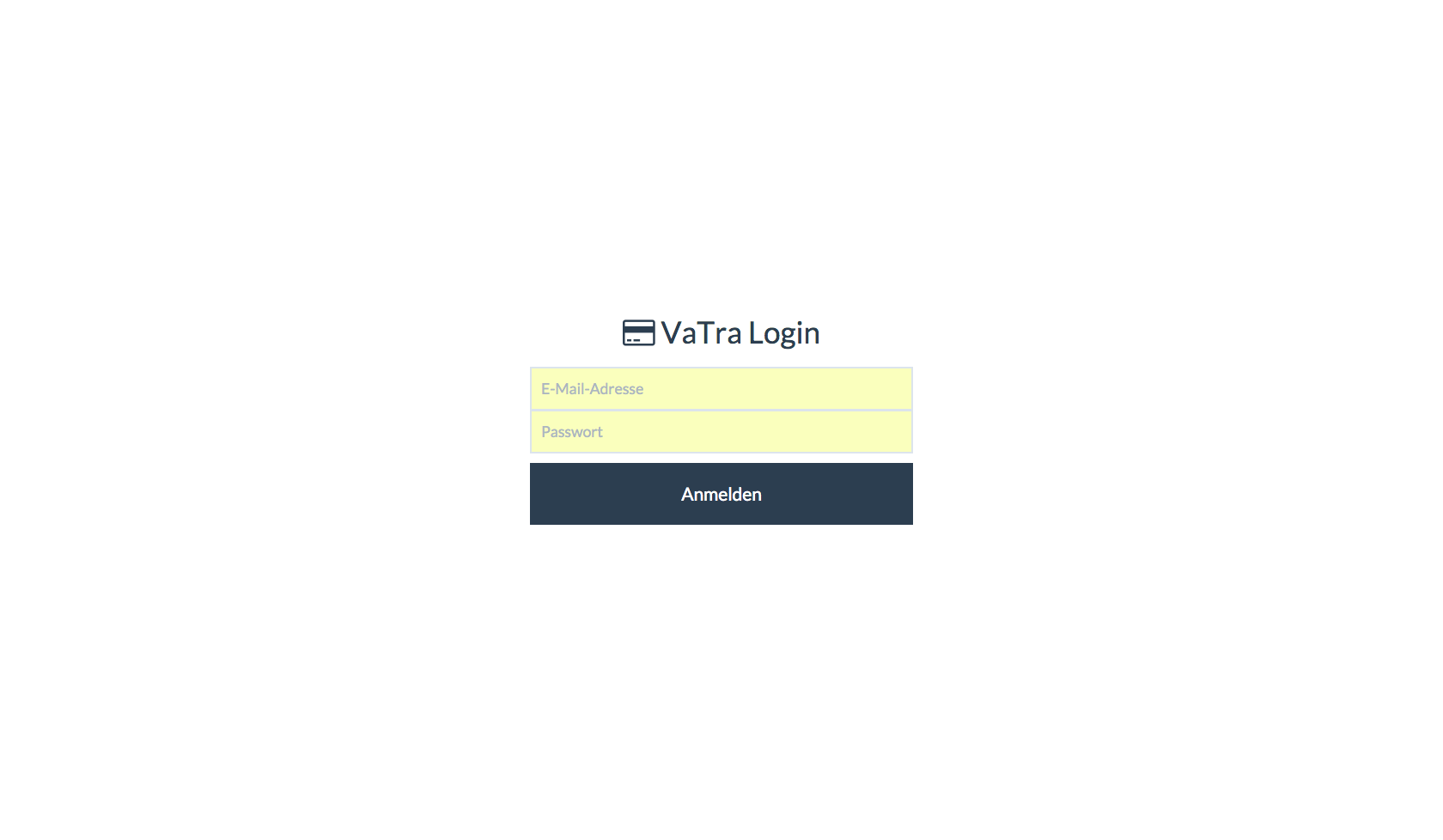


