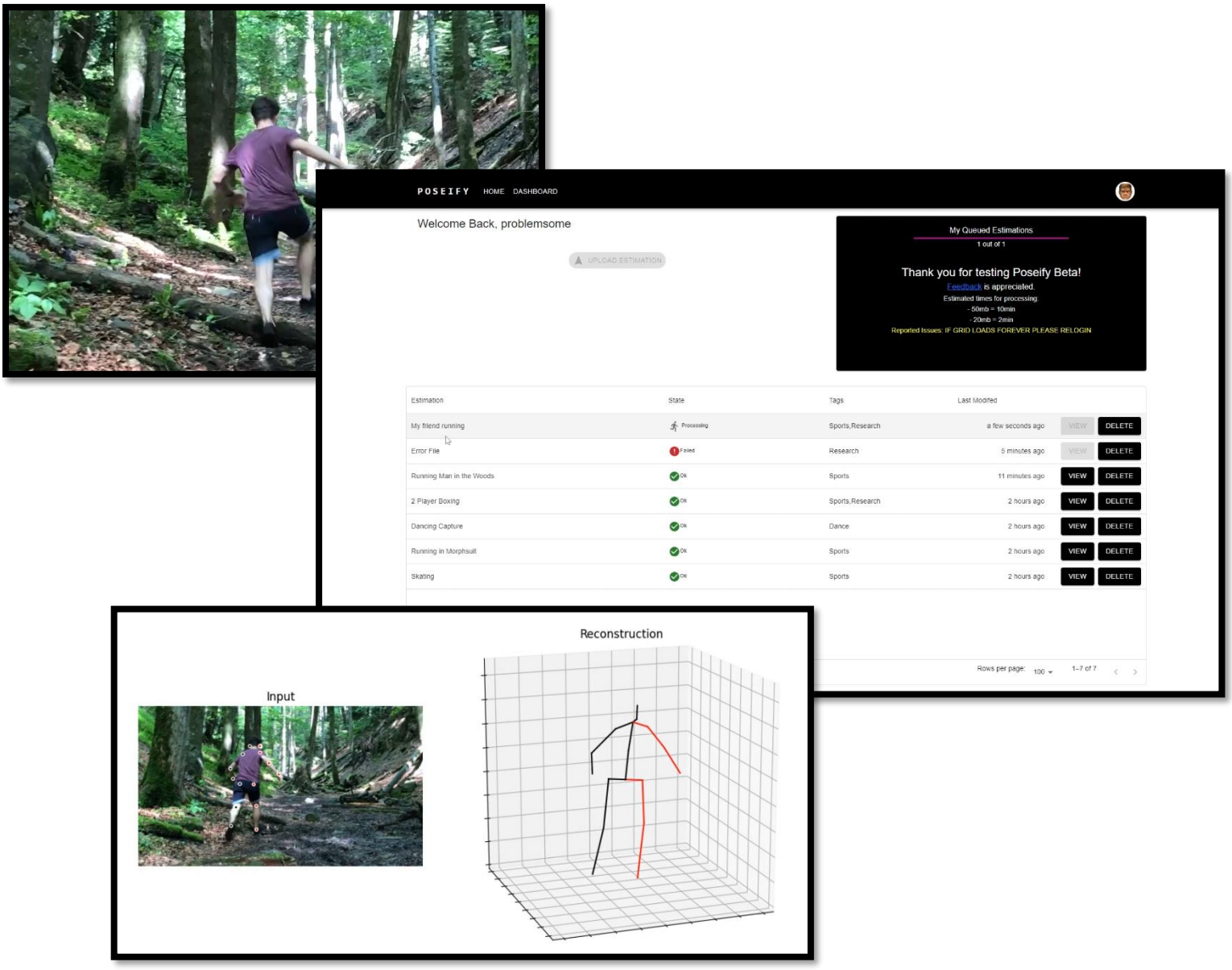
Wirtschaftsprojekt 2023



< 3D Pose Estimation Server >

Hochschule Luzern - Informatik
Corsin Kirchhofer | Mike Pullen

Wirtschaftsprojekt an der Hochschule Luzern – Informatik

Titel: 3D Pose Estimation Server

Studentin/Student: Pullen Mike

Studentin/Student: Kirchhofer Corsin

Studiengang: BSc Informatik

Jahr: 2023

Betreuungsperson: Dr. Smolic Aljosa

Expertin/Experte: Dr. Zank Markus

Auftraggeberin/Auftraggeber: HSLU Immersive Realities Lab

Codierung / Klassifizierung der Arbeit:
☒ Öffentlich (Normalfall)
☐ Vertraulich

Eidesstattliche Erklärung
Ich erkläre hiermit, dass ich/wir die vorliegende Arbeit selbständig und ohne unerlaubte fremde Hilfe angefertigt haben, alle verwendeten Quellen, Literatur und andere Hilfsmittel angegeben haben, wörtlich oder inhaltlich entnommene Stellen als solche kenntlich gemacht haben, das Vertraulichkeitsinteresse des Auftraggebers wahren und die Urheberrechtsbestimmungen der Hochschule Luzern respektieren werden.

Ort / Datum, Unterschrift Luzern 30.05.2023 C. Kirchhofer

Ort / Datum, Unterschrift Luzern 31.05.2023

Geistiges Eigentum gemäss der Studienordnung für die Ausbildung an der Hochschule Luzern, FH Zentralschweiz

Abstract
Durch *Pose Estimation* kann mit einer einfachen Videoaufnahme eine Art *Motion-Capture* gemacht werden, bei welcher die Gelenkpositionen einer Person im Video von einem Algorithmus geschätzt werden. Da *Pose Estimation* Algorithmen meist durch Forschung entstehen sind diese meistens nicht ohne lokale Installation und technisches Know-How zugänglich. Die vorliegende Arbeit versucht den *Pose Estimation* Algorithmus *VideoPose3D* als einfach zugänglicher Service verfügbar zu machen. Dafür wurde eine *Webapplikation* implementiert, durch die der *Pose Estimation* Algorithmus unkompliziert öffentlich zugänglich gemacht werden kann. So kann *VideoPose3D* auf Servern gehostet werden und Enduser müssen nicht eine lokale Installation vornehmen und oder Zugang zu geeigneter Hard- und Software haben. Die Webapplikation besteht effektiv aus mehreren Teilprojekten, welche alle in einem *Open-Source* GitHub Repository öffentlich zugänglich sind. Insbesondere besteht das Projekt besteht aus folgenden Elementen:

- Webapplikation, als UI
- Backend-for-Frontend, zur Absicherung vor unerlaubten Zugriffen
- Identity Server, zum Authentifizieren und Autorisieren von Personen und der Webapplikation
- Core Backend, als *API* über welche die *VideoPose3D* Funktionalität angeboten wird.
- Datenbank, zum Speichern der *Estimation* Ergebnisse

Das Projekt ist lauffähig. Es kann auf einem oder mehreren Servern ausgeführt werden. Nutzende können darauf zugreifen und Videos hochladen, welche durch *VideoPose3D* zu *Joint-Positions* umgerechnet werden und als «.npz» oder «.json» Datei und einem Vorschau-Video zum Herunterladen angeboten werden.

Inhaltsverzeichnis

1 Vision

In diesem Kapitel wird die aktuelle Ausgangslage und das Potential von *Pose Estimation* erläutert.

1.1 Ausgangslage

«Menschliche *Pose Estimation* ist definiert als das Problem, die menschlichen Gelenke (auch als Keypoints – Ellenbogen, Handgelenk, etc.) in Bildern und Videos zu lokalisieren»[1]

Konventionell werden menschliche Bewegungen mittels eines *Motion Capture Systems* aufgezeichnet und in digitale Formate verarbeitet, womit *3D Modelle* animiert werden. *Motion Capture Systeme* sind generell teuer, da diese spezifische Hardware, Software, Umgebung und eine gewisse Vorbereitungszeit voraussetzen. Ein *Pose Estimation* Algorithmus ermöglicht die Abbildung von *Posen* und *Bewegungen* in einem digitalen Format, ohne dafür *Motion Capture Hard-* und *Software* zu benötigen. Mit nur einer Videoaufnahme ist es möglich, durch *Pose Estimation,* Bewegungen in einem digitalen 3D Raum abzubilden, auszuwerten oder weiterzuverarbeiten. Da *Pose Estimation* kostengünstiger und einfacher ist, könnte dessen Umsetzung als attraktive Alternative zu *Motion Capture Systems* in der digitalen Animation von Charakteren, der Sport- und Fitnessanalyse Anwendung finden. *Motion Capture* bietet weiterhin höhere Präzision und einen weitaus grösseren Detailgrad als *Pose Estimation*, jedoch ist dies je nach Fall weniger wichtig als Kosten, Zeitaufwand und Komplexität.

1.1.1 Anwendungsbereiche

In der Fitnessanalyse kann aus einer Videoaufnahme von einem Smartphone mittels *Pose Estimation* eine *3D-Rekonstruktion* einer oder mehreren ausgeführten Übungen gemacht werden. Damit kann ein Fitness-Coach entweder Übungen als *3D-Animation* vorzeigen oder Feedback zur Haltung und Ausführung einer Übung eines/er Klienten/in geben kann.

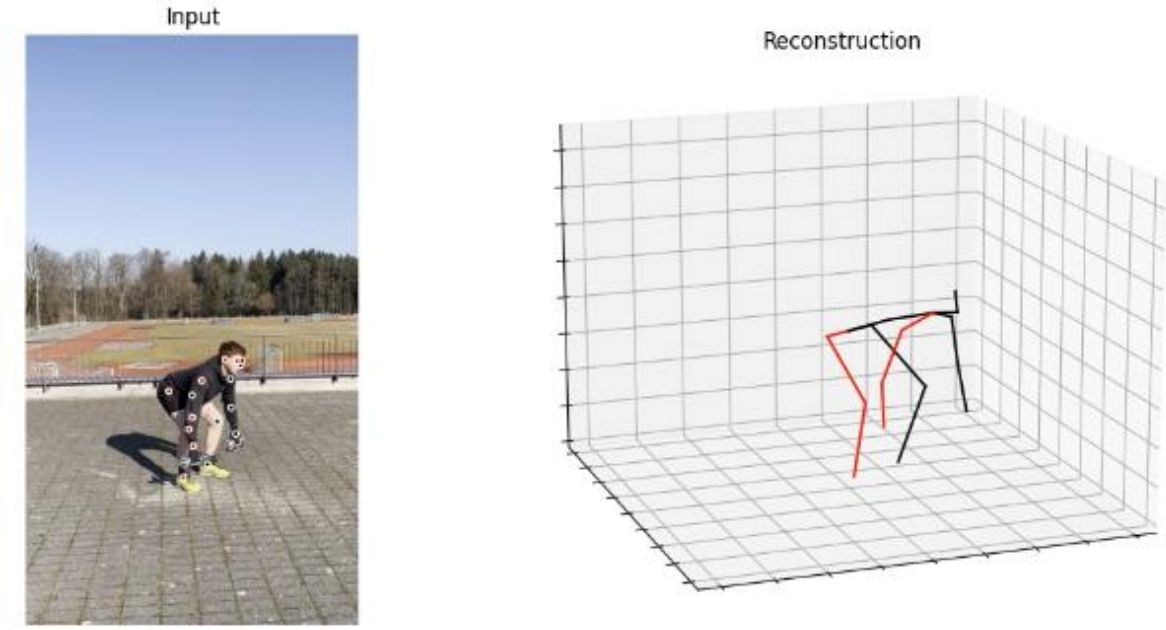


Abbildung 1: Rumänisches Kreuzheben (Volet, 2022)

[1] «Human Pose Estimation is defined as the problem of localization of human joints (also known as keypoints - elbows, wrists, etc) in images or videos.» (Babu, 2019)

Bei der Animation von *3D-Modellen* in der Hobby-/Indie-Gameentwicklung kann mittels *Pose Estimation* schnell *prototyping* betrieben werden. Die generierten Ergebnisse können noch bearbeitet werden, um Animationen den Wünschen der Endnutzenden anzupassen. Von Vorteil ist daher, dass eine Animation nicht von Grund auf erstellt werden muss, womit Aufwand und Zeit gespart werden können.

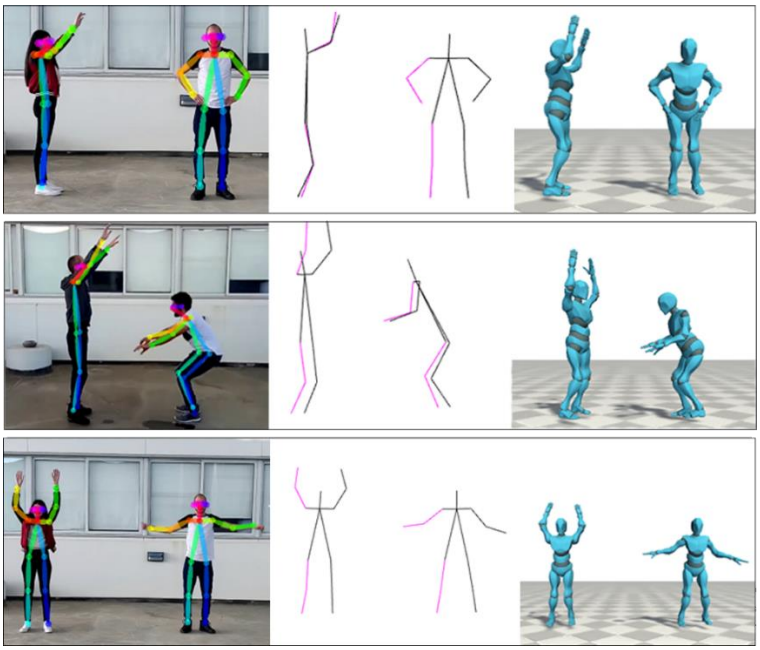


Abbildung 2: Pose Estimation to Animation (Yiannakides et al., 2019)

Pose Estimation kann aber auch in der Erstellung eines digitalen *Fashion Try-on Systems* verwendet werden. Eine Person kann eine Aufnahme mit Posen und Bewegungen bei einem Online Fashion-Shop hochladen und die gewünschten Kleider können in der Aufnahme auf die Person projiziert werden, damit diese sich besser vorstellen kann, wie das Produkt im Spiegel aussehen würde.

