

Team Dachschaden

GVB Challenge

26.08.2016

# Teilnehmer

* Gian-Luca Frei
* Christian Wenger
* Daniel Ziltener
* Daniil Belov

## rEPOSITORY ZUM BEURTEILEN

<https://github.com/danbelov/GVBChallenge>

## ausgangslage

Als Challenge haben wir eine von GVB (Gebäudeversicherung Bern) ausgewählt, weil die im Aufgabe aufgestellte Problematik uns tiefsten angesprochen hat. Nach einem persönlichen Gespräch mit der Firma-Repräsentative haben wir festgestellt, dass die Automatisierungsgrad bei dieser Unternehmen noch sehr gering ist, und ein grosses Bedarf nach unserer zukünftiger Lösung vorhanden ist. Deshalb sind wir zu einer Idee gekommen, eine komplette Webapplikation aufzubauen, die eine Kundenverwaltung für E-Mail-lastige Kunden deutlich vereinfacht.

## Technischer AUfbau

Backend:

Wir haben ein backend welches eine JSON-REST Schnittstelle bietet mit Java programmiert. Order: /backend

Das Userinterface ist mit php/html/js/css programiert und findet sich unter /new-front Dieses renderet im Grunde HTML und leitet Anfragen an das Backend weiter.

Wir haben ein ML-Classifier entwickelt mit Python. Dieser findet sich unter /classifier

1. HTML, CSS, Javascript und PHP – Für eine sicheres und reibungsloser Applikationsbedienung.
2. Play Framework (basiert sich auf Java und sbt) für Backend.
3. Sklearn für ML Pipeline

## Implementation

Im Rahmen dieser Hackathon erarbeitete Lösung ist ein ‘**völlig‘** funktionstüchtiges System, die alle Eigenschaften eines MVP aufweist. Alle technischen Entscheide waren mit grosser Rücksicht auf Kundenbedürfnisse und User Experience sowie Maintainability und Simplicity ausgefertigt.

Unsere innovative Machine-Learning Pipeline durchsucht neue Schadensmeldungen automatisch nach betrügerischen Mustern und transformiert damit die Geschäftssprozesse in die nächste AI-Generation.

Der Arbeit war in mehrere Stufen aufgeteilt und agile durchgeführt:

1. Als erstes war ein Mail-Grabber im Python mit Hilfe eines kontinuierlichen Prozesses geschrieben der nach jedem Mail-Eingang die nötigen Daten aus der Mail zum Backend bereitstellt.
2. Als zweites wurde eine AI-Lösung erarbeitet, die nach einem Machine-Learning Vorgang auf Basis der existierenden Daten der Firma Verarbeitung des E-Mails durchführt. Diese wurde auch im Python geschrieben.
3. Nachdem der Bearbeitungsprozess abgeschlossen wird, schickt der dritte Teil der Lösung - unseres Backend Server im Java einen E-Mail zum Kunde, wo einen Link für Eingabeaufforderung in unserem Frontend in reinem HTML, PHP, CSS und JS vorhanden ist.
4. Das Frontend hat auch ein Bereich für den Mitarbeiter, der ganz bequem den Dialog mit der Kunde oder der Fall abschliessen kann