FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

PHILOSOPHISCHE FAKULTÄT
UND FACHBEREICH THEOLOGIE

SeeFood

Portable Food Nutrition Service

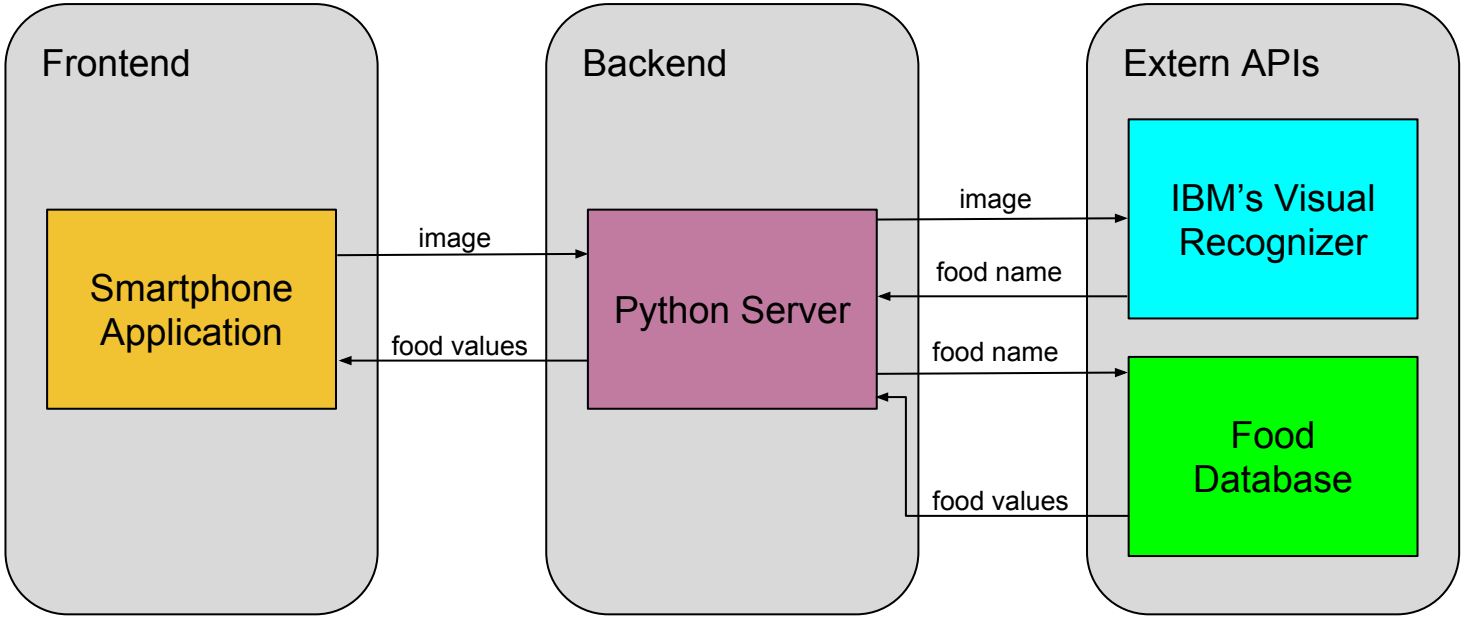
Outline

- General idea
- First approach
- Failures
- Final products

General idea

- take a photo of food with a smartphone
- Watson detects what food is shown
- get back the food values