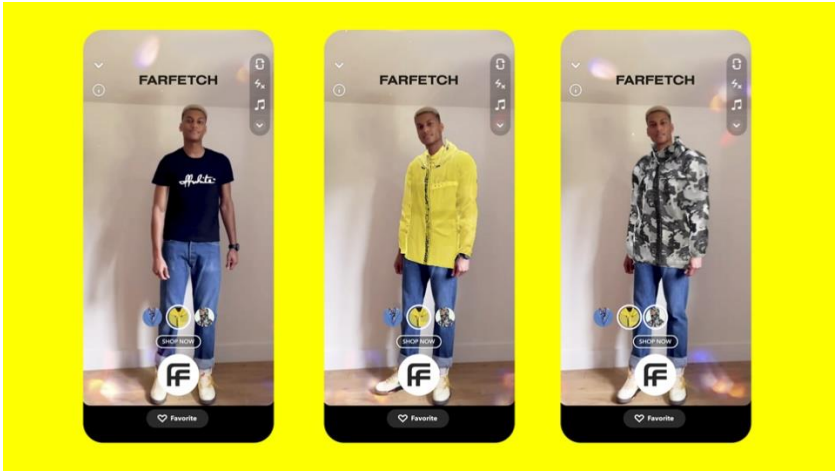


Abbildung 3: Virtual Try-On example (London, n.d.)

1.2 Aufgabenstellung und Ziele

In dieser Arbeit wird ein *Pose Estimation* Algorithmus in eine Server-Anwendung integriert die als *Open Source* Projekt öffentlich nutzbar ist. Dieses Projekt soll eine freizugängliche Alternative zu traditionellen *Motion Capture Systemen* bieten und als Grundlage zum öffentlichen Anbieten eines *Pose Estimation* Algorithmus dienen.

2 Stand der Technik

Pose Estimation hat in kurzer Zeit grosse Fortschritte erlebt und es sind einige Algorithmen entstanden, welche mit verschiedenen Methoden *Pose Estimation* umsetzen.

2.1 Übersicht

Eine Auswahl von verschiedenen aktuellen *Pose Estimation* Algorithmen:

- VideoPose3D (Pavllo et al., 2019)
  - 3D, Facebook, Convolutional
- VIBE (Kocabas et al., 2020)
  - 3D, Volume, Temporal, Adversarial Training
- OpenPose (Cao et al., 2019)
  - 2D, Realtime
- HRNet (Sun et al., 2019)
- BlazePose (Bazarevsky et al., 2020)
  - 3D, google, Realtime, Convolutional
- AlphaPose (Fang et al., 2022)
  - 3D, Multiperson, Realtime
- PoseNet (Kendall et al., 2016)
  - Convolutional
- DensePose (Güler et al., 2018)
  - 3D, Facebook, Convolutional
- DeepCut (Pishchulin et al., 2016)
  - Multiperson, Convolutional

VideoPose3D, VIBE, BlazePose, AlphaPose sind die neusten und am vielversprechendsten Algorithmen.

Grundsätzlich gibt es drei Modelle für das Modellieren eines menschlichen Körpers mit *Pose Estimation*, *kinematisch, planar* und *volumetrisch* (Zheng et al., 2022). Gewisse Algorithmen können die *Pose Estimation* auch in Echtzeit durchführen, andere können mit mehreren Personen in einer Aufnahme gleichzeitig umgehen. Nicht jeder *Pose Estimation* Algorithmus produziert 3D-Resultate. Trendmässig sind neuere *Pose Estimation* Algorithmen als *Convolutional Neural Networks (CNN)* gebaut, da eine solche Architektur bessere Ergebnisse liefert.

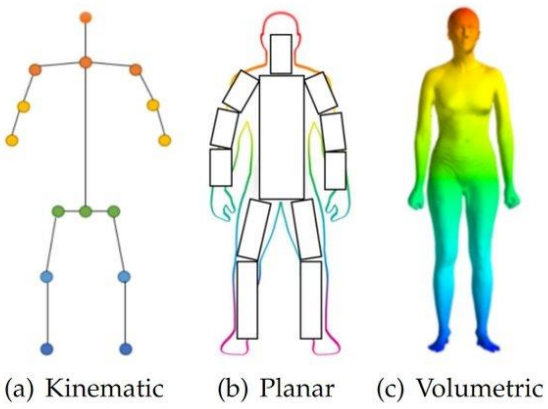


Abbildung 4: Modelle für Human Body Modeling (Zheng et al., 2022)

2.2 VideoPose3D

Für die *3D Pose Estimation* gilt *VideoPose3D* als einer der «state of the art» Algorithmen. *VideoPose3D* ist ein «vollkommenes convolutional Model basierend auf dilated temporal convolutions auf 2D Keypoints angewendet.»(Pavllo et al., 2019)[2] Es beinhaltet *Back-Projection* und eine simple *semi-supervised* Trainingsmethode auf *ungekennzeichneten* Video-Daten (Pavllo et al., 2019). Entwickelt wurde *VideoPose3D* durch ein Forschungsteam von *Facebook Research*, wobei der Algorithmus anschliessend auf GitHub veröffentlicht wurde. Die Resultate und Implementation von *VideoPose3D* wurden in der Arbeit «3D human pose estimation in video with temporal convolutions and semi-supervised training» veröffentlicht.

VideoPose3D ist theoretisch mit jedem *2D Keypoint Detection* Algorithmus kompatibel, auf GitHub wird *Detectron2* verwendet, der auch ein von *Facebook Research* entwickelter Algorithmus ist. Durch den «time dilated convolutional» Aufbau der Architektur werden Langzeit-Abhängigkeiten innerhalb einer Aufnahme effizient modelliert, dies bedeutet, dass die Vergangenheit und Zukunft in den einzelnen Frames einen wesentlichen Einfluss nehmen und das Resultat verbessern.

VideoPose3D bietet ein kinematisches Resultat (Eckpunkte) und liefert dies als «.npz» (Numpy Archive) Format.

2.2.1 Evaluation Guillaume Volet

Die Genauigkeit und Brauchbarkeit im Bereich von Fitnessübungen mit *Detectron2/VideoPose3D* wurde im Vorjahr an der HSLU von Guillaume Volet im Rahmen einer Bachelorarbeit durchgeführt. Dort hat sich gezeigt, dass die Aufnahmeperspektive, Art der Bewegung und Position der Person im Video einen grossen Einfluss auf die Qualität des Ergebnisses haben. Da das Tracking bei einem Teil der Übungen in entscheidenden Momenten fehlschlägt, können Fehlausübungen nicht festgestellt werden. «Zum aktuellen Zeitpunkt ist *VideoPose3D* also noch nicht geeignet, um im grossen Rahmen bestehende *Motion-Tracking Systeme* abzulösen.» (Volet, 2022, Kap. 6.1, 6.3).

2.2.2 Performance

In den Testläufen mit *VideoPose3D* wurde festgestellt, dass *FFMPEG* und *Detectron2* wesentliche Performance-Bottlenecks sind, wenn die *Pose Estimation* im CPU-Modus ausgeführt wird. Deshalb war es kritisch, mittels *CUDA* diese Bottlenecks zu verkleinern. *CUDA* ist von NVIDIA entwickelt und ermöglicht Software Engineers NVIDIA GPUs für «paralell processing» zu nutzen, welches benötigt wird, um *VideoPose3D* und spezifisch *Detectron2* Tasks auf GPU ausführen zu können.

Der Installationsaufwand für *CUDA* ist gross und erfordert vertiefte Kenntnisse, da das Version-Matching zwischen Python, Numpy, *CUDA*, *VideoPose3D* und *Detectron2* sehr mühsam ist. Es ist jedoch von Vorteil *CUDA* zu installieren, da es den *Pose Estimation* Prozess um etwa 5x-10x schneller macht.

Dateigrösse	**Zeit mit CPU**	**Zeit mit GPU (CUDA)**
4.13MB	13min	1min
9.00MB	22min	2min
25.4MB	47min	12

Tabelle 1: CPU und GPU Vergleich

Alle Zeiten wurde auf einem System gemessen mit einem AMD Ryzen 7 3800X 8-Core Prozessor und der NVIDIA GeForce RTX 3080 Grafikkarte.

[2] «fully convolutional model based on dilated temporal convolutions over 2D keypoints. We also introduce back-projection, a simple and effective semi-supervised training method that leverages unlabeled video data.» (Pavllo et al., 2019)

3 Ideen und Konzepte

In diesem Kapitel werden die Planungsresultate und Entscheidungen erläutert, welche zur Lösung der Aufgabenstellung beschlossen wurden. Insbesondere werden auf Branding, Code-Verwaltung und Technologieentscheidungen eingegangen.

Um einen Webserver für *Pose Estimation* zu realisieren, müssen Entscheidungen getroffen werden mit welchen Technologien, und Sprachen das System gebaut werden kann. Eine Serverarchitektur, Datenbankstruktur und eine Dateistruktur müssen definiert werden. Die Architektur drückt aus, wie die einzelnen Komponenten des Systems miteinander interagieren und zusammenhängen, die Datenbankstruktur legt fest in welchem Format Daten und Resultate persistiert werden, und mit der Dateistruktur wird festgelegt, wie eine Videodatei von Endnutzenden zur effektiven *Pose Estimation* durchgereicht wird.

3.1 Branding

Um ein erkennbares Software-Produkt zu entwickeln, ist es wichtig, sich Gedanken über das Branding zu machen. Der Name und das Aussehen der Anwendung sollten unkompliziert sein und ihren Zweck genau widerspiegeln. Um dieses Ziel zu erreichen, führte das Team ein ausführliches Brainstorming durch, um Namen zu generieren und gleichzeitig geeignete Stylingrichtlinien zu bestimmen. Die folgenden Prinzipien dienten als Grundlage für die vereinfachten Designrichtlinien:

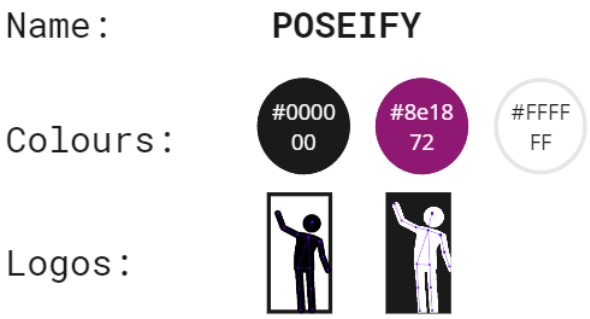


Abbildung 5: Poseify Design System

Der Name *Poseify* ist griffig, beschreibt die Kernfunktionalität und wird von keinem Unternehmen verwendet. Die schlichte Farbpalette mit den Schwerpunkten Schwarz, Weiss und Lila in Kombination mit der *Material Design* sorgt für ein sauberes und einfaches Erscheinungsbild, das auf extravagante Elemente verzichtet. Das Icon deutet die Funktionalität der *Pose Estimation* an, indem es die Gelenkpositionen darstellt, so wie es *VideoPose3D* in seinen Vorschauen tut.

3.1.1 Material Design

Material Design ist ein von *Google* entwickeltes *Open-Source-Designsystem*, das darauf abzielt, eine konsistente und visuell ansprechende *UI* für verschiedene Plattformen und Geräte zu schaffen. Dies führt dazu, dass sich *Poseify*, für Nutzende, die eine andere Anwendung im *Material Design* verwendet haben, bekannt anfühlt und sich schnell orientieren können (Google, 2023).

3.2 Open-Source

Das umgesetzte Projekt ist auf *GitHub* veröffentlicht und ist somit *Open-Source* (MIT LICENSE). Das bedeutet, dass jede Person, die möchte, das Projekt herunterladen, ausführen und erweitern kann. So können andere *Pose Estimation* Algorithmen (zukünftige oder aktuelle) eingebaut und angeboten werden. Diese Entscheidung wurde getroffen, da diese Arbeit im Rahmen eines Wirtschaftsprojekts entsteht, bei dem die Stakeholder hauptsächlich die Hochschule Luzern ist und das Projekt somit den Dozenten, Studenten und der Öffentlichkeit zugänglich sein soll. An der HSLU (oder anderen Institutionen) kann das Projekt in

Modulen eingesetzt werden und oder in zukünftigen Wirschaftsprojekt-/Bachelorarbeiten erweitert, verbessert oder integriert werden. Grundsätzlich ist eine Kommerzialisierung des Projekts verboten durch die Lizenz von *VideoPose3D* und *Detectron2*. Jedoch eine weitere Möglichkeit wäre, das Projekt als non-profit selbsterhaltend von Nutzenden finanzieren zu lassen, um die Kosten für das Hosting zu decken.