## API

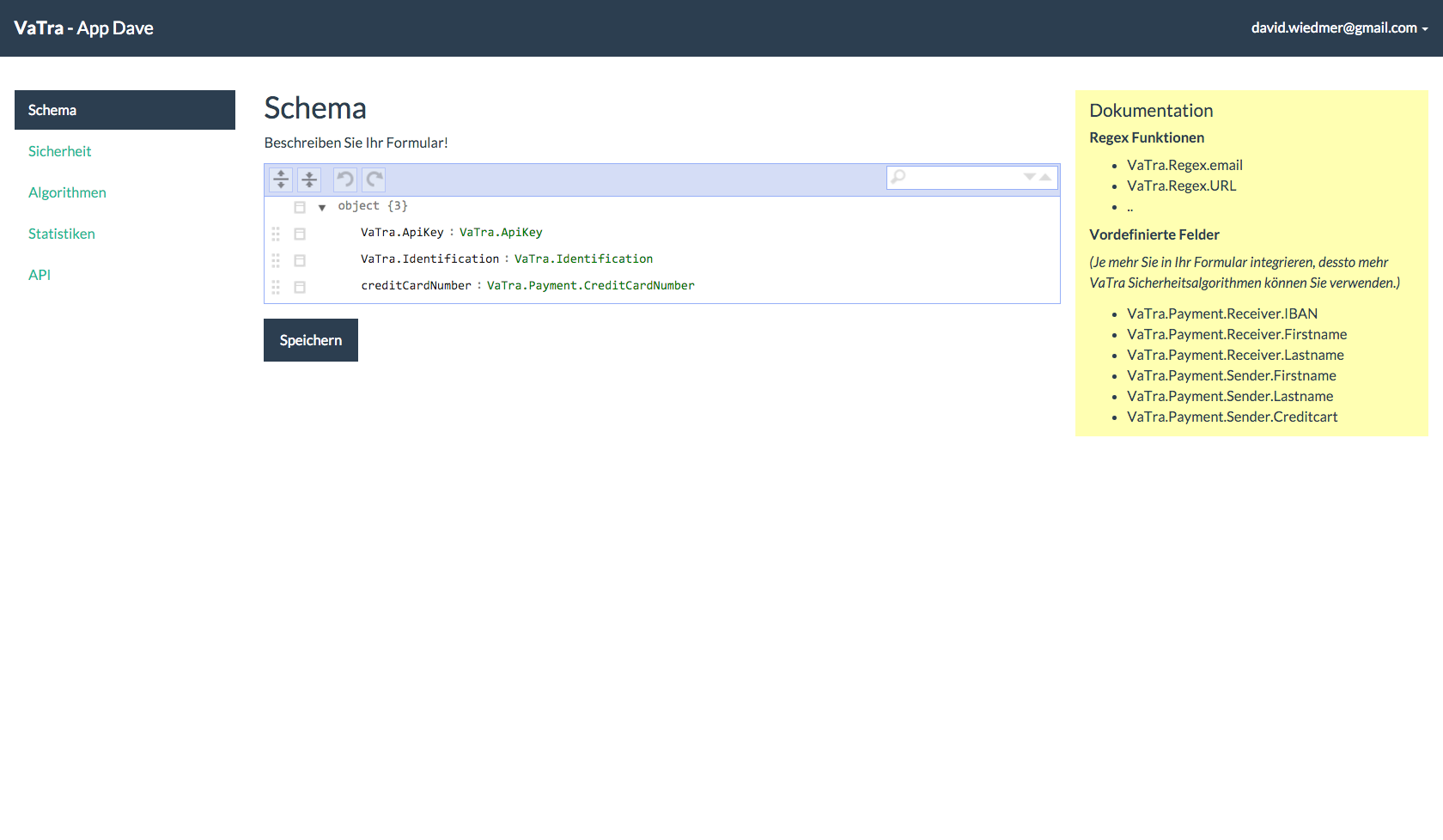
