Dokumentation STARTHack Asimov

xxxx

David Schafer, Serge Hauri, Steve Ineichen, Remo Schwarzentruber

2019-03-17

Inhaltsverzeichnis

1 Getting Started **3**

1.1 Challenges . . . 3

1.2 AUTOSENSE / VOLVO Challenge . . . 3

1.3 Setup . . . 4

2 Tasks **5**

3 Tasks **6**

4 Damage drawer **7**

4.1 Funktionsumfgang . . . 7

4.2 Funktiondesign . . . 7

4.3 Umsetzung . . . 7

4.3.1 Init . . . 7

4.3.2 Kulturen Auto . . . 7

4.3.3 Zeichnen . . . 8

4.3.4 Zeichnen - Pfeil . . . 8

4.3.5 Zeichnen - Kreis . . . 8

4.3.6 Text auf das Bild . . . 8

4.3.7 Berechnung Beschädigung . . . 8

4.3.8 Schreiben der Bilddatei . . . 8

4.3.9 Pfad der geschriebene Datei . . . 9

4.3.10 Löschen von allen gerendert Dateien . . . 9

4.3.11 Anzeige der Datei auf dem Bildschirm . . . 9

4.4 Implementierung im Projekt . . . 9

4.5 Mögliche Darstellung der Datei in einem Protal . . . 9

5 Tasks **11**

1 Getting Started

1.1 Challenges

There were 8 different challenges which you could apply. We were mainly interested in the Challenges from the following partners:

- Autosense (Crash Visualization)
- SBB (Recylce)
- Laica (AR)
- BOSCH IOT-Lab (Sensor Car)

Alls case descriptions can be viewed here: http://live.starthack.ch/case-descriptions/

We applied for the Autosense challenge and got it (limit of 15 Teams per challange).
The challenge is as follow:

1.2 AUTOSENSE / VOLVO Challenge

Generate Car Crash Image, visualize impact and direction using sensor data

Your challenge if you choose to accept:

Build Microservice(s) to generate Image with 3D object simulating impact forces for given time offset (from crash). Deploy Microservice(s) on Swisscom Application Cloud (cloud foundry). Provide API(s) for submitting Input data (stream) and getting the Result. Generate output for each submitted Crash Record : Direction of the impact (Impact angle and energy), visualize the damage show expected place of impact on car

Winner is the Team who:

Has identified the maximum number of crashes correctly providing - Correct impact direction & Most accurate 3D simulation (compared to real crash picture)

How it will be measured:

For each submitted Crash Record AND time offset, generate Image with Direction of the impact (Impact angle and energy), Visualized damage and Time offset with the maximum force/damage on the object. Crash Record is submitted to the service. The calculated impact direction will be compared with pictures from real crash.

Restrictions:

Service must be deployable on cloud infrastructure (AWS/Cloud Foundry/Kubernetes/Docker). Service should use as few as possible external APIs. Given Data Models and API POST Requests structure must be used.

1.3 Setup

xxxxx

```
pwd
```

2 Tasks

- Data Parsing (transform in more structured way -> acceleration, calibration)
  - define useful functions
  - implement functions
  - crash_record.py
- Webserver
  - create webserver (sanic)
  - implement requests
  - return some dummy data for the moment
  - webserver.py (rename main.py)
  - docker container
- Image
  - define interface
  - library to draw arrows
  - library to draw circles
  - image.py
- Visualization & Math
  - jupyter notebook visualization
  - define functions to calculate angles & impact
  - start crash_record_calculator.py