3.2.1 GitHub Wiki / Dokumentation

Auf *GitHub* wurde das «Wiki» Feature verwendet, um das Projekt kurz zu summieren und zu beschreiben. So können interessierte Aussenstehende einen Überblick über das Projekt erlangen, was es einfacher macht es zu verstehen und Anpassungen daran anzubringen.
Das Wiki ist folgendermassen aufgebaut:

- Home
  Kurzer Überblick über die Inhalte des Wikis
- Architecture
  Bietet eine Übersicht über den Aufbau des Projekts
- Database and Datastructure
  Definiert die Struktur der Datenbank und wie das Filesystem verwendet wird um Videos nach dem Hochladen an *Detectron2/VideoPose3D* zu übergeben.
- Developer Guidelines
  Beschreibt wie im Projekt vorgegangen werden soll, respektiv zum Umgang mit *Tickets*, *Commits* und *Pullrequests* und Installation für die Lokale Entwicklung
- Libraries
  Dokumentiert die wichtigsten Libraries für das Projekt
- Pose Estimation
  Dokumentiert, wie die *Pose Estimation* installiert und ausgeführt werden kann

3.3 Komponente

Grundsätzlich wurde beschlossen einen Server, der *VideoPose3D* ausführt, und eine Webapplikation, als *UI,* getrennt aufzubauen. Etwas granularer betrachtet besteht das Projekt namentlich aus 5 Komponenten:

- *Single Page Application Frontend*, dient als Userinterface (UI)
- *BFF-Web*, für Proxy der Webapplikation und als Security-Checkpoint.
- *Identity Server*, ist zuständig für Useradministration, Zugriffsrechte, Ein- und Ausloggen.
- *Core Backend Server*, als *API* die *Pose Estimation* Funktionalität bietet
- *Core Datenbank*

(Siehe Kapitel 5.1 Architektur für Erläuterung)

3.4 Technologien und Libraries-Entscheidungen

Für die einzelnen Teile des Projekts musste entschieden werden, anhand welcher Technologien entsprechende Umsetzung erfolgen sollte. Es ist ein Ziel des Projekts möglichst viele *Cutting-Edge* Technologien einzusetzen, deshalb ist es ein stark gewichtetes Kriterium bei jeder Entscheidung, wie modern eine Option ist. Eine Auflistung der verwendeten Libraries und Frameworks ist im Anhang zu finden.

3.4.1 Core Backend

Für den *Core Backend Server* bieten sich grundsätzlich 3 Sprachen an, diese Auswahl ergibt sich aus den im Team vorhandenen Erfahrungen und Know-how.

- C#
- Java
- Python

Python wird ausgeschlossen, weil die Performance, Wartbarkeit und Entwicklung eines *Python* Servers schlechter sind als die beiden anderen Optionen. Weiter kann *Python* nicht als Cutting-Edge erachtet werden. *Java* wird ausgeschlossen, da, obwohl *Java* in letzter Zeit weiterentwickelt wurde, die Sprache und deren

Ökosystem in den Augen des Teams zu *C#* und dem *.Net Core Framework* in den Punkten Lesbarkeit, Entwicklungseffizienz und Geschwindigkeit unterlegen sind. *C#* und das *.Net Core Framework* werden also verwendet, um das *Backend* umzusetzen. Diese Entscheidung unteranderem auch dadurch beeinflusst, dass *.Net Core Framework* im November 2022 ein Major Update zur Version 7 erhalten hat, welches die Performance verbessert und die Containerisierung von *.Net Core Applikationen* vereinfacht.

3.4.2 Core Datenbank

Die Aufgabe der Datenbank in diesem Projekt ist nicht gross. Es müssen die Resultate der verarbeiteten Files und Tags gespeichert werden. Für Futureproofing soll die Datenbank über mehrere Server skalierbar sein und Änderungen am Schema unkompliziert zulassen. Insbesondere wurden für dieses Projekt folgende Kriterien an eine Datenbank ausgearbeitet:

- Skalierbarkeit (Verteilbar über mehrere Server)
- Änderbarkeit des Schemas
- Cutting Edge
- Einfache Integration in das Backend
- File Storage
- Geringer Setup Aufwand

Durch diese Kriterien werden *SQL Datenbank* Technologien fast grundsätzlich ausgeschlossen. Heutzutage werden stattdessen *NoSQL-Datenbanken* weitgehend eingesetzt, da *NoSQL* eine horizontale Skalierung ermöglicht. Des Weiteren können *NoSQL Datenbanken* schneller und effizienter als *SQL-Datenbanken* sein und ermöglichen flexible Datenmodelle, welche sich leicht on-the-fly ändern lassen.

Für das Projekt wird *RavenDB* verwendet. *RavenDB* ist eine schemalose *NoSQL-Datenbank,* welche eine auf *SQL* basierende *Query Language* («*RQL*») bereitstellt, um mit der Datenbank zu interagieren. *RavenDB* ist eine der jüngsten Datenbanken, ist über *Cluster* (Verteilte Datenbank Server) vollkommen transaktionell und bietet eine native *C#* Integration in Form einer Library.

Als Alternativen wurden *Cassandra*, *MongoDB*, *CouchDB* und *Azure Cosmos DB* in Betracht gezogen.

3.4.3 Web Frontend

Das Frontend soll modern aussehen und einfach zu bedienen sein. Es gibt einige Optionen, womit ein modernes Webinterfaces gebaut werden kann:

- *React*
- *Angular*
- *Blazor*

… und weitere

Für den Grundbaustein des Frontend wurde *React* verwendet. *React* ist eine weitverbreitete und moderne *Javascript* Library mit welcher man ein *Single Page Application* (*SPA*) innerhalb ein paar Minuten bauen kann. Die Entscheidung für *React* wurde getroffen, weil das Team extensive Erfahrung mit *React* hat. Das Team hat auch bestehende Erfahrungen mit *Angular*, jedoch wurde gegen *Angular* als Framework entschieden, da es für dieses Projekt zu aufgebläht ist. *Blazor* ist eine brandneue Entwicklung im *.Net Framework*, aber kein Teammitglied hat mit *Blazor* Erfahrung. Deshalb wäre der Initialaufwand, *Blazor* zu lernen, zu gross.

Um auch ein modernes und userfriendly UI zu implementieren, entschied sich das Team eine *React* Komponenten Library zu verwenden. Es wurden die folgenden 3 Komponenten Libraries angeschaut und getestet:

- *Fluent UI*
- *MUI*
- *Devextreme*

Alle 3 Libraries haben ihre Stärken und Schwächen, jedoch nach dem jede Library einzeln getestet und analysiert wurde, fiel der Entscheid für *MUI*, da *MUI* eine sehr aktive User Base hat und vielzählige Komponente anbietet, welche unsere Anforderungen für das UI abdeckt. Welche Komponente gebraucht werden, konnte vor der Implementierung ermittelt werden mit einem Mockup (3.5). Da *MUI* auf die Design Prinzipien von «*Material Design»* aufbaut, kann auch sichergestellt werden, dass die einzelnen Komponenten modern und bekannt für die Nutzer wirken.

3.4.4 BFF-Pattern

Vor diesem Projekt war dem Team das *BFF-Pattern* unbekannt. Der Vorschlag ein *BFF-Pattern* anzuwenden, kam von externen Senior Software Engineers, die mit dem Team vernetzt sind. *BFF* steht für "*Backend For Frontend*" und ist ein Architekturmuster, das in der Web-Entwicklung verwendet wird. Es beinhaltet die Erstellung einer speziellen *Backend-Schicht* für jede *Frontend-Anwendung* oder jeden *Client*. Das *Backend-For-Frontend-Pattern* wird unter Softwareengineers immer beliebter, da es zahlreiche Vorteile für die Erstellung effizienter und skalierbarer Webanwendungen bietet. Die 3 relevantesten Vorteile für dieses Projekt sind (Duende, 2023; *Duende IdentityServer Terminology*, 2023):

1. Erhöhte Frontend-Unabhängigkeit: Mit einer *BFF-Schicht* können unabhängige Endpoints geschrieben werden, die ansonsten zu spezifisch für das *Core Backend* wären. Dies hilft damit, dass das *Core Backend* nicht zu überbläht wird.
2. Erhöhte Sicherheit: Dank des *BFF-Patterns* wird die Sicherheit erheblich erhöht, da *HTTPOnly Cookies* verwendet werden, die im Gegensatz zu "normalen" *Cookies* nur von einem Backend geändert und gelesen werden können und nicht von Frontend *Javascript* modifizierbar sind. Diese Architektur hat letztens an Popularität gewonnen da traditionelle Authentifizierungs- und Autorisierungsmethoden sich als anfällig für *Man-in-the-Middle* und Cookie Hijacking Attacken herausgestellt haben.
3. Erweiterbarkeit: *BFF* ermöglicht die Entwicklung von Anwendungen in verschiedenen Programmiersprachen für weitere *Clients*. Das bedeutet, dass in Zukunft weitere *Clients* hinzugefügt werden könnten, wie z. B. eine *Java-Android-Anwendung*, die jedoch problemlos in die bestehende Architektur integriert werden kann, ohne andere *Clients* zu beeinträchtigen.

Um die Architektur richtig einzusetzen, wurde das *Duende-BFF-Framework* verwendet und ein Identity Server, der auf dem *Duende-IdentityServer* basiert, implementiert. Die Implementationsdetails sind im Kapitel 5.1.1 beschrieben.

3.4.5 VideoPose3D

Da *VideoPose3D* und *Detectron2 Python-Programme* sind, werden diese auf dem *Backend* in einem *Python-Prozess* gestartet. *Python* kann nicht direkt im *.NET Framework* ausgeführt werden also wird mittels einer *Prozess-Funktion* ein *Python-Prozess* gestartet und auf dessen Terminierung gewartet.

3.4.6 Containerization

Die einzelnen Teile des Projekts werden in *Docker-Container* verpackt um diese mit wenig Aufwand auf *Host-Maschinen* zu deployen. *Docker* ist eine Art des *Applikations-Deployments*, welche in letzten paar Jahren an grosser Popularität gewonnen hat. Der Vorteil von Applikationen in einem *Docker-Container* ist, dass der *Container* als ein Image auf eine *Zielmaschine* heruntergeladen werden kann und unabhängig vom Betriebssystem oder den dort installierten Programmen ausgeführt werden kann. Dies liegt daran, dass *Docker-Container* eine Applikation mit all ihren Abhängigkeiten und einem minimalen Betriebssystem kapseln. So wird sichergestellt, dass wenn ein Programm in einem *Docker-Container* auf einer *Maschine* läuft, dass dies auf jeder *Maschine* problemlos läuft. Die *Hostmaschine* stellt lediglich den Zugriff auf einen *virtualisierten Kernel*, welcher bereit auf der *Docker-Container* alle Operationen ausführen kann. Da *Container* «lightweight» und isoliert sind und zum Ausführen keine *virtuelle Maschine* gestartet werden muss kann eine *Hostmaschine* mehrere *Docker- Container* mit minimalem Overhead ausführen.

Das Projekt besteht aus 5 Containern:

- *BFF-Web* mit *React Web-applikation*
- *Identity Server*
- *Microsoft SQL Server*
- *Core Backend API-Server*
- *RavenDB*

3.5 Mockups

Es wurden *Mockups* für das *Frontend* erstellt, um die Anforderungen für das UI zu ermitteln und den Aufwand schätzen zu können. Weiterhin dienten die *Mockups* dazu vorzustellen, welche *Features* wo angeboten werden und um sicherzustellen, dass die *Webapplikation* userfriendly und modern designt wird.

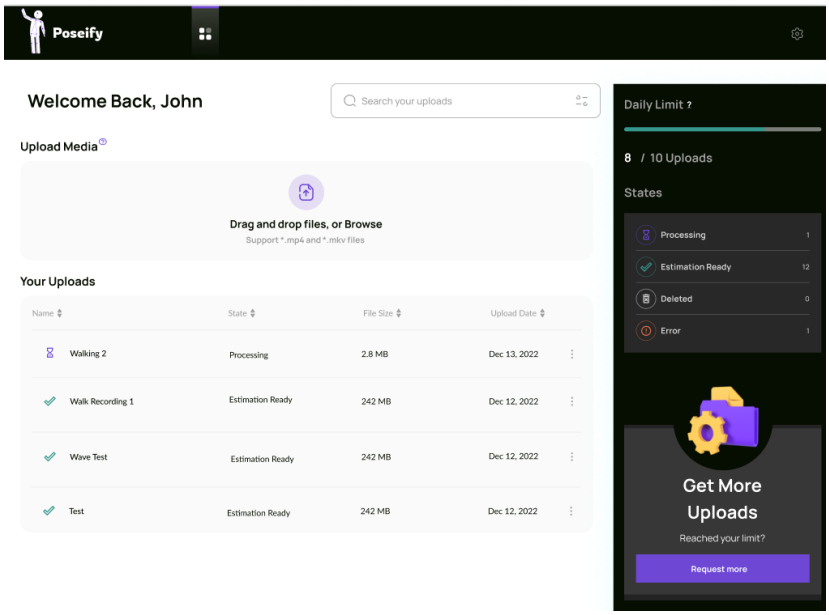


Abbildung 6: Erste Version Dashboard Mockup

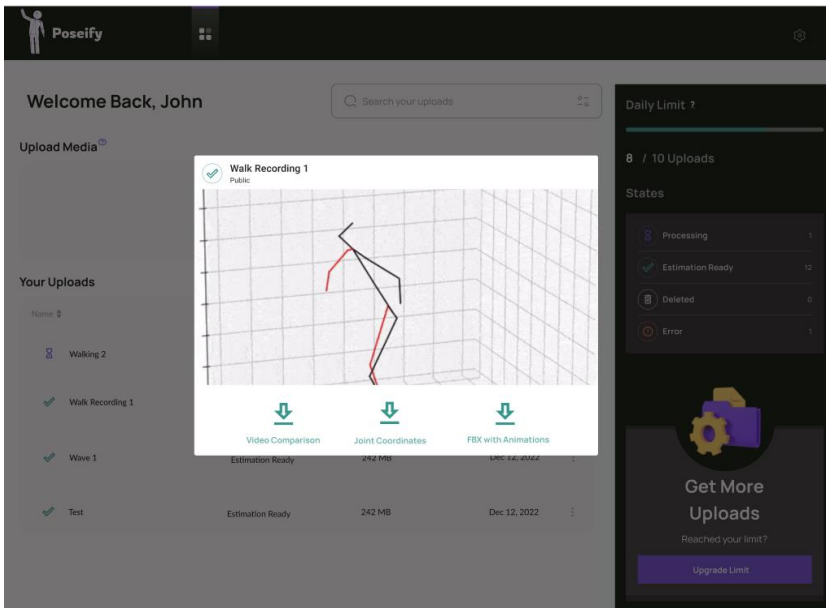


Abbildung 7: Erste Version Estimation View Mockup

Das *Webinterface* soll folgende Punkte umfangen:

- Liste mit den vom Nutzer gemachten *Pose Estimations*
- Vorschau eines Resultats
- Download eines Resultats
- Login und Logout

- Upload eines Videos
- Eine visualisierte Limite für die *Pose Estimations* pro nutzende Person, da eine *Estimation* relativ rechnungs- und zeitaufwändig ist und im Rahmen dieser Arbeit keine stark leistungsfähige Umgebung geplant ist.