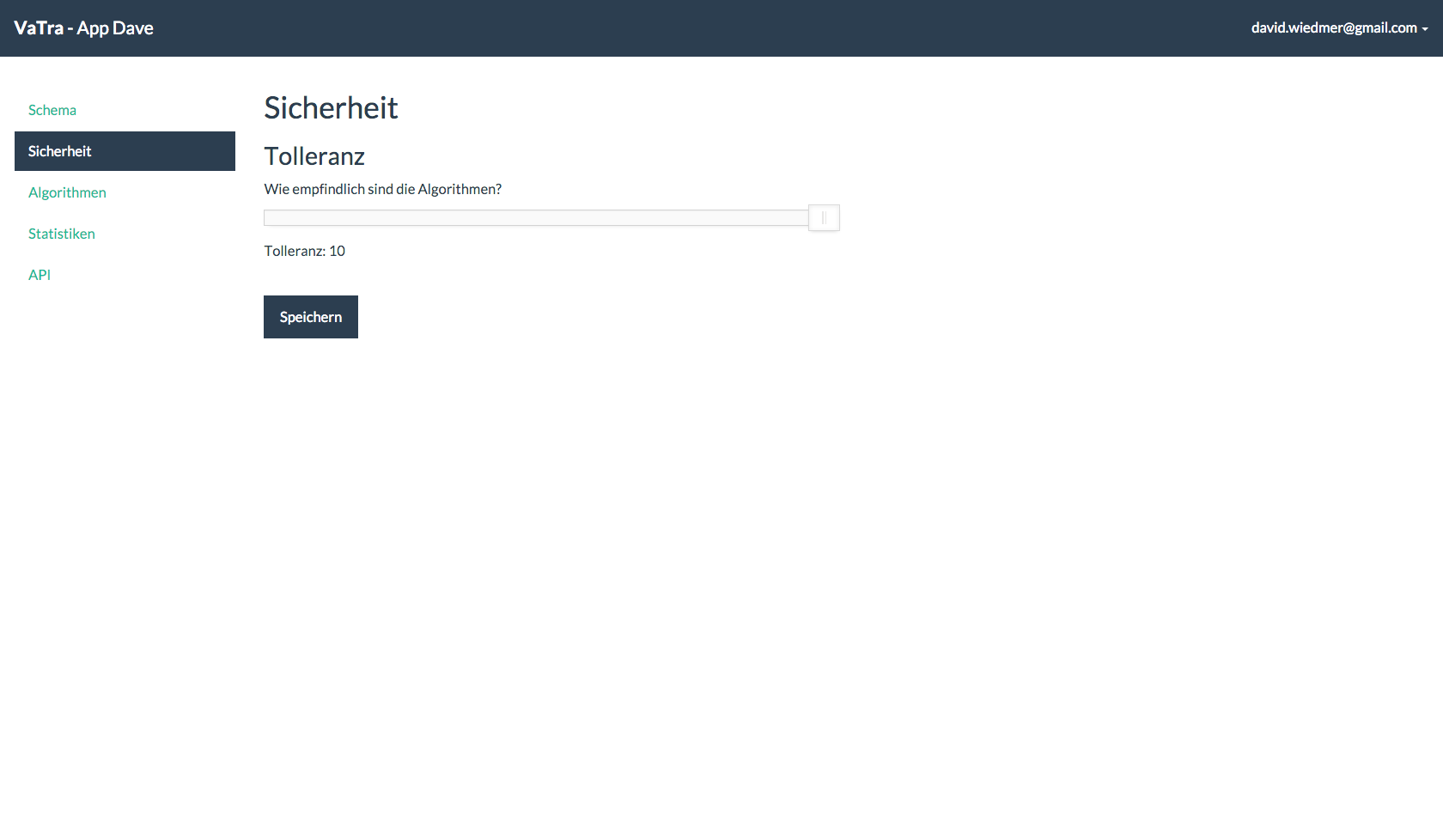