3 Tasks

## yes

4 Damage drawer

4.1 Funktionsumfgang

sads asd

4.2 Funktiondesign

Print-Screen Tablet Skizze

Skizze Damage drawer

4.3 Umsetzung

Klassendiagramm und Beschreibung der einzelnen Methoden

# Klasse DamageImage

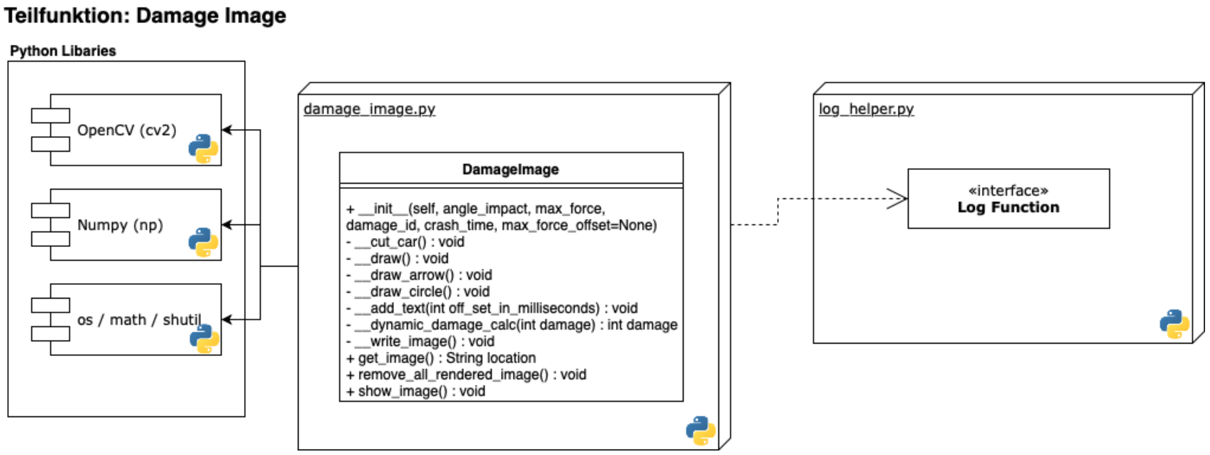

Teilfunktion: Damage Image
Python Libaries
OpenCV (cv2)
Numpy (np)
os / math / shutil
damage_image.py
DamageImage
+ __init__(self, angle_impact, max_force, damage_id, crash_time, max_force_offset=None)
- __cut_car() : void
- __draw() : void
- __draw_arrow() : void
- __draw_circle() : void
- __add_text(int off_set_in_milliseconds) : void
- __dynamic_damage_calc(int damage) : int damage
- __write_image() : void
+ get_image() : String location
+ remove_all_rendered_image() : void
+ show_image() : void
log_helper.py
«interface»
Log Function

Abbildung 4.1: Klassendiagramm Damage drawer

4.3.1 Init

```
def __init__(self, angle_impact, max_force, damage_id, crash_time, max_force_offset=None):
```

4.3.2 Kulturen Auto

```
def __cut_car(self):
```

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum.

4.3.3 Zeichnen

```python
def __draw(self):
```

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum.

4.3.4 Zeichnen - Pfeil

```python
def __draw_arrow(self):
```

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum.

4.3.5 Zeichnen - Kreis

```python
    def __draw_circle(self):
```

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum.

4.3.6 Text auf das Bild

```python
def __add_text(self, off_set_in_milliseconds):
```

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum.

4.3.7 Berechnung Beschädigung

```python
def __dynamic_damage_calc(self, damage):
```

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum.

4.3.8 Schreiben der Bilddatei

```python
def __write_image(self):
```

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum.

4.3.9 Pfad der geschriebene Datei

```
def get_image(self):
```

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum.

4.3.10 Löschen von allen gerendert Dateien

```
def remove_all_rendered_image(self):
```

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum.