3.6 Authentifizierung

Um Resultate nur den Nutzern zur Verfügung zu stellen, die diese auch hochgeladen haben, müssen diese nutzenden Personen eindeutig identifiziert werden können. Jedoch sind eine eigene *Useradministration* und *Zugangsdatensicherung* nicht trivial, weshalb dieses Projekt sich darauf beschränkt bereits authentifizierte Nutzende zu identifizieren und diese zum Zugriff eigens erstellten *Estimations* zu autorisieren. Als Hoheit für die *Authentifizierung* wurde Google gewählt, da Google-Accounts weit verbreitet sind.

4 Methoden

In diesem Kapitel wird die Arbeitsmethodik, das Zeitmanagement und das Controlling des Projekts beschrieben.

4.1 Agile Entwicklung

Für die Entwicklung des Projekts wird eine Mischung zwischen klassischer *Wasserfall-Methode* und *agilen* Ansätzen verwendet. Am Anfang werden *Meilensteine* definiert, bei deren Erreichung ein Subset der Features des Projekts erledigt sein sollen. Zwischen den *Meilensteinen* liegen jeweils 2-4 Wochen, welche als einzelne *Sprints* behandelt werden. Die einzelnen Anforderungen eines *Meilensteins* werden in Teilaufgaben zerlegt, welche mittels Tickets überwacht werden. Am Anfang eines *Sprints* wird ein *Sprint-Planning* gemacht, bei welchem *Tickets* an Teammitglieder zugewiesen werden, die in dem jeweiligen *Sprint* erledigt werden sollen. Am Ende eines *Sprints* wird ein *Sprint-Review* durchgeführt, wo die gelösten *Tickets* zusammen angesehen werden und die Ergebnisse schlussendlich mit der vorherig erledigten Arbeit vereint wird. Im *Sprint-Review* werden auch Probleme besprochen und anfallende Entscheidungen getroffen. Das *Sprint-Planning* für den nächsten Sprint erfolgt direkt nach dem *Sprint-Review* des letzten *Sprints*. *Tickets*, die durch Entscheidungen im *Sprint-Review* entstanden sind, können im *Sprint-Planning* einem *Meilenstein* zugeteilt werden oder zum Erledigen im anstehenden *Sprint* einem Teammitglied zugeteilt werden. Nach dem *Sprint-Planning* findet ein Meeting mit dem betreuenden Dozenten statt bei dem der aktuelle Stand und das weitere Vorgehen kurz besprochen wird.

4.2 Management Tool

Die Planung und Überwachung der *Sprints* und den *Meilensteinen* erfolgt über *GitHub* und dem Integrierten *Projekt-Management Tool*. Mit dem *GitHub-Projektmanagement Tool* können *Tickets* definiert und einem *Meilenstein* zugeordnet werden. In den *Tickets* werden die Akzeptanzkriterien definiert, welche erfüllt sein müssen, bevor ein *Ticket* abgeschlossen werden kann. Wenn in einem *Ticket* definiert ist, dass Code erforderlich ist, um es abzuschliessen, dann wird das *Ticket* generell mit einem *Pullrequest* von einem *Feature Branch* auf den *Main Branch* abgeschlossen. *GitHub* ermöglicht das Schliessen von *Tickets* innerhalb eines Projekts, wenn bei einem *Pullrequest* das *Ticket* angegeben wird, wenn der *Pullrequest* zusammengeführt wird, wird das spezifizierte *Ticket* geschlossen.

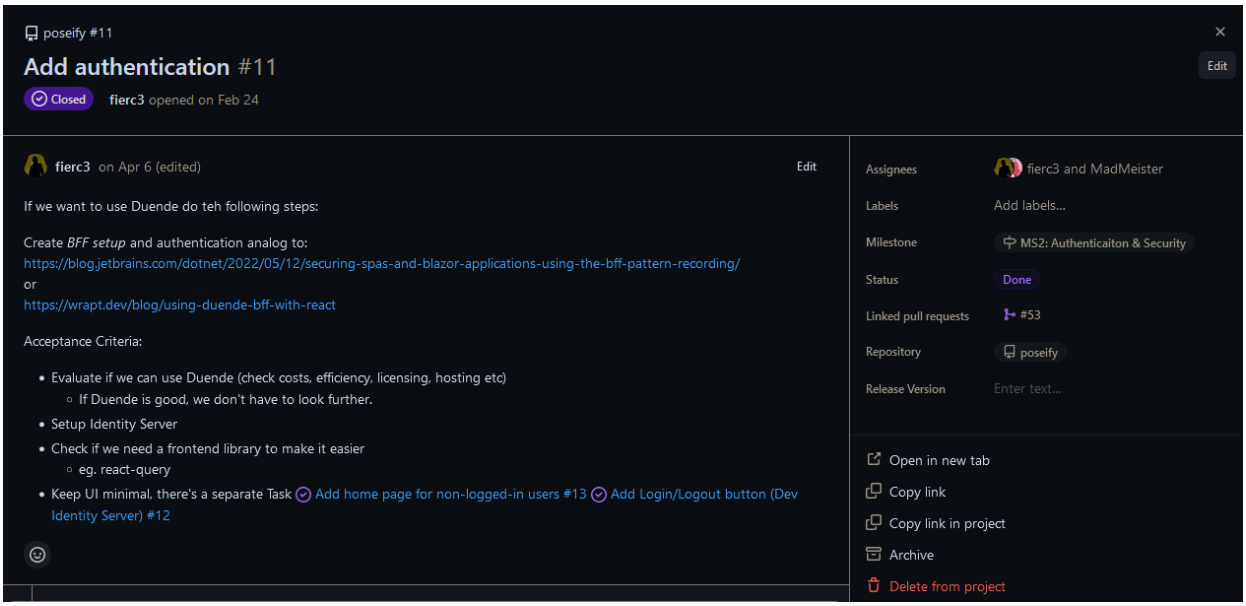


Abbildung 8: Beispiel Ticket Github

4.3 Git Repository

Mittels *GitHub* und einem *Git-Client* wird das *Code-Versioning* gemacht. Code wird auf *Feature-Branches* comitted, welche meistens an ein *Ticket* gebunden sind. Wenn ein *Feature-Branch* abgeschlossen ist, wird ein *Pullrequest* erstellt, welcher den Code des *Feature-Branches* in den *Main-Branch* einfügt. Ein *Pullrequest* muss immer von einem anderen Teammitglied als dem Teammitglied überprüft und genehmigt werden, welches den *Pullrequest* erstellt. Wenn ein *Pullrequest* genehmigt wurde kann der Code des *Feature-Branches* mit dem *Main-Branch* zusammengeführt werden.

Innerhalb des *Code Repositories* sollen für *Commits* die “Git Commit Best Practices” eingehalten werden. Diese umfassen die folgenden Punkte (simplifiziert) (Matos, 2023):

- Commit Related Changes
- Commit Often
- Don't Commit Half-Done Work
- Test Your Code Before You Commit
- Write Good Commit Messages

4.4 Meilensteine

Es wurden 6 *Meilensteine* definiert, durch welche beschrieben wird in welchem Zeitrahmen welche *Features* und *Arbeiten* erledigt werden sollen. Die *Meilensteine* sind nicht zwingende Deadlines aber sollten ungefähr eingehalten werden damit das Projekt im definierten Zeitrahmen fertiggestellt werden kann.
Die *Meilensteine* sind so aufgeteilt:

- MS1: Setup
  - Grundsätzliches aufsetzen der Projektstruktur
  - *VideoPose3D* testen
  - Libraries aussuchen
- MS2: Authentication & Security
  - Frontend Aufsetzen
  - Userlogin erstellen
  - Datenbank Struktur definieren
- MS3: Core Features
  - Einbindung *VideoPose3D*
  - Endpoints auf *API* um mit Datenbank zu interagieren und *Estimations* zu machen
  - Frontend Für Upload, Download von *Estimations*
- MS4: Deployment
  - Dockerisation
  - Vom Internet erreichbar machen für Usertests
- MS5: Feedback, Cleanup and Optional Features
  - Bugfixes und Features basierend auf User Feedback
  - Extra-Features wenn Zeit vorhanden ist
- MS6: Submission & Publishing
  - Ein 1.0 Release
  - Dokumentation beenden und abgeben

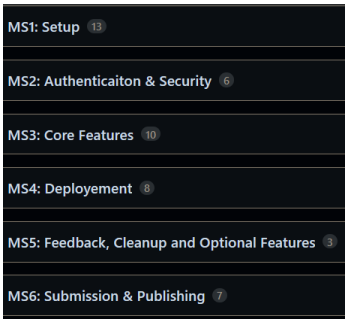


Abbildung 9: Meilensteine auf GitHub (Pullen & Kirchhofer, 2023b)

4.5 Entwicklungsumgebung

Für die Frontendentwicklung mit *React* und *Typescript* wird mit *Visual Studio Code* gearbeitet. *BFF*, *Identity Server* und das *Core Backend* wird mit *Visual Studio 2022* erstellt. Für *Python* wird *PyCharm* und für die Interaktion mit *GitHub* der «*Git Fork Client*» und das *GitHub-Webinterface* verwendet.

4.6 Teststrategie

Während der Entwicklung wird kontinuierlich getestet. Wenn ein neues *Feature* implementiert wird, wird es auch auf der eigenen *Maschine* und manchmal auf einer anderen *Maschine* getestet, bevor es auf den *Main-Branch* zugelassen wird. Das Projekt im Endstadium wird von dem betreuenden Dozenten, eventuell ein paar Studenten/innen der HSLU und Freunden des Teams getestet.

Den Testern wird die URL mitgeteilt, mit welcher sie auf das Projekt zugreifen können. Die Tester werden dann instruiert das *User-Interface* zu navigieren, einige Videos hochzuladen und die Ergebnisse anzuschauen und herunterzuladen. Nach dem Test wird ein *Feedback* von den Testern entgegengenommen und evaluiert. Es wurde gegen *Unittests* entschieden, da es keine gute Logik-Blöcke gibt, um Tests dafür zu schreiben. *Integrationstest* für das Zusammenspiel zwischen den einzelnen *Komponenten* hätte Sinn gemacht, war aber aus zeitlichen Gründen nicht realisierbar.

4.7 Arbeitsjournal

Das Arbeitsjournal basiert auf den *Git-Commits*, die in den *Repositories* von *Poseify*, *PoseifyIdentityServer*, *VideoPose3d_Poseify* und *detectron2-python-37* erstellt wurden, sowie auf Arbeiten ausserhalb des Codes, die separat über *GitHub Issues* oder in *Onedrive* in einem Word-Dokument verfolgt werden.

Das *Arbeitsjournal*, sowie die Links auf die *GitHub-Repositories*, können im Anhang gefunden werden (A.3 und A.4)

5 Realisierung

Mit dem vorherigen Kapitel sollten die einzelnen Entscheidungen genauer beschrieben worden sein, hier wird im Detail beschrieben, basierend auf diese Entscheidungen, wie das Projekt realisiert worden ist. Mit Fokus auf die einzelnen Komponenten die vom Team implementiert worden sind.