First approach



Nils Häusler

Mustafa Öztürk

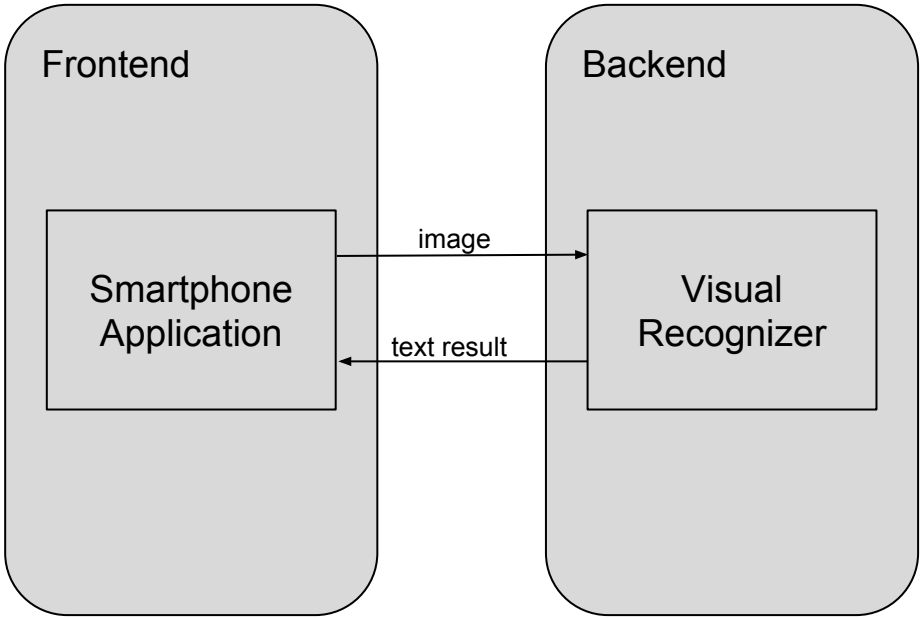
Abdulkadir Aymaz

Max Wiesner

Failures

- limited support from IBM to kinetise
- missing test device (IOS)
- similar limitations on other creators (e.g. mobincube)

Final product - Android Application

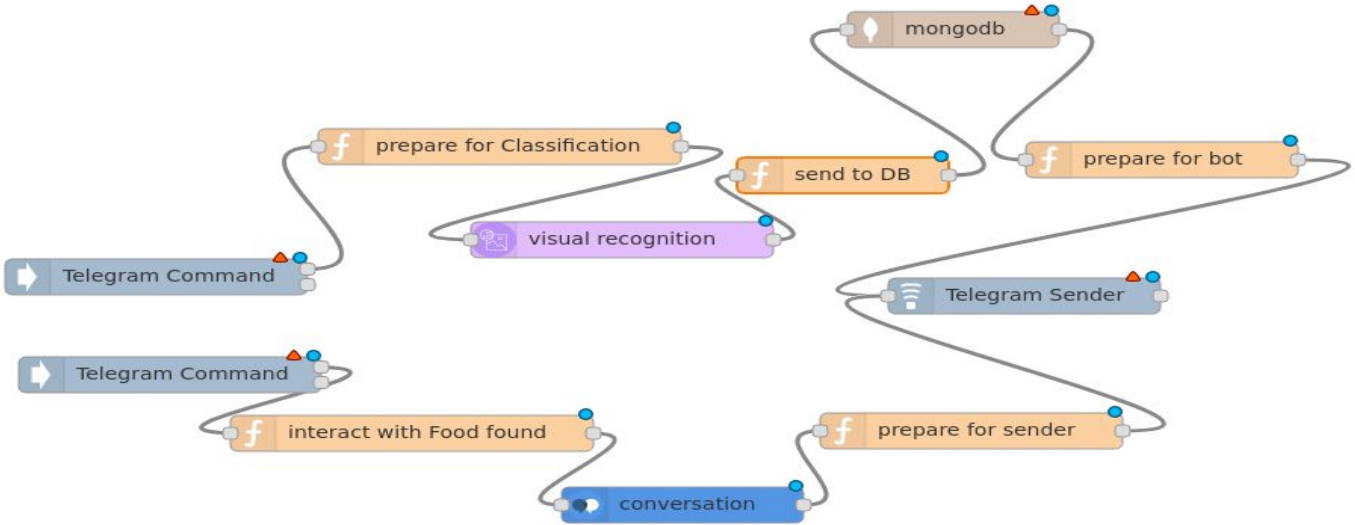


Final Product - Android Application

```
//foods (kcal, protein, fat, carbohydrates, names
foods.add(new Food(55, 0.3, 0.3, 11.7, "apple"));
foods.add(new Food(95, 1.1, 0.3, 21.0, "edible banana", "banana"));
foods.add(new Food(21, 0.8, 0.3, 3.2, "tomato"));
foods.add(new Food(742, 15.9, 70.8, 7, "walnut"));
foods.add(new Food(885, 0.1, 98.3, 0, "butter"));
foods.add(new Food(265, 9.2, 4.7, 44.6, "bread"));

VisualRecognition service = new VisualRecognition(
        VisualRecognition.VERSION_DATE_2016_05_20);
service.setApiKey("245824153bc52f9e8473f4c1bb1040bc3fbb670a");
ClassifyImagesOptions options = new ClassifyImagesOptions.Builder().images(file).build();
VisualClassification result = service.classify(options).execute();
```