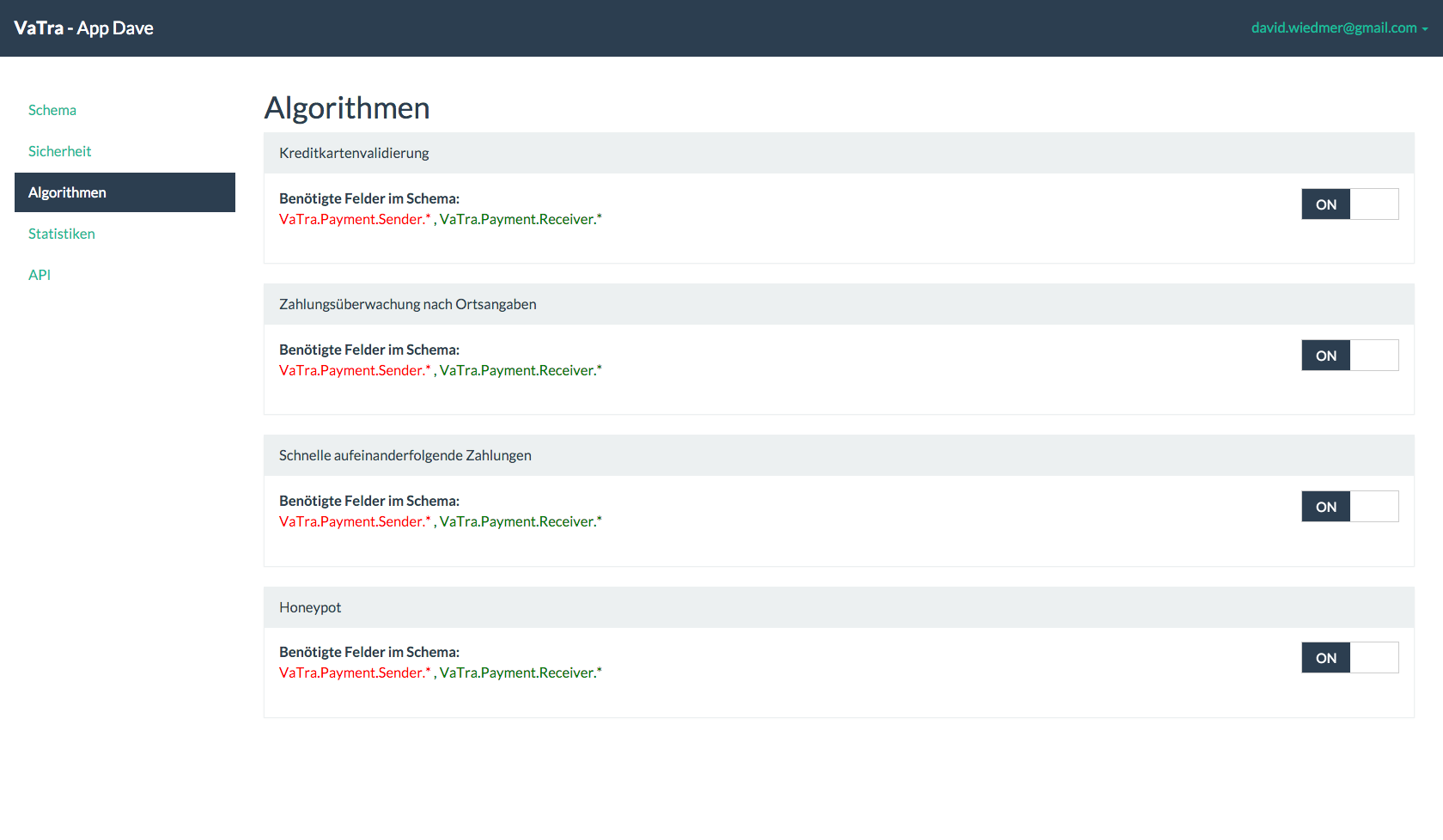
# Screens

Formular

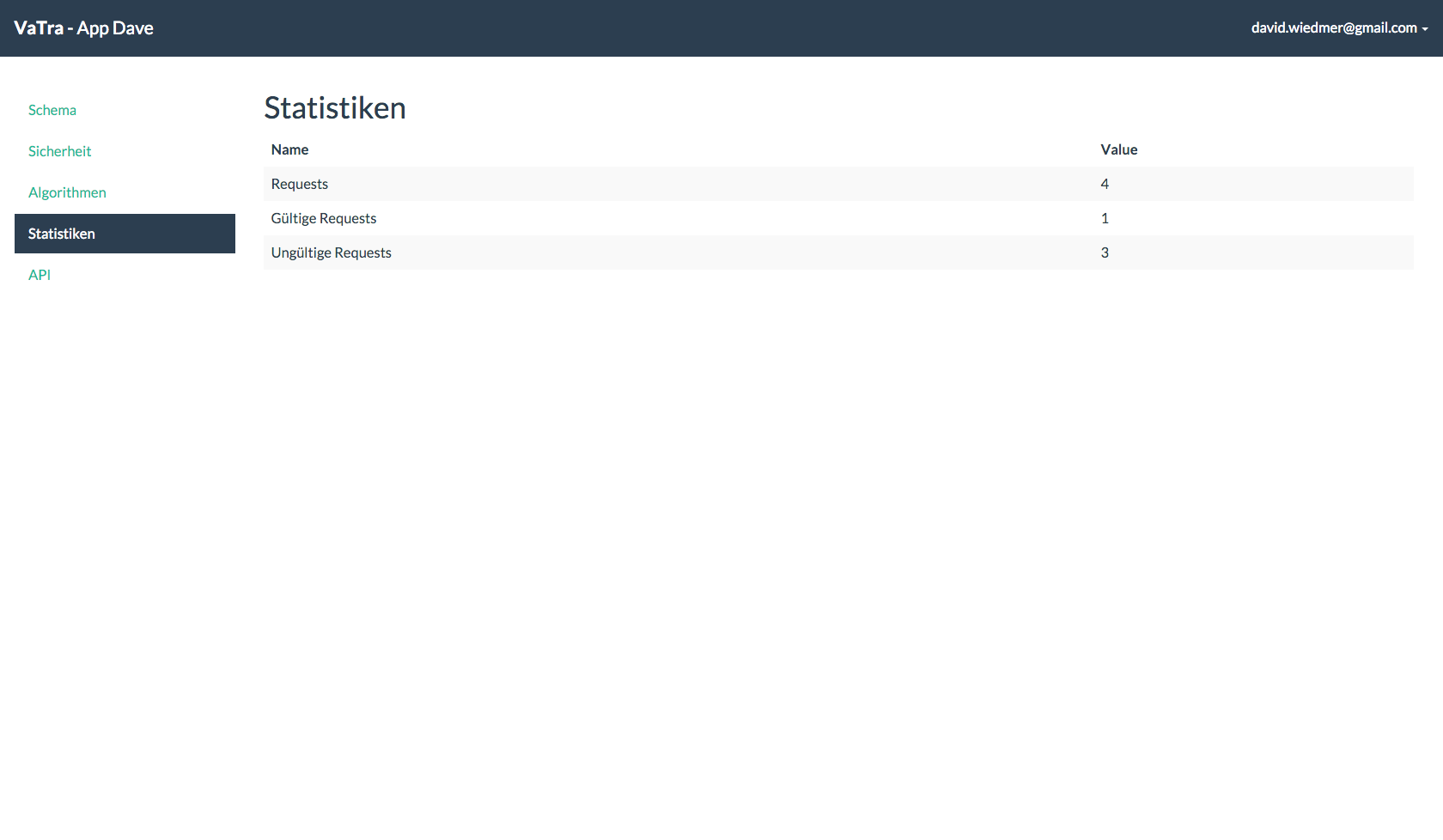
Login

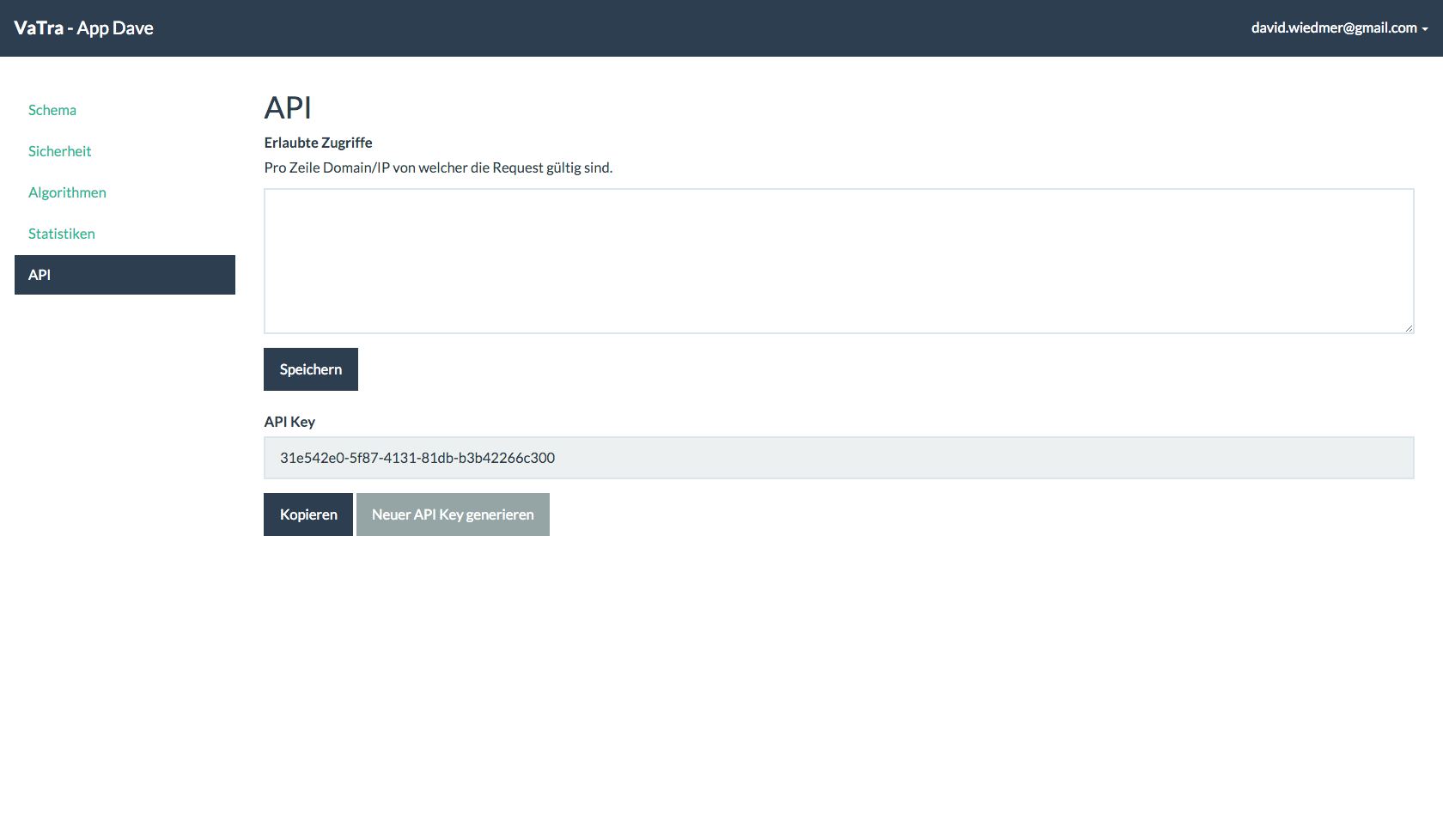
Dashboard

Sicherheit

Algorithmen

Statistiken



API

# Verwendete Hilfsmittel (Software)

|  |  |
| --- | --- |
| Bereich | Software |
| Java (API) | IntelliJ IDEA und Eclipse |
| PHP (Webserver) | PhpStorm IDE |
| HTML, CSS, JS (Backend) | PhpStorm DIE |
| Projekt Management | Jira |
| Versionierung | Git, Bitbucket, SourceTree |
| Auto Deployment | DeployBot |
| JS Task Runner | Grunt |

# Eingesetzte Technologien

|  |  |
| --- | --- |
| Bereich | Technologien |