5.1 Architektur

Bevor ein Code geschrieben werden konnte, musste eine robuste Architektur entschieden werden. Mit dem Ziel das *BFF-Pattern* anzuwenden und *VideoPose3D* als *Pose Estimator* zu verwenden, wurde nach ein paar Iterationen diese Architektur entworfen:

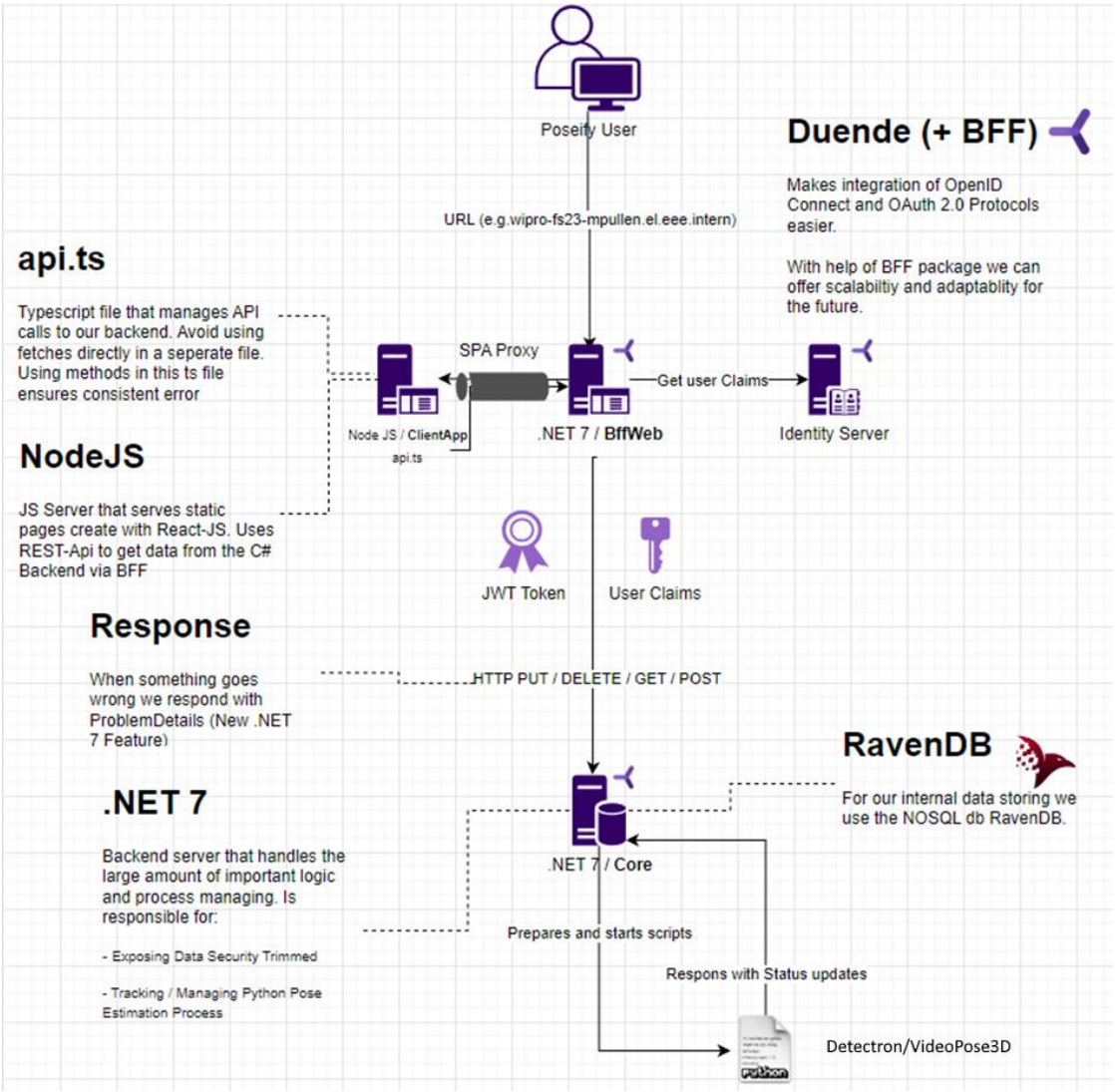


Abbildung 10: Architekturdiagran Poseify: (Pullen & Kirchhofer, 2023a)

Die *Webapplikation* dient als das *UI* des Projekts. Durch die *Webapplikation* kann die nutzende Person Dateien hochladen und Resultate herunterladen. Mit dem *Identity Serve*r wird der Zugriff auf den *Backend-Server* und somit die *Datenbank* und den *Pose Estimation* Algorithmus reguliert. Nur eine eingeloggte Person kann *Pose Estimations* ausführen und Ergebnisse seiner eigenen oder öffentlichen *Estimations* herunterladen. Für die Implementation der *Authentifizierung* und *Autorisierung* mit dem *Identity Server* wird das *BFF-Pattern* (*Backend for Frontend*) eingesetzt. Der *BFF-Server* in diesem Projekt öffnet einen Proxy zum *UI* (*ClientApp*), welches via *Node.js* als *Single Page Applikation* läuft. Alle Anfragen an das *Core Backend* von der *ClientApp* aus gehen über den *BFF-Server,* der mit dem *Identity Server* sicherstellt, dass nur berechtigte Zugriffe stattfinden. Der *Backend-Server* ist das Programm, welches die *Pose Estimation* ausführt und Zugriff auf eine *Datenbank* hat, auf welcher die Ergebnisse gespeichert werden. Das *Backend* übernimmt das

Überreichen einer empfangenen Datei an den *Pose Estimation* Algorithmus, das Speichern von *Estimation*-Resultaten in der Datenbank und das Abholen von *Estimation*-Resultaten von der Datenbank zum Herunterladen. Das Projekt soll in *Docker-Containern* verfügbar sein, damit das *Deployment* möglichst umgebungsunabhängig erfolgen kann.

5.1.1 BFF-Web

Der *BFF-Server* verwendet die *Duende-BFF-Framework* Library. Diese Library automatisiert das *Security Token-handling* und stellt Methoden bereit mit der die *API-Endpoints* auf dem *Core Backend* von unberechtigten Zugriffen geschützt werden können. Alle Zugriffe auf das *Core Backend* müssen via dem BFF-Web erfolgen. Ein *Client* hat nie direkten Zugriff auf das *Core Backend*, jeder Zugriff wird von *BFF-Web* ausgeführt. Mit dem *BFF-Pattern* werden die *Access-Tokens* im Browser als *HTTPOnly-Cookie* gespeichert, dies ist ein verschlüsseltes und signiertes *Cookie*. Ein *HTTPOnly-Cookie* lässt sich nicht trivial im *Client* auslesen oder modifizieren, nur der *BFF-Server* kann das Cookie lesen und verändern. Das *HTTPOnly-Cookie* wird vom *Client* mit jeder Anfrage mitgegeben und der *BFF-Server* kann damit den *Client* identifizieren und Zugriffe auf den *Core Backend-Server* ausführen. Wenn ein *Access-Token* abgelaufen ist oder eine Person sich einloggt wird der *Identity Server* damit beauftragt die Person zu authentifizieren und ein neues *Access-Token* für diesen auszustellen. Damit können Attacken erschwert und sogar verhindert werden. Durch das *BFF-Pattern* muss auf dem *Client* kein *Token Management* errichtet werden, eine nutzende Person muss lediglich an den *Identity Server* weitergeleitet werden, um sich einzuloggen. Es können mehrere *BFF-Server* parallel vorhanden sein, jeder dieser Server kann eine andere Art von *Frontend* bedienen. So könnte zum Beispiel ein *BFF-Server* für ein *UI* in der Form eines *Mobile Apps* erstellt werden.

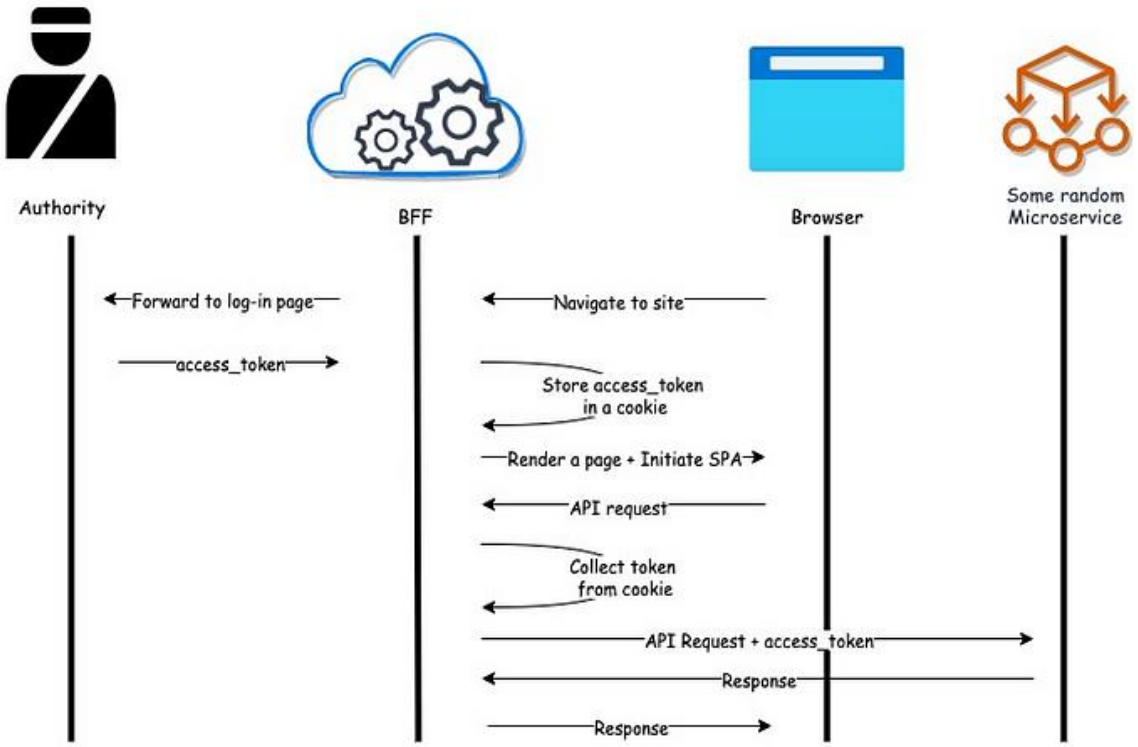


Abbildung 11: BFF Authentication Sequence (Starreveld, 2022)

5.1.2 ClientApp

Die Komponente, die für die visuelle Darstellung zuständig ist und als einziger Interaktionspunkt für den Nutzenden dient, wird als *Client-App* bezeichnet. Sie wurde mit Hilfe der Sprache *Typescript* und der Rendering-Bibliothek *React* entwickelt. Dieser Code im *Typescript-Format* wird durch Transpilierung in *Javascript* umgewandelt und mit *NodeJS* über den von *BFF-Web* erstellten *SPA-Proxy* zugänglich gemacht. Die Ordneranordnung unterteilt den *Typescript-Code* in folgende Kategorien:

- Src
    - Components
        - Alle erstellten Komponenten werden in diesem Ordner gespeichert. Die folgenden *React-Komponenten* wurden erstellt: estimations-list, estimation-view, navbar und upload-button. Die Aufteilung in Komponenten hat den Vorteil, dass sie effizienter gerendert werden kann und die Codeverwaltung einfacher ist.
    - Helpers
        - Der gesamte Code, der keine *React-Komponente* ist und wiederverwendet werden kann, wird in diesem Ordner gespeichert. Meistens für *React Hooks* und *Typescript-Schnittstellendefinitionen* verwendet.
    - Pages
        - Alle Seiten, die besucht werden können, werden hier gespeichert. Auch wenn es sich bei *React* um eine *SPA*-basierte Bibliothek handelt, können mit Hilfe von *React-Router* separate Seiten mit eindeutigen URLs erstellt werden, die den Zugriff auf verschiedene Teile der Webanwendung wie poseify.com/dashboard möglich machen
    - Themes
        - Alle verwendeten Themen werden in diesem Ordner gespeichert. Für zukünftige Implementierungen könnten weitere Themen erstellt werden, z. B. wenn eine Person eine private Version dieser Software hosten möchte, könnte er das Thema ändern, um es an seine Designrichtlinien anzupassen.

Die beiden Hauptaufgaben des *Frontends* sind der Aufruf der *API* und die effiziente und verständliche Darstellung der Daten und Aktionen in einem modernem UI.

5.1.2.1 API Calls

Der Zugriff auf die *API* ist die einzige Möglichkeit, Daten aus der Datenbank zu erhalten oder auf die im Core implementierte Logik zuzugreifen. Um effiziente und intelligente *API-Aufrufe* zu ermöglichen, wurde eine Kombination aus der *React-Query-Bibliothek* und *Axios* verwendet.

Axios bietet eine vereinfachte und effektive Schnittstelle für die Verwaltung asynchroner Anfragen und Antworten. In diesem Projekt liegt der Hauptvorteil in der unkomplizierten Implementierung, die auch die Lesbarkeit verbessert. Der Aufruf von Endpunkten wird schnell und einfach, vor allem durch die Unterstützung von *Javascript-Promises*, was insgesamt zu einem saubereren Code führt. Andererseits ist *React-Query* eine mächtige Bibliothek zum Abrufen und Zwischenspeichern von Daten. Sie vereinfacht die Verwaltung von *Remote-Daten* durch die Bereitstellung von *Hooks*, die das Abrufen, Zwischenspeichern und Synchronisieren von Daten mit minimaler Konfiguration ermöglichen. In diesem Projekt werden nur die Grundlagen dieser umfangreichen Bibliothek verwendet, aber sie bietet dennoch enorme Vorteile, insbesondere bei *API-Aufrufen*, die wiederholt werden müssen, um kontinuierliche Änderungen im *Backend* anzuzeigen (z. B. Zustandsänderungen von *Estimations*).

Beides zusammen führt zu leichtgewichtigen *React-Hooks*, die für jeden Endpunkt erstellt werden können, für den eine *Refetching*- oder *Caching-Logik* erforderlich ist.

```
const fetchData = async () =>
  axios.get('/api/GetUserEstimations', config)
  .then((res) => res.data as IEstimation[]);

function useEstimations() {
  return useQuery(
    claimsKeys.claim,
    async () => fetchData(),
    {
      staleTime: Infinity,
      cacheTime: Infinity,
      refetchInterval: 1000 * 15,
    }
  )
}
```

Abbildung 12: Fetch Beispiel mit React-Query und Axios

5.1.3 UI

Schon früh in der Evaluierungsphase der Bibliotheken fiel die Entscheidung auf *MUI*, eine *React-Bibliothek*, die *Material Design*-konforme Komponenten bietet und damit den Entwicklungsaufwand erheblich reduziert. Die wichtigsten verwendeten Komponenten sind im folgenden Screenshot hervorgehoben.