4.3.11 Anzeige der Datei auf dem Bildschirm

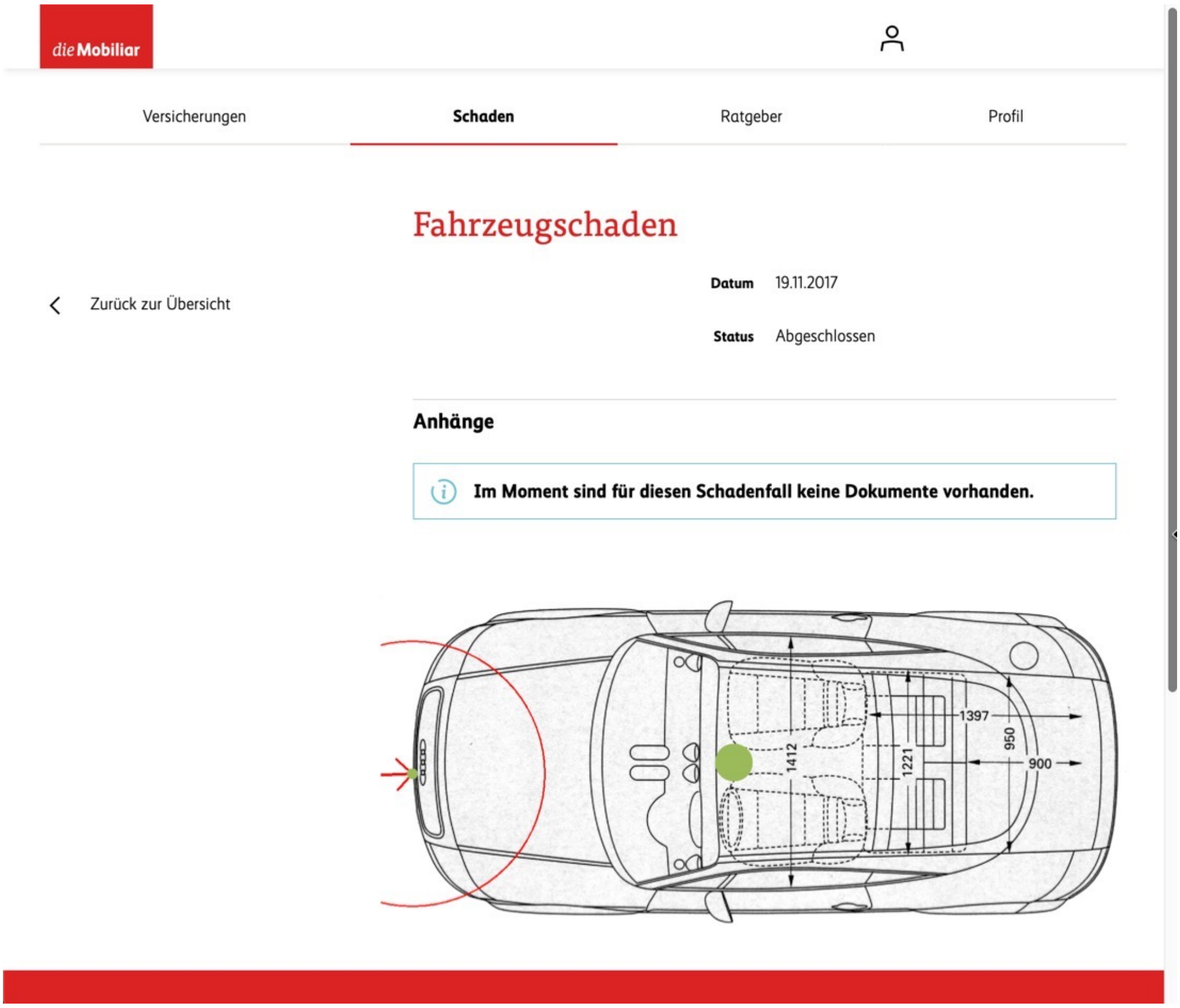
```
def show_image(self):
```

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum.

4.4 Implementierung im Projekt

Aufruf bei /api/v1/getCrashImage wie auch bei /api/v1/play

4.5 Mögliche Darstellung der Datei in einem Protal



Fahrzeugschaden

Datum 19.11.2017

Status Abgeschlossen

Anhänge

Im Moment sind für diesen Schadenfall keine Dokumente vorhanden.

Abbildung 4.2: Anzeige in einem Portal

5 Tasks

## yes