| API / Webserver | Java, Spring Boot Framework, H2 Datenbank |
| PHP-Server | Apache (XAMPP), PHP, cURL |
| Backend (Styling) | HTML5, Sass (Syntactically Awesome Style Sheets) |
| Backend (Funktionalität) | AngularJS, Yeoman |
| Auto Deployment | Shell Script |

# Ausarbeitungsmöglichkeiten

* Schema mit eigenen Regeln
* Noch mehr Stats für Alogs und Zugriffe.

# Journal

## Termine

### 23.09.2015 - Meeting 1 mit Herr Kaltz

Die Grundidee des Projekts wurde in einem ersten Meeting von Herr Kaltz kommuniziert. Auf das nächste Meeting ist eine Präsentation eines konkreten Konzepts zu erstellen.

### 01.10.2015 - Workshop im Projektteam

In einem Workshop haben wir unsere Ideen zusammengetragen. Bestanden hat letztendlich die Idee von einem Webservice, der als zentrale Stelle für Validierungen entwickelt werden soll: VaTra.

### 05.10.2015 - Meeting 2 mit Herr Kaltz

Vorstellung unserer Idee in Form einer kurzen Präsentation. Die Idee wurde so von Herr Kaltz verabschiedet. Das Konzept soll jedoch ausgearbeitet werden. So soll eine erste Version des Entity-Relationship-Modells (ERM) erstellt werden.

### 19.10.2015 - Meeting 3 mit Herr Kaltz

Wir haben unsere geplante Vorgehensweise gezeigt. Das ERM haben wir kurz erläutert. Ausserdem ist die Freigabe der beiden Repositories erfolgt.