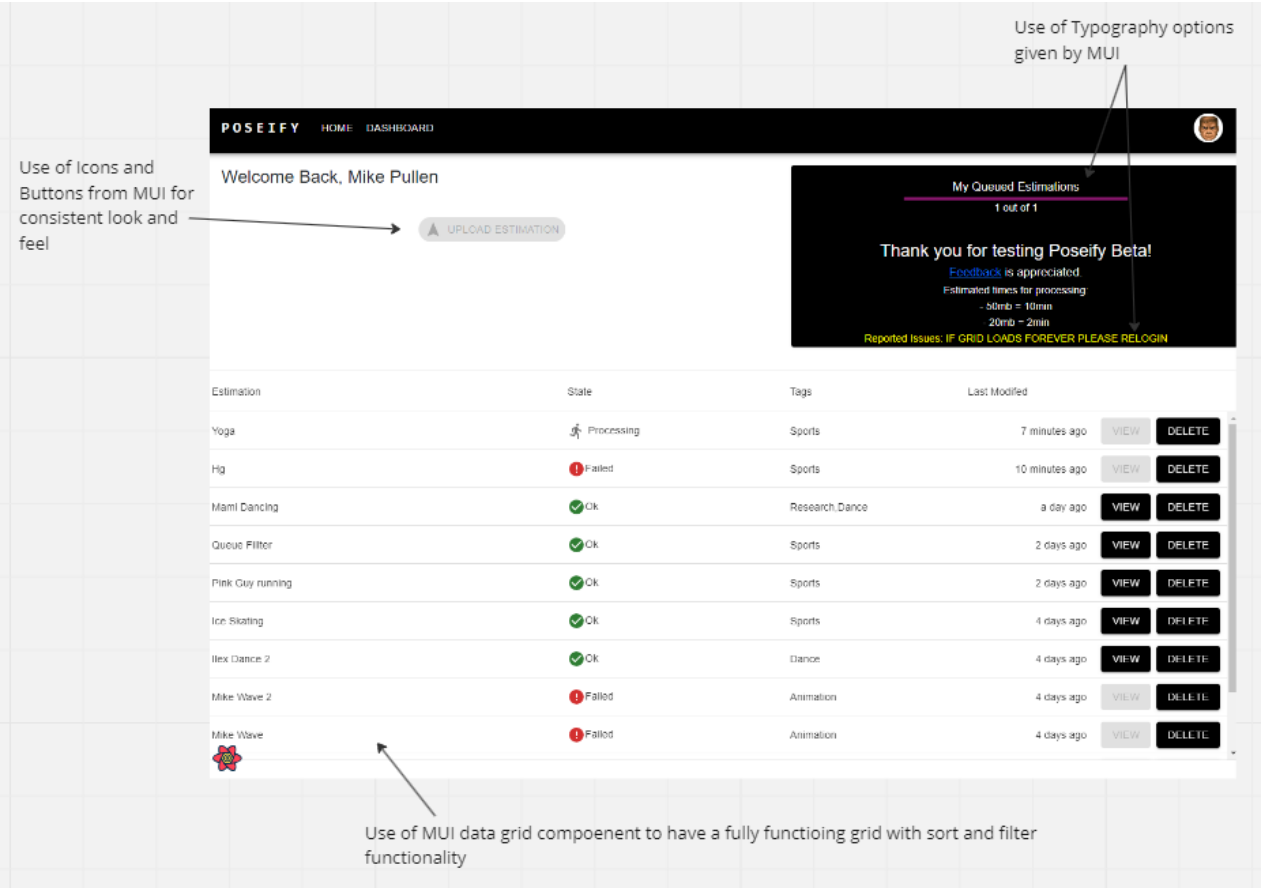


Abbildung 13: Verwendete MUI-Komponente

5.1.4 Core

Der *Core* Teil des Projekts ist die *Backend-API*. Das *Core Backend* ist zuständig für die Interaktion mit *VideoPose3D* und der Datenbank. Von hier aus werden *Python-Prozesse* gestartet, die mit *FFMPEG*, *Detectron2* und *VideoPose3D* ein hochgeladenes Video in ein Preview und ein «.npz» File umrechnen. Das Resultat und der Preview werden in der Datenbank gespeichert. Das *Core Backend* ist modular aufgebaut in folgende klar funktionsgetrennte Ordner.

- Core
    - Controllers
        - Die *API-Endpoints*, die vom *BFF-Server* angesteuert werden. Von den *Controllern* werden die Funktionalitäten des *Core Backend* ausgelöst.
    - DB
        - Die Klasse wo die Datenbank kennt, durch eine Instanz dieser kann mir der Datenbank interagiert werden.
    - Models
        - Klassen, welche die Datenstruktur von Objekten innerhalb des *Cores Backend* und der Datenbank abbilden.
    - PoseEstimation
        - Beinhaltet die Installation von *Detectron2* und *VideoPose3D*. Für jede *Estimation* die ausgeführt werden soll wird ein eigener *Python-Prozess* gestartet, welcher das «*estimate_pose.py*» Skript ausführt (siehe 5.1.7).
    - Services
        - Hier wird die *Pose Estimation* ausgeführt, mit der Datenbank interagiert und Uploads in einer Queue gehalten, wenn der *Pose Estimation* Dienst ausgelastet ist.

Die *Controller* und *Services* implementieren *Interfaces*, damit diese austauschbar sind.

5.1.4.1 Controller

Um einen guten Überblick über die *Controller* zu haben, musste eine gewisse Abstraktion zwischen den *Controllern* geschaffen werden. Jeder *Controller* ist gekapselt, so dass jeder *Controller* unabhängig laufen kann.

5.1.4.1.1 EstimationController

Stellt alle Bearbeitungs- und Auslesemöglichkeiten für *Estimations* zur Verfügung.
- **GetUserEstimations**: Alle *Estimations* des eingeloggten Nutzenden werden zurückgegeben
- **DeleteEstimation**: Mit einem Parameter namens «*estimationId*» kann eine *Estimation* gelöscht werden

5.1.4.1.2 UploadController

Verantwortlich für den Video-Upload und die Zwischenspeicherung im Dateisystem.
- **PostUpload**: Liest das Video als *FormsFile* und speichert es im Upload-Verzeichnis

5.1.4.1.3 AttachmentController

Stellt *Attachments* zur Verfügung, die in der *RavenDB* gespeichert sind.
- **GetAttachment**: Mit den Parametern «*estimationId*» und «*attachmentType*» wird ein bestimmtes *Attachment* von einer bestimmten *Estimation* heruntergeladen

5.1.4.2 Services

Damit die meiste Logik der Software nicht an die *Controller* gekoppelt ist, wurden die folgenden *Services* erstellt.

5.1.4.2.1 EstimationService

Die Klasse *EstimationService.cs* ist der Teil des *Core Backends*, in welchem die *Pose Estimation* ausgeführt und mit der Datenbank interagiert wird. Hier werden *Estimation*s in der Datenbank angemeldet, gespeichert und zum Herunterladen abgeholt. Bevor eine *Estimation* gestartet wird, wird ein Eintrag dafür in der Datenbank generiert und dieser Eintrag als «Processing» gekennzeichnet. Eine *Estimation* im Status «Processing» wird an das *Python-Skript* «*estimate_pose.py*» übergeben und dieses wird gestartet. Erst wenn der *Python-Prozess* terminiert ist, wird der Status auf «Success» oder «Failed» gesetzt, je nachdem ob die *Estimation* erfolgreich durchgeführt werden konnte.

5.1.4.2.2 QueueService

Ein Service, der neue *Estimations* in eine In-Memory Queue schiebt und der Status als «Queued» kennzeichnet, wenn die GPU ausgelastet ist. Sobald die GPU nicht mehr ausgelastet ist, wird die *Estimation* auf Status «Processing» gesetzt und die Verarbeitung der *Estimation* auf der GPU wird gestartet.

5.1.4.3 Python Estimation Skript

Das «*estimate_pose.py*» Skript vereinigt das Ausführen von *Detectron2*, die Datenaufbereitung für *VideoPose3D* und *VideoPose3D* in einem Skript mit allen nötigen und optionalen Argumenten. So muss nur ein einziger *Python-Prozess* für eine *Pose Estimation* ausgeführt und auf dessen Terminierung gewartet werden.
Das Skript kann 6 Argumente annehmen:

Argument	**Format**	**Beschreibung**
--dir	Absoluter oder Relativer Ordnerpfad als String (z.B. C:\Uploads\)	Upload-Ordner
--user-id	Unique Identifier einer nutzenden Person als String (z.B. 1234)	Unterordner im Upload-Ordner
--guid	Unique Identifier einer Datei als String (z.B. 5678)	Datei im Unterordner
--file-ext	Dateiendung als String (z.B. .mp4)	Dateiendung der Input-Datei

--new-file-ext	Ziel-Dateiendung als String (z.B. .mp4)	Wenn Input-Datei vor *Pose Estimation* in anderen Dateityp konvertiert werden soll (optional)
--scale-fps	FFMPEG Framerate Skalieren als Boolean (z.B. True)	Skaliert Framerate der Input Datei auf 50 FPS (optional)

Tabelle 2: estimate_pose.py Argumente Liste

Siehe Kapitel 5.1.4.4 für Erklärung der Ordnerstruktur.

Das Skript wird zuerst, mithilfe von *FFMPEG,* die Input-Datei in das in «--new-file-ext» spezifizierte Dateiformat konvertieren und dann die Framerate skalieren, wenn das «--scale-fps» Argument True ist. Hierbei interpoliert oder entfernt *FFMPEG* Frames aus dem Video. Dies ist eine Option da *VideoPose3D* auf 50 FPS Videos trainiert wurde und deshalb 50 FPS Videos zu besseren Ergebnissen führen können. Danach wird *Detectron2* ausgeführt, die Daten für *VideoPose3D* aufbereitet und zuletzt *VideoPose3D* ausgeführt.

5.1.4.4 File-Management

Video-Uploads sind auf maximal 50 Megabyte Grösse beschränkt. Dies wurde so festgelegt, damit der *Core Backend Server* nicht überlastet wird und den *Service* zum Halten bringt. Wenn auf der Umgebung, auf der das *Core Backend* Projekt läuft, *CUDA* verwendet werden kann und dadurch der *Pose Estimation-Prozess* erheblich beschleunigt wird, kann diese Limite angepasst werden. Hochgeladene Videos werden im lokalen Filesystem abgelegt, damit das *Python-Script* für *VideoPose3D* einfachen Zugriff darauf erhält. Der Zielort von Videos ist in der *Konfigurationsdatei* des *Core Backend Projekts* anpassbar und setzt sich wie folgt zusammen:

- **[***UploadDirectory* (aus appsettings.json)**]/[***Nutzer-ID***]/[***Datei-ID***].[***Dateiendung***]**

Der Speicherort von einer empfangenen Datei wird dem *Pose Estimation Skript* mitgeteilt. Egal, ob erfolgreich oder fehlgeschlagen, wenn das *Skript* terminiert ist, wird die ursprüngliche Videodatei aus dem Filesystem gelöscht.

5.1.5 Identity Server

Der *Identity Server* basiert auf der *Duende IdentityServer Library* für *.Net 7*. «*Duende IdentityServer* ist eine *OpenID Connect* und *OAuth Engine* – Die Engine implementiert die *OpenID Connect* und *OAuth 2.0 Protokoll-Familien*.»[3] *OpenID Connect* ist ein *Identity-Layer*, welcher auf *OAuth 2.0* aufbaut. OAuth 2.0 ist ein *Industrie Standard Protokoll* für Autorisierung. Mit *OpenID Connect* können Identitäten von authentifizierten Personen verifiziert werden. Der *Identity Server* besitzt eine *SQL-Datenbank* in der verschlüsselte Userdaten persistiert werden, damit die nutzenden Personen unseres Projekts permanent identifiziert werden können. Eine Anmeldung kann im Moment nur per Google erfolgen. Die Sicherung von Benutzerdaten, wie das sichere Speichern von Passwörtern, ist eine Aufgabe, die ausserhalb des Umfangs dieses Projekts fällt. Google ist also die *Authentifizierungsautorität* und der *Identity Server* stellt lediglich sicher, dass eine Person zuverlässig identifiziert werden kann, damit *Pose Estimation* Resultate dieser Person zugeordnet werden können.

5.1.6 RavenDB

In der Datenbank werden die Resultate von *Estimations* gespeichert. Ein «*Estimation*» Eintrag wird für jede hochgeladene Datei erstellt. Das Attribut «*DisplayName*» wird von der Input-Datei genommen und dient zur Identifizierung des Resultats. Mit «*EstimationState*» wird verfolgt ob eine *Estimation* fehlgeschlagen, erfolgreich oder noch in Bearbeitung ist. «*UploadingProfile*» ist der *Unique-Identifier* der nutzenden Person, welcher die Ursprungsdatei hochgeladen hat. Das Resultat und die Vorschau des Resultats werden als Attachment angefügt.

[3] «Duende IdentityServer is an OpenID Connect & OAuth engine - it implements the OpenID Connect and OAuth 2.0 family of protocols.» (*Duende IdentityServer Terminology*, n.d.)