Chatbot with Node Red and Telegram



Final Product

- send the image to the Visual Recognizer, extract the right answer and pass the result on
- I used image URLs from Google to start, because there was no connection to the APP yet
- results like “edible banana” or “lemon yellow color” are not in our database
- output looked like this:

```
{
    "images_processed": 1,
    "custom_classes": 0,
    "images": [
        {
            "resolved_url": "http://lebensmittel-warenkunde.de/assets/images/bananen.jpg"
,
            "classifiers": [
                {
                    "classifier_id": "default",
                    "classes": [
                        {
                            "score": 0.984,
                            "class": "edible banana",
                            "type_hierarchy": "/plant/herb/banana/edible banana"
                        },
                        {
                            "score": 0.984,
                            "class": "banana"
                        },
                        {
                            "score": 0.984,
```

imports etc.

- import json
  from watson_developer_cloud import VisualRecognitionV3

❑ test_url = 'http://lebensmittel-warenkunde.de/assets/images/bananen.jpg'

test_url is a picture of a banana (for testing purposes)

★ visual_recognition = VisualRecognitionV3('2016-05-20', api_key='156f02f7a2afd0e4c1c50197db1f66a6e1fd6229')

here we use the VisualRecognition with the API-Key

```
url_result = visual_recognition.classify(images_url=test_url, threshold=0.6)
# score_results is a list of dicts
score_results = url_result["images"][0]["classifiers"][0]["classes"]
sorted_score_results = sorted(score_results, key = itemgetter('score'), reverse=True)
```

1. send image to the Visual Recognizer; only results of minimum 0.6 probability displayed
2. need only the part „classes“
3. sort the output descending by „score“

```
[
    {
        "type_hierarchy": "/plant/herb/banana/edible banana",
        "score": 0.984,
        "class": "edible banana"
    },
    {
        "score": 0.984,
        "class": "banana"
    },
    {
        "score": 0.984,
        "class": "herb"
    },
    {
        "score": 0.984,
        "class": "plant"
    },
    {
        "score": 0.939,
        "class": "lemon yellow color"
    },
    {
        "score": 0.938,
        "class": "pale yellow color"
    }
]
```

- search = sorted_score_results[0]['class']
  save the food name with the highest probability to *search*

```
myVal = next((item for item in sorted_score_results if item["class"] in str(lownames)), None)
if myVal is None:
    print("type of myVal is None")
else:
    search = myVal['class']
    print(search)
```

Final Product

- My task was to get a connection between the output of the app and the Nutrition value database
- E.g. if we take a photo of a banana and the app recognizes the picture as a banana, the nutrition facts should be shown for the specific food, in that case the nutrition facts of the banana
- The database consists of:
  - name of the food
  - category of the food
  - energy per 100g
  - protein per 100g
  - fat per 100g
  - carbohydrates per 100g

- My first question was: How do I get a good food value database?
- I searched the internet for good databases and I found one
- The problem was, that this database had a lot of unnecessary information about food, like the amount of salt per 100g
- So I removed the unnecessary informations and only the important ones left
- After that I tried to get a connection between the database
- In the next part I try to explain my code further more for a better understanding of what I tried to do

```
import pandas
import xlrd
import json
from os.path import join, dirname
from os import environ
from watson_developer_cloud import VisualRecognitionV3
from operator import itemgetter
```

- These are all the packages we need for this task
- xlrd is for the connection between the food value database

```
book = xlrd.open_workbook("Swiss Food Database.xlsx")

sheet = book.sheet_by_index(0)

total_rows = sheet.nrows
total_cols = sheet.ncols

table = list()
record = list()
```