### 02.11.2015 - Meeting 4 mit Herr Kaltz

Heute konnten wir die klickbaren Views des Backends zeigen. Diese kamen gut an, es ist jedoch wichtig, dass wir auch genügend Zeit für die Algorithmen einplanen.

### 18.11.2015 - Meeting 5 mit Herr Kaltz

Vorführung der Authentifizierung anhand des Backends.

### 08.12.2016 - Meeting 6 mit Herr Kaltz

Vorführung des Dashboards: App hinzufügen, App auswählen und App wieder löschen.

### 06.01.2016 - Meeting 7 mit Herr Kaltz

Kurze Vorführung der Applikation und aufzeigen der Funktionsweise des Geolocation Algorithmus. Herr Kaltz konnte uns fragen betreffend Umfang der Dokumentation geben. Zudem schaut er sich unsere Requirement Engineering Dokumentation an um konkretes Feedback zugeben.

### 13.01.2016 - Meeting 8 mit Herr Kaltz

Herr Kaltz konnte uns direkte Verbesserungsmöglichkeiten anhand unserer ersten Dokumentversion aufzeigen. Zudem haben wir über die weiteren Inhalte, die in der Endversion benötigt werden, gesprochen. Der Abgabetermin für den Source Code ist am 22.01.2016.

### 22.01.2016 – Abgabe

Abgabe Termin für die Dokumentation und den Source Code. Keine weiteren Commits nach dem 22. Januar sind erwünscht.

## Arbeiten

### 23.09.2015 – 05.10.2015

Gemeinsame Erarbeitung eines Konzepts. Erstellen der Mockups auf Basis der erarbeiteten Idee.

### 06.10.2015 – 19.10.2015

Im Team haben wir ein ERM ausgearbeitet und einzelne Aufgaben vergeben.

* Einrichten und Anlegen der Tasks in Jira
* ERM ins reine übernehmen
* Anlegen der zwei git Repositories auf Bitbucket
* Initialisierung der Applikation
  + Webservice mit Springboot
  + Backend mit AngularJS

### 20.10.2015 – 02.11.2015

* Umsetzung der Views des Backends mit Bootstrap
* Erstellen der Models und Services im Webservice

### 03.10.2015 – 18.11.2015

* Programmierung der Authentifizierung für VaTra Anwender.

### 19.11.2015 – 08.12.2015

* Programmierung des Dashboards inklusive Apps
* Optimierung der Webservice Calls
* Ausarbeiten der einzelnen Services
* Implementierung der Whitelist

### 09.12.2015 – 06.01.2016

* Implementierung der Algorithmen
  + GeoLocation
  + CreditCard
  + Robotic
* Arbeiten an der Dokumentation
* Abfüllen der restlichen Views
* Programmierung eines Beispielformulars mit PHP
* Bugfixing

### 07.01.2016 – 22.01.2016

* Implementierung des Algorithmus Honeypot
* Formular Überprüfung anhand des App spezifischen Schemas
* Dokumentation abschliessen
* Erstellen einiger Unittests

### 22.01.2016+

Vorbereitung der Präsentation.

# Erkenntnisse

## Problem beim Geo-Algorithmus

Beim Testing vom Geo-Algorithmus ist uns folgendes Problem aufgefallen:

**Ausgangslage:**

Für die Berechnung des Geo-Algorithmus haben wir eine Schnittstelle von Google verwendet. Diese liefert uns Resultate, wie lange (zeitlich) bzw. wie weit die beiden Ortschaften voneinander entfernt sind. Dadurch können wir herausfinden, wie Wahrscheinlich es ist, dass die zweite Zahlung plötzlich im Ausland getätigt wurde.

**Bedingung:**

* 1. Zahlung aus der Schweiz (Bsp. Bern)
* 2. Zahlung von einem Ort, der zu Fuss, mit dem Auto oder per öffentlichem Verkehr nicht erreichbar ist.