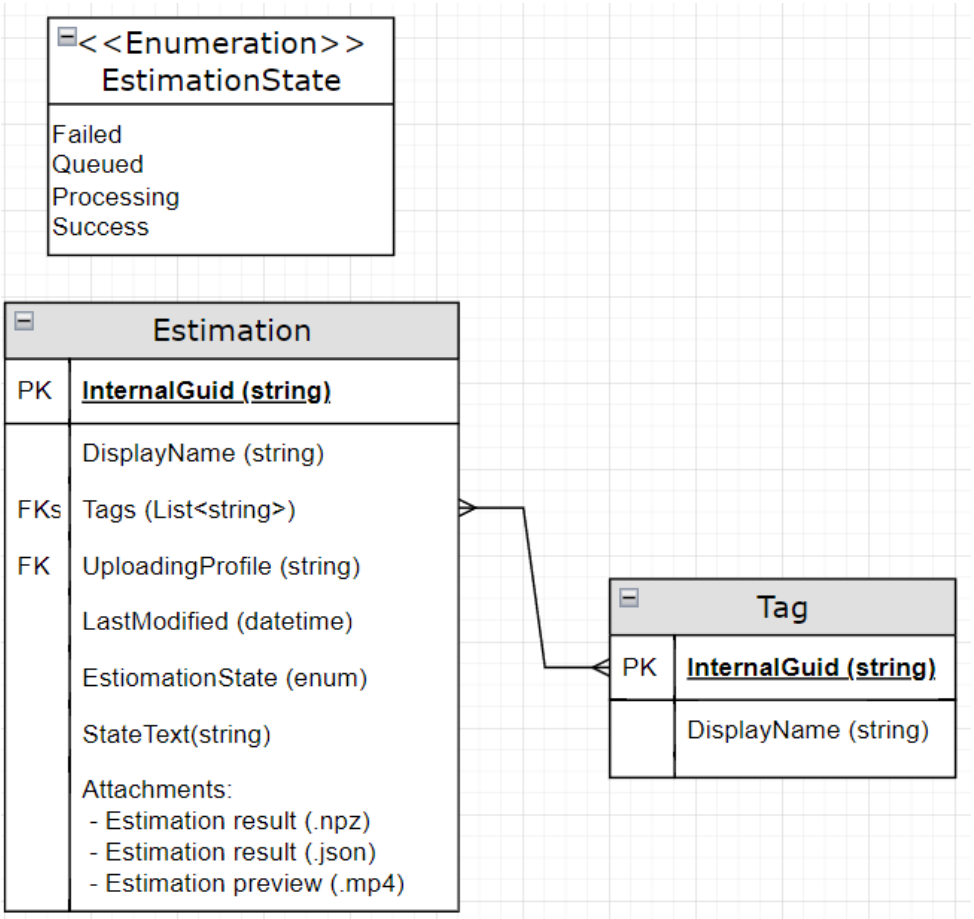


Abbildung 14: Datenbank Struktur Poseify (Pullen & Kirchhofer, 2023b)

5.1.7 Detectron2/VideoPose3D

Es wird ein *Python-Skript* verwendet, welches den ganzen Prozess von der Eingabe des Videos zum *Estimation*-Resultat abhandelt. Das Skript unifiziert den Prozess, welcher auf dem *GitHub Repo* von *VideoPose3D* beschrieben ist, um die *Pose Estimation* durchzuführen.

1. Optionales Preprocessing mit *FFMPEG*
    a. Videoformat ändern
    b. Framerate standardisieren
2. *2D Joint-Position* Inference mit *Detectron2*
3. Datenaufbereitung
4. *3D Joint-Positions* generieren mit *VideoPose3D*

Im *Core Backend Projekt* wird dieses *Skript* in einem eigenen *Python-Prozess* aufgerufen mit dem Ordnerpfad zur Input-Datei. Wenn das *Skript* fertig ist, legt es das Ergebnis im gleichen Pfad mit dem Suffix «*_result*» ab. Die Integration von *VideoPose3D* erfolgt mittels eines *Python-Skripts,* welches auf dem *Core Backend Server* als eigener Prozess gestartet wird.

5.2 Deployment und Installation

Ziel dieses Abschnitts ist es, Anweisungen für die Einrichtung einer lokalen Instanz des Projekts zu geben. Alle weiteren Anforderungen, die mit der Bereitstellung von Test- oder Produktionssystemen verbunden sind, wurden weggelassen, da sie sich erheblich unterscheiden können.

Im Folgenden sind die Voraussetzungen aufgeführt, die auf dem lokalen System installiert sein müssen:

- Python 3.8.0
- Anaconda
- .NET 7
- CUDA 11.7

Um die globale Python-Installation nicht zu beeinträchtigen, ist es logisch, ein *Anaconda Environment* einzurichten und nur in diesem zu arbeiten. In den App Settings des *Core Backend* kann der *Python-Pfad* geändert werden. Um *CUDA* ausführen zu können, müssen die erforderlichen *Pytorch-Pakete* mit folgendem Befehl installiert werden. Es sei jedoch darauf hinzuweisen, dass zuerst zu überprüfen ist, ob die Grafikkarte *CUDA* unterstützt:

conda install pytorch torchvision torchaudio pytorch-cuda=11.7 -c pytorch -c nvidia

Um den Rest der *VideoPose3D*-Installation zu erleichtern, wurde ein Installationsskript geschrieben (A.5). Es installiert *FFMPEG*, *Detectron2* und *VideoPose3D*, die alle für die Ausführung von *Pose Estimation* benötigt werden. Um sicherzustellen, dass *Detectron2* und *VideoPose3D* mit diesem Setup funktionieren, wurden 2 Forks erstellt, die ein paar Änderungen vorgenommen haben, so dass sie problemlos in diesem Projekt verwendet werden können.

Sobald das Installationsskript durchgeführt worden ist, ist der nächste Schritt, die *Poseify .NET Solution* in Visual Studio zu öffnen und die App Settings zu aktualisieren.

- App Settings im Core
  - EstimationScriptLocation: Speicherort von *estimate_pose.py*
  - UploadDirectory: Verzeichnis, in dem Uploads zwischengespeichert werden, bevor sie verarbeitet und in der Datenbank gespeichert werden
  - OverridePythonVersion: Pfad zu gewünschter *Python* Version (*Anaconda environment*)

Bevor die Solution gestartet werden kann, müssen einige Docker-Images ausgeführt werden. Folgende Ausführungen sind die offizielle *Poseify* Docker-Installationsanleitung, die auch auf *GitHub* in englischer Sprache zu finden sind:

Schritt 1: Command line wählen
Öffnen von *Powershell* oder andere bevorzugte Befehlszeile. TIPP: Die folgende Installationsanleitung wurde nur unter Windows getestet, sollte aber auch unter Linux funktionieren (die *Powershell-Befehle* müssen in die Sprache einer anderen Shell übersetzt werden, wenn *Powershell* nicht verfügbar ist).

Schritt 2 Raven DB starten mit Docker

1. Raven DB Eula akzeptieren
   Setup args $rvn_args = "--Setup.Mode=None --License.Eula.Accepted=true"
2. RavenDB auf dem korrekten Port starten
   Start docker on port 6969 docker run -p 6969:8080 -e RAVEN_ARGS=$rvn_args ravendb/ravendb

Schritt 3 Identity Server starten mit Docker

1. HTTPS einstellen
   dotnet dev-certs https -ep $env:USERPROFILE.aspnet\https\aspnetapp.pfx -p dotnet dev-certs https –trust

2. Ein Docker Netzwerk einrichten, um die Kommunikation zwischen Docker Containers zu erlauben.
   docker network create --subnet=172.18.0.0/16 poseify-net
3. MS SQL Server starten.
   docker pull microsoft/mssql-server-linux:2022-latest
   docker run --net poseify-net --ip 172.18.0.18 -e "ACCEPT_EULA=Y" -e "MSSQL_SA_PASSWORD=<pw>" -p 1433:1433 -d mcr.microsoft.com/mssql/server:2022-latest
4. Migrations script ausführen für die Datenstruktur (A.6)
5. Docker container starten im richtigem Netz und mit HTTPS (image momentan nicht öffentlich, muss direkt angefragt werden)
   docker run --net poseify-net --ip 172.18.0.19 --rm -it -p 8000:80 -p 8001:443 -e ASPNETCORE_URLS="https://+;http://+" -e ASPNETCORE_HTTPS_PORT=8001 -e ASPNETCORE_Kestrel__Certificates__Default__Password="<pwhttps>" -e ASPNETCORE_Kestrel__Certificates__Default__Path=/https/aspnetapp.pfx -v $env:USERPROFILE\.aspnet\https:/https/ identityserverposeifydev:1.0.0

Schritt 4 Frontend vorbereiten

1. Zu *ClientApp* Ordner wechseln
2. NPM Packages installieren (npm install)

Wenn diese Schritte durchgeführt wurden, können *BFFWeb* und *Core* über Visual Studio bereitgestellt werden und sind auf *localhost* zugänglich. Bei Fehlern mit *Python* und *VideoPose3D* empfiehlt es sich, die Paketliste zu überprüfen, um zu sehen, ob alle korrekten Versionen installiert wurden (A.7).

6 Evaluation und Validation

Zur Evaluierung und Validation des Projekts wurden einige Usertests durchführt, um Feedback von verschiedenen Personengruppen zu erhalten. Insgesamt wurden 4 Usergroups für die Tests gebildet:

1. **Interne Tests**: Das Entwicklungsteam verwendet mehrere Konten, um in der Testumgebung zu testen. Der Schwerpunkt liegt auf der Leistungsanalyse und dem Auffinden von Fehlern.
2. **Technische Usertests:** Eine Gruppe von technikaffinen Personen testet die Anwendung als Endnutzende.
3. **Nicht-technische Usertests**: Eine Gruppe von Nutzenden, die nicht in der Technologiebranche arbeiten, testen die Anwendung.
4. **Themenexperte / Forscher Usertests**: Eine Gruppe von Anwendern, die über fundierte Kenntnisse im Bereich der *Pose Estimation* verfügen und auch den Output/Service technisch verstehen, den *Poseify* bietet, nutzen die Anwendung.

Jede Usergroup hat ein zugewiesenes Zeitfenster. Um den Datenschutz zu gewährleisten, werden alle Daten nach den Tests gelöscht. Alle Feedbacks wurden via einem *Google Form* gesammelt.

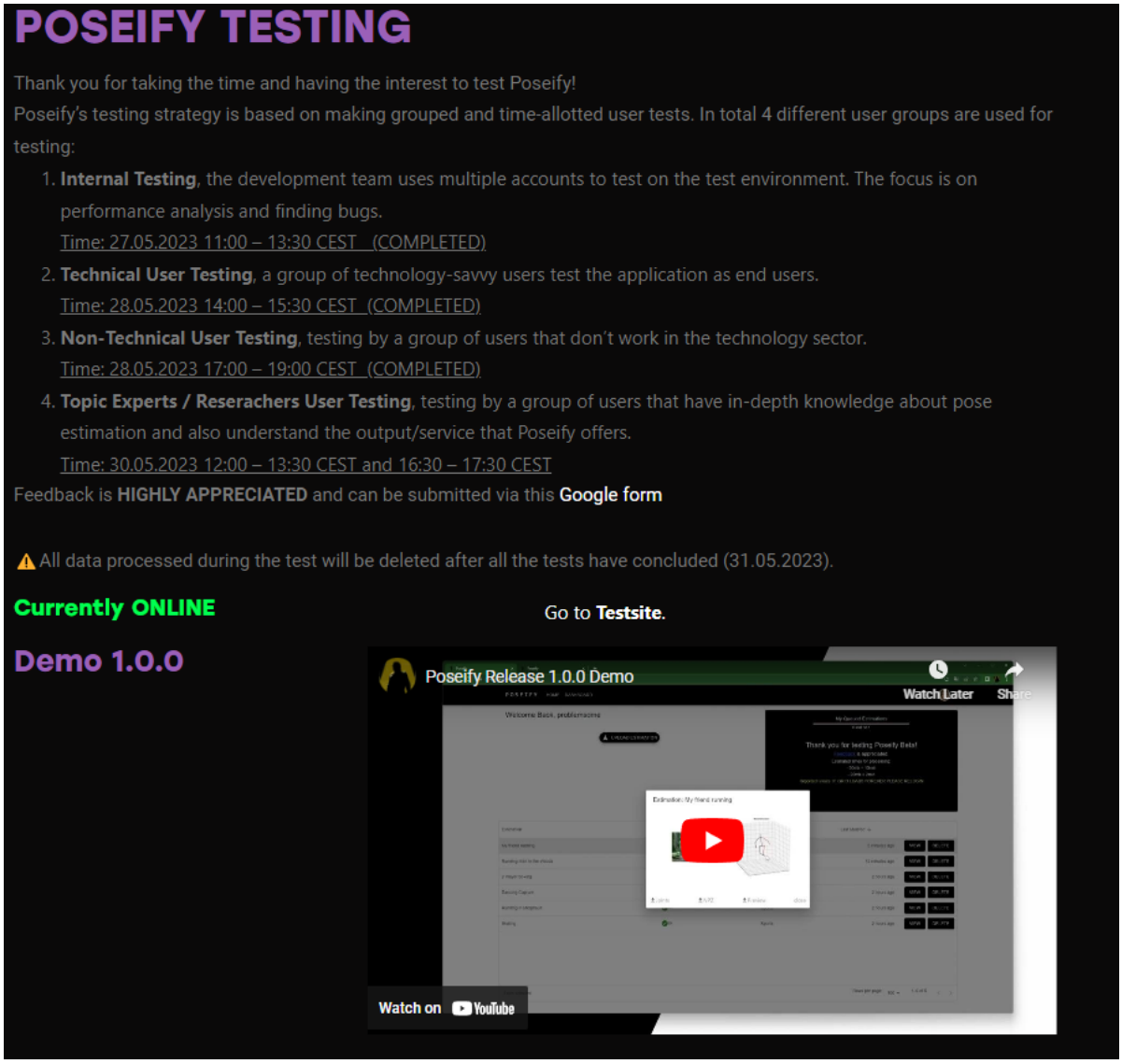


Abbildung 15: Infoseite für Usertesting

6.1 Ziel der Evaluation

Das primäre Ziel der Tests war es, die folgenden Fragen zu beantworten
- Stellt die Anwendung *Estimations* korrekt in die Warteschlange, wenn die Grafikkarte ausgelastet ist (Funktion des Queue Systems)?
- Stürzt die Anwendung nicht ab, wenn mehrere Nutzende *Estimations* anfordern?
- Gibt es irgendwelche Bugs?
- Gibt es fehlende Funktionen?
- Ist das UI user-friendly und verständlich?
- Bringt diese Software einen Mehrwert?