- Here I first open the database and reading the first sheet
- After that I get the total rows and columns of the database
- Last I create two lists, which are helping us later to convert the database entries into lists

```
for x in range(total_rows):
    for y in range(total_cols):
        record.append(sheet.cell(x,y).value)
    table.append(record)
    record = []
    x += 1
```

- Im reading the total rows and columns and save them into the table list
- After that the table variable is a list of lists, containing every entry in the food value database

```
upnames = list()

for i in table:
    upnames.append(i[1])

lownames = [element.lower() for element in upnames]
```

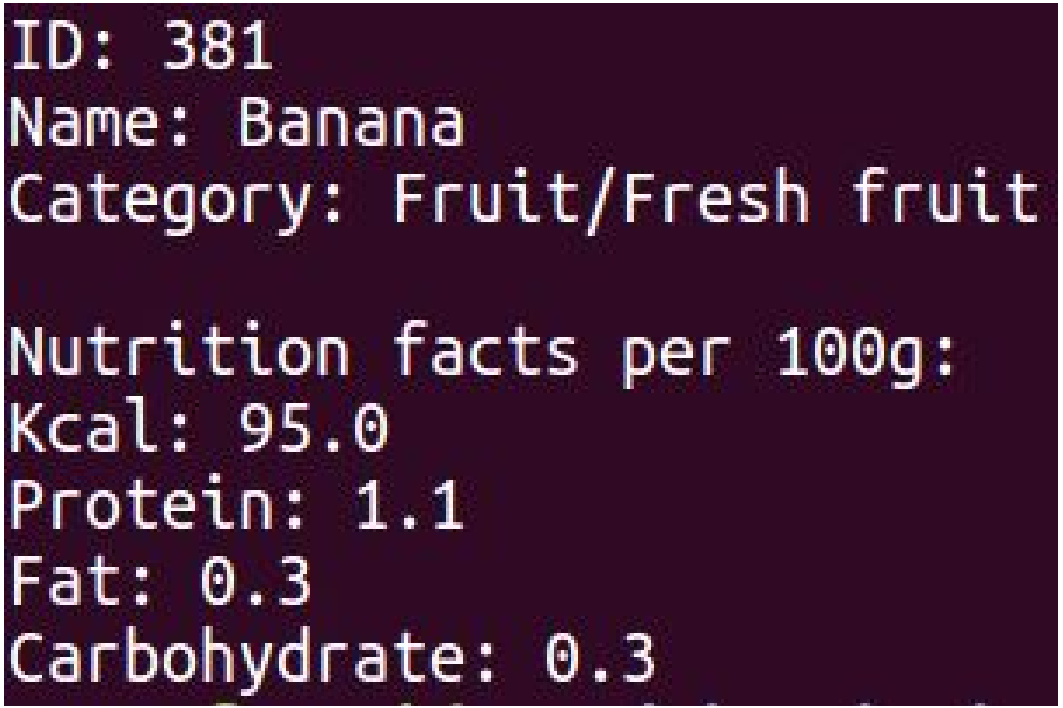
- I just convert the entry names into lowercase and save them to lownames so Mustafa can check if his result is listed in the database

```
def searcher(tab, x):

    for i in tab:
        h = 3
        if i[1].lower() == x.lower():
            print("ID:", int(i[0]), "\n" "Name:", i[1], "\n" "Category:",
"\n" "\n" "Nutrition facts per 100g:", "\n" "Kcal:", i[3], "\n" "Protein:'
, "\n" "Fat:", i[5], "\n" "Carbohydrate:", i[5],)
            break


searcher(table, search)
```

- The function searcher gets the name of the food from the visual recognizer
- It searches for the food in the database (table)
- If the searcher finds the food in the database, it prints his nutrition facts in the order: ID, Name, Category, Energy, Protein, Fat, Carbohydrate
- Table ist the database and search is the food name, which we search in the database


ID: 381
Name: Banana
Category: Fruit/Fresh fruit
Nutrition facts per 100g:
Kcal: 95.0
Protein: 1.1
Fat: 0.3
Carbohydrate: 0.3