

Vorwissenschaftliche Arbeit

Monolithic vs. Microservices: a comparison based on the frameworks Flask and Django

Verfasst von: Benedict Armstrong

Klasse: 8A

Schuljahr: 2017/18

Betreuung durch: Mag. Christian Schöbel

Wien, {Monat der Einreichung}

Abstract

{Text}

Vorwort

{Text}

Table of Contents

[1 Introduction 5](#_Toc498768349)

[2 Monolithic Architecture 6](#_Toc498768350)

[2.1 Definition 6](#_Toc498768351)

[2.2 Historical Perspective 6](#_Toc498768352)

[3 Microservices Architecture 6](#_Toc498768353)

[3.1 Definition 6](#_Toc498768354)

[3.2 Historical Perspective 6](#_Toc498768355)

[4 Cinema Seat Reservation Application 7](#_Toc498768356)

[4.1 Concept 7](#_Toc498768357)

[4.2 Problems 7](#_Toc498768358)

[4.3 Implementation 7](#_Toc498768359)

[4.3.1 Database 7](#_Toc498768360)

[4.3.2 Userstories 7](#_Toc498768361)

[5 Monolithic and Microservices Architectures in Practice 7](#_Toc498768362)

[5.1 Introduction 7](#_Toc498768363)

[5.2 Microservices 8](#_Toc498768364)

[5.2.1 Database Operations 8](#_Toc498768365)

[5.2.2 Rest Server 8](#_Toc498768366)

[5.2.3 Open API with connexion 8](#_Toc498768367)

[5.2.4 Clientside 8](#_Toc498768368)

[5.2.5 Design 8](#_Toc498768369)

[5.3 Monolithic 8](#_Toc498768370)

[5.3.1 Database Operations with Django Database API 8](#_Toc498768371)

[5.3.2 Integrating Legacy Database in Django 8](#_Toc498768372)

[5.3.3 Views 8](#_Toc498768373)

[5.3.4 Forms and working with Django Framework 8](#_Toc498768374)

[5.3.5 Design 8](#_Toc498768375)

[6 Comparitive Advantages and Disadvantages 8](#_Toc498768376)

[6.1 Monolithic Architecture 8](#_Toc498768377)

[6.1.1 Development concerns 8](#_Toc498768378)

[**6.1.2** Production concerns 8](#_Toc498768379)

[6.2 Microservices Architecture 9](#_Toc498768380)

[6.2.1 Development concerns 9](#_Toc498768381)

[6.2.2 Production concerns 9](#_Toc498768382)

[7 Conclusion 9](#_Toc498768383)

[8 Sources 9](#_Toc498768384)

# Introduction

{Text}

# Monolithic Architecture

## Definition

Monolithic architecture in general describes software where the user interface (UI) and the business logic are combined on a single platform. Similarly, in webdevelopment terms it referes to a model for the design of software in which the application is composed all in one piece. The UI is generated as a so called “View” on the server and sent out finished in one piece to the user. Typically, the components are interconnected and all of these components need to be present for code to be executed or compiled. A framework is not required but it greatly eases the development so for this paper I have chosen to uses Django, developed and maintained by the Django Software Foundation.

## Historical Perspective

The first applications to be developed for mainframes in the early 50’s up to the 70’s were monolythic. Only with the standardisation of TCP wide spread networking became possible and with that microservices as we know them.

# Microservices Architecture

## Definition

Microservice architecture describes the logical structure and design of a software application which consists of modular, loosely coupled components. These Services should be lightweight, independent and simple. From a web development point of view the key difference, compared to a monolithic architecture, is the separation of the presentation layer and the business logic, that means that the UI is composed in the browser on the client’s machine instead of the server. Applications like Netflix use this to reduce server load which dramatically reduces costs for the company. As with Monolithic architecture there are many different frameworks to ease the development process and for this paper I have chosen to use Flask which is open source and can be found on Github.

## Historical Perspective

The term Microservice was first used to describe this style of application design in 2011 by a workshop of software architects near Venice and has since gained popularity.

{I need to do further research on the topic as there are a lot of conflicting sources out there}

# Monolithic and Microservice Architectures in Practice

## Cinema Seat Reservation Application

### Introduction

In order to compare Monolithic and Microservice Architectures I have decided to get some first-hand experience and write my own Application. As a practical real-world example, I have created a web app for reserving seats in a cinema using both approaches.

### Requirements

The User should be able to pick a Movie from a List and then reserve seats for a specific Time and Date. The System should also support the adding and removal of movies by an administrator. For simplicity, the authentication and authorization were ignored.

### Structure



{Explain Structure?}

### Problems

{Include Section?}

### Implementation

#### Database

{Insert Entity diagram}

For this application, I have chosen to keep the database as simple and straight forward as possible. I am using a Postgres database structured as seen in {Picture of entity diagram}

#### Userstories

## Microservices

### Database Operations

### Rest Server

### Open API with connexion

### Clientside

### Design

## Monolithic

### Database Operations with Django Database API

### Integrating Legacy Database in Django

### Views

### Forms and working with Django Framework

### Design

# Comparitive Advantages and Disadvantages

## Monolithic Architecture

### Development concerns

#### Learning curve

Learning monolithic programming might seem simple in the beginning especially with a framework like Django but that can be deceiving because even though you might be able to write a simple application quickly you’ll most likely lack an understanding of the processes going on under the hood which will eventually lead to problems.

#### Teamwork

#### Development Time and

### Production concerns

#### Performance

#### Scalability

#### Security

#### Upgradeability

#### Crossplatform

## Microservices Architecture

### Development concerns

#### Learning Curve

#### Teamwork

### Production concerns

#### Performance

#### Scalability

#### Security

#### Upgradeability

#### Crossplatform

# Conclusion

# Sources

<http://whatis.techtarget.com/definition/monolithic-architecture>

<https://en.wikipedia.org/wiki/Monolithic_application>

<http://flask.pocoo.org>

<https://www.djangoproject.com>

<https://github.com/benedict-armstrong/VWA>