Auf die Ergebnisse der Usertests wird in den unteren Kapiteln und in der Reflexion genauer eingegangen.

6.2 Demo

Eine Demo der Version, die für den Test verwendet wurde, ist auf YouTube:
https://www.youtube.com/watch?v=sN8toxFbWoY

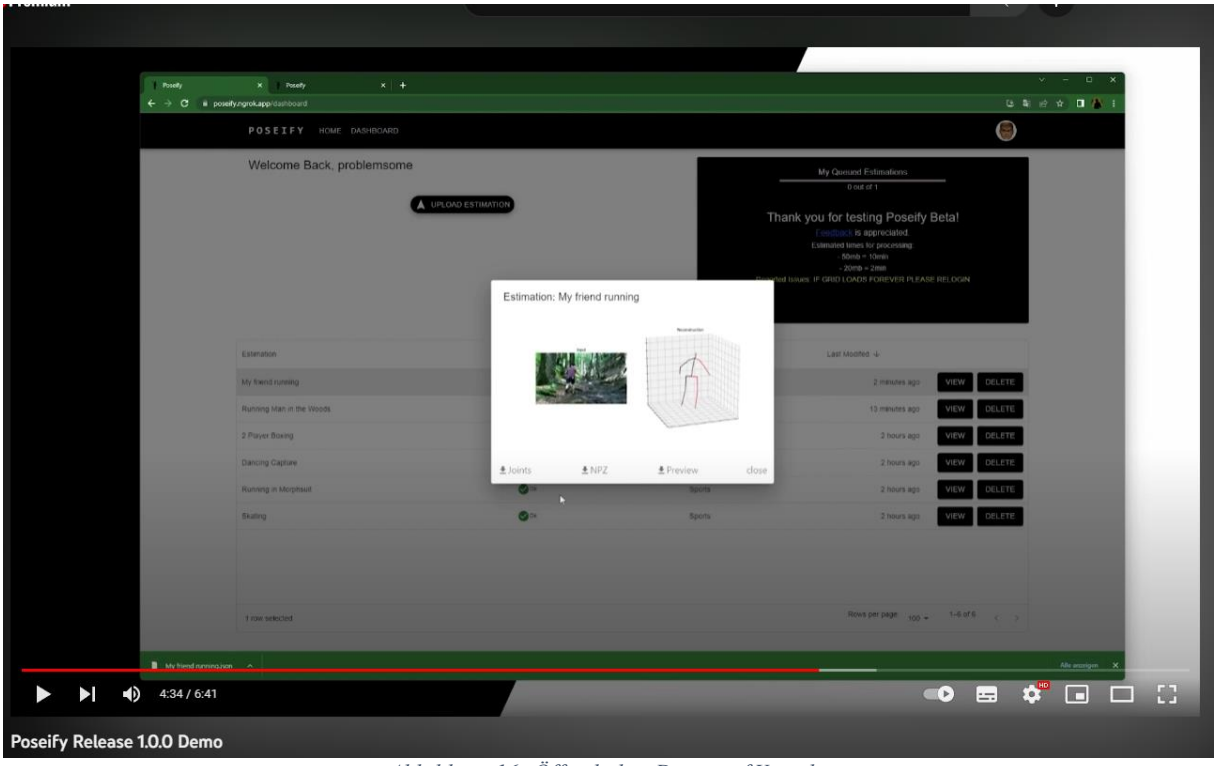


Abbildung 16: Öffentliches Demo auf Youtube

6.3 Zusammenfassung: Internal Tests

Testdatum	Testpersonenanzahl	Fehlgeschlagene Estimations	Erfolgreiche Estimations	Totale Anzahl Estimations
25.05.2023 11:00 – 13:30 CEST	2	2	7	9

Tabelle 3: Eckdaten Internal Tests

Bevor Tests mit externen Personen durchgeführt wurden, testete das Team die Anwendung zuerst selbst. Hier wurde festgestellt, dass eine *Estimation* manchmal endlos im Status «Processing» befinden konnte und, dass eine Videodatei nach einem Fehlschlag nicht aus dem Dateisystem gelöscht wurde. Diese Bugs wurde mit verbesserter Fehlerbehandlung behoben.

6.4 Zusammenfassung: Technische Usertests

Testdatum	Testpersonenanzahl	Fehlgeschlagene Estimations	Erfolgreiche Estimations	Totale Anzahl Estimations
27.05.2023 17:00 – 19:00 CEST	5	2	10	12

Tabelle 4: Eckdaten Technische Usertests

Durch das Feedback-Formular konnten einige Verbesserungsvorschläge am UI gesammelt werden. Folgende Kritiken wurden verbessert:

- Der Login-Button war nicht genug ersichtlich.
- Der Status einer *Estimation* in der Liste konnte nur durch Darüberfahren mit der Maus gelesen werden, bzw. das Icon des Status war nicht genug selbsterklärend.
- Localization war nicht vorhanden, Nutzende aus anderen Zeitzonen als der Server hatten falsche Zeitangaben.
- Aus Fehlermeldungen war nicht ersichtlich, wo ein Fehler passiert ist.
- Der Button, um ein Video hochzuladen war nicht intuitiv verständlich und zu klein.
- Bei einer Testperson konnte eine erfolgreich beendete *Estimation* nicht heruntergeladen werden.

Rückmeldungen, die nicht umgesetzt wurden durch zeitliche oder andere Gründe:

- Eine Angabe wie lange die Verarbeitung noch gehen wird.
- Angabe wie lange gewartet werden muss, bis ein Verarbeitungsslot auf dem Server frei wird.
- Mehre Personen in einem Video und Erkennung für Personen die out-of-frame gehen.
- Eine App, damit direkt aus der Mobile-Galerie ein Video ausgesucht werden kann.
- Drag & Drop für Upload.
- Löschen Button hat keine Bestätigung.

Generell wurde die Verarbeitungsgeschwindigkeit als in Ordnung war genommen, aber schneller wäre besser. Das UI wurde als mehrheitlich verständlich bezeichnet.

6.5 Zusammenfassung: Nicht-technische UserTests

Testdatum	Testpersonenanzahl	Fehlgeschlagene Estimations	Erfolgreiche Estimations	Totale Anzahl Estimations
27.05.2023 17:00 – 19:00 CEST	4	3	14	17

Tabelle 5: Eckdaten nicht-technische Usertests

Von den Nutzenden die nicht in einem technischen Umfeld arbeiten wurden folgende weitere Kritiken zuzüglich zu den bereits erwähnten gesammelt:

- Nach einer Zeit müssen sich gewisse Personen neu einloggen, obwohl die Sitzung nicht abgelaufen sein sollte.
- «Freundlichere» Farben wurden gewünscht.

Weshalb gewisse Nutzende sich manchmal nach einer Zeit neu einloggen mussten, konnte noch nicht festgestellt werden. Durch das *Material UI Theme Feature* kann die Farbkomposition der *ClientApp* schnell und unkompliziert gewechselt werden. Es wurde beschlossen die Farben nicht anzupassen, aber es ist jeder Person, die eine Instanz dieses Projekt hostet, freigestellt das *Theme* nach Belieben zu ändern. Die Nutzenden beschrieben das UI in seiner Anwendung als einfach und selbsterklärend.

6.6 Zusammenfassung: Themenexperte / Forscher Usertests

Testdatum	Testpersonenanzahl	Fehlgeschlagene Estimations	Erfolgreiche Estimations	Totale Anzahl Estimations
30.05.2023 12:00 – 13:30 CEST 16:30 – 17:30 CEST	3	7	17	24

Tabelle 6: Eckdaten Themenexperte Usertests

Während der letzten Tests trat ein unerwartetes Problem auf. Während der ersten Testsitzungen in dieser Testgruppe wurde festgestellt, dass das Inferenzskript von *Detectron2* viele Probleme mit mov-Dateien hatte, was zu einer höheren Anzahl von Fehlern als bei allen anderen Testsitzungen führte. Für die abendliche Sitzung wurde die Dateieingabe auf mp4-Dateien umgestellt. Nach dieser Änderung gab es keinen einzigen Fehler mehr. Das bedeutet, dass die Anwendung bei mp4-Dateien sehr stabil ist, aber bei anderen Videotypen instabil sein kann. Dies müsste verbessert werden, wenn ein breiteres Publikum erreicht werden soll. Dieses Problem könnte auch vermieden werden, wenn eine Aufzeichnungsoption über eine mobile App integriert wäre.

7 Ausblick

Hier wird die geleistete Arbeit kritisch reflektiert und ein Ausblick auf die Möglichkeiten für die Zukunft des Projekts gegeben.

7.1 Reflexion

Das Projekt erwies sich als überaus ambitioniert. Es hat viel Aufwand gekostet in einer relativ kurzen Zeitspanne das Projekt nach bestem Ermessen umzusetzen. Funktionsmässig sind alle Grundfunktionen vorhanden und das Projekt ist als Ganzes lauffähig und einsetzbar. Mit dieser Arbeit wurde eine solide Grundstruktur erbaut auf welcher weiter aufgebaut werden kann, entweder durch das Team selbst, die HSLU oder Interessierte auf der ganzen Welt, welche das Projekt auf GitHub finden. Es gibt noch einige Features, Erweiterungen und Verbesserungen, welche das Team gerne umgesetzt hätte, für welche es jedoch an der Zeit fehlte. Die Planung für die Umsetzung der Hauptfunktionalitäten war zu knapp bemessen. Unvorhergesehene Probleme haben Features ausgebremst und den Rest des Projekts verzögert. Die Probleme lagen unteranderem bei der Installation von Nividia *CUDA*, bei vielen kleinen Verbesserungen und Bugfixes, der Implementation des *Identity Servers* und der Absicherung mit diesem. Es wäre eine gute Idee gewesen von Anfang an das Projekt in Containern aufzubauen, anstelle dies erst am Ende zu machen, so wäre es einfacher gewesen das Projekt zu entwickeln und eine konsistente Umgebung zu haben – egal auf welcher *Maschine*.
Bei den Usertests hat sich das System als stabil herausgestellt und fast alle Anfragen konnten vom System problemlos bearbeitet werden. Mit den Usertests konnten einige kleine, aber wertvolle Verbesserungen implementiert werden. Trotz einiger Bugs und konstruktiven Vorschlägen war das Feedback der Testpersonen im Allgemeinen sehr positiv. Das Team ist im Ganzen sehr zufrieden mit dem Ergebnis und individuell konnten wertvolle Erfahrungen. und Wissen gesammelt werden.

7.2 Ausblick und Möglichkeiten

Es könnten mehrere *Pose Estimation* Algorithmen angeboten werden, wobei eine Person dann auswählen könnte anhand welches Algorithmus eine Eingabe verarbeitet werden soll. Bei einem unzufriedenstellenden Ergebnis bestünde dann die Möglichkeit die *Estimation* mit einem anderen *Algorithmus* zu wiederholen. Die Ausgabe von *VideoPose3D* könnte direkt in eine «.fbx» Datei konvertiert werden, damit diese in z.B. Unity oder andere *3D-Software* importiert werden kann. Auf dem *Core Backend* könnte eine öffentlich zugängliche *REST API* erstellt werden, damit die Dienste des Projekts in andere Software eingebunden werden können. Während der Projektumsetzung ist es leider nicht zu einer Anwendung auf einem leistungsfähigen Server mit mehreren GPUs gekommen. Es fehlt daher an einer Untersuchung, wie gut das Projekt auf mehrere Server verteilt, funktionieren würde. Viel Potential wird bei einer möglichen Mobile App gesehen, welches das Recording und Videoupload vereinfachen würde und eine 3D Preview von der Estimation darstellen könnte. Dank dem BFF-Pattern kann eine solche App effizient in das Software Ecosystem integriert werden. Interessant wäre es auch einen eigenen *Deeplearning Pose Estimation Algorithmus* und/oder einen *Algorithmus*, der detektiert, ob ein Video reale Personen enthält oder von schlechter Qualität ist, zu entwickeln. Mit *Publicity* könnten Nutzende angeworben werden und wenn genügend Interesse vorhanden ist, könnte auch eine aktive *Entwicklungs-Community* entstehen, welche das Projekt weiterentwickelt und verbessert. Was im Moment noch fehlt, ist ein *Userlogin*, welches nicht auf Google beruht und einen Google Account voraussetzt. Von den Tests mit Nutzenden wurden noch weitere Vorschläge gesammelt, welche in Zukunft umgesetzt werden könnten. Eine Zeitangabe wie lange es noch dauert, bis eine *Estimation* fertig ist und eine Zeitangabe wie lange eine Person warten muss bis deren Video aus der Warteschlange in die Verarbeitung kommt wurde gewünscht sowie die Möglichkeit eines Drag & Drop Hochladens und das Aussuchen aus einer Galerie auf dem Mobiltelefon anstelle vom Dateissystem. Ausblickend sieht das Team sehr viel Potenzial für dieses Projekt, obwohl es als Produkt erst in den Anfangsphasen ist. Mit den erwähnten Anpassungen und Erweiterungen könnte Poseify einen wahren Mehrwert generieren und den Zugang zu Pose Estimations vereinfachen.

8 Abkürzungs-, Abbildungs-, Tabellenverzeichnis

8.1 Abkürzungsverzeichnis

API	Application Programming Interface / Programmierschnitstelle. Ermöglicht unabhängige Kommunikation und Datenaustausch zwischen Programmen.
BFF	Backend for Frontend, Software Architektur Pattern oft mit einem Identity Server eingesetzt um eine API vor unerlaubten zugriffen zu schützen
CUDA	