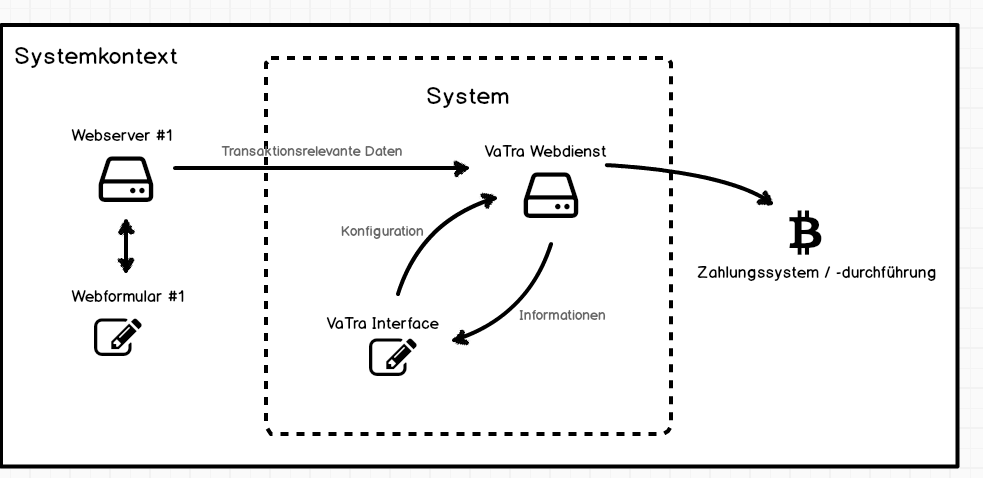
**Problem:**

Die Schnittstelle liefert uns teilweise ein leeres Resultat, da z.B. Amerika oder Australien nur per Flugzeug (oder Schiff) erreichbar ist und die Schnittstelle nicht dafür ausgelegt ist. Das impliziert, dass auch der Geo-Algorithmus nicht dafür ausgelegt ist.

## Weiterleitung von Formulardaten an Drittsystem

Während der Umsetzung ist uns aufgefallen, dass wir während der Planung einen grundlegenden Überlegungsfehler gemacht haben. Es macht keinen Sinn, die erfolgreich validierten Daten einer Zahlung direkt an ein Drittsystem zu senden. Aus dem Grund, weil der Anwender unserer Schnittstelle im Normalfall die Kommunikation mit dem Drittsystem selbst in die Hand nehmen möchte. Auch schon nur wegen dem Resultat, welches das Drittsystem liefert. Dieses müsste dann von unserer Schnittstelle wieder zurück zum Anwender gesendet werden, was entsprechend umständlich ist und wir uns mit der neuen Lösung sparen können.

Nachfolgend eine Grafik zum alten Systemkontext zur Veranschaulichung der ursprünglich geplanten Realisierung.



# Glossar

|  |  |
| --- | --- |
| **VaTra** | Valid Transaction |
| **API** | Application Programming Interface (auf Deutsch "Programmierschnittstelle") |
| **Validierung** | Überprüfung, ob eine Zahlung sicher ist. |
| **API Zugang** | Sicherheitsschlüssel: Wird für die VaTra API Authentifizierung verwendet. |
| **App** | Jeder VaTra-Administrator kann ein oder mehrere App(s) erstellen. Pro App können Sicherheitskonfigurationen vorgenommen werden. Im Normalfall verwendet man pro Zahlungsformular ein App. |
| **Backend** | Administrationsoberfläche des VaTra-Clouddienstes |
| **Laien Benutzer** | Ein Benutzer mit eingeschränktem technischem Know-how. Sollte sich VaTra Backend zurechtfinden. |
| **Empfindlichkeit Validierung** | Bei hoher Empfindlichkeit lässt die Validierung nur Zahlungen zu, die alle Validierungsalgorithmen bestanden haben. |
| **Toleranz Validierung** | Bei hoher Toleranz lässt die Validierung Zahlungen zu, die nicht alle Validierungsalgorithmen bzw. nicht alle zu 100 Prozent bestanden haben. |

1. Priorität: 1 – Niedrig | 2- Mittlere | 3 – Hohe Priorität [↑](#footnote-ref-1)
2. Komplexität: 1 – Niedrig | 2- Mittlere | 3 – Hohe Komplexität [↑](#footnote-ref-2)
3. Variabilität: 1 – Niedrig | 2- Mittlere | 3 – Hohe Variabilität [↑](#footnote-ref-3)
4. Risiko: (Priorität + 3\*Komplexität + 2\*Variabilität) / 6 [↑](#footnote-ref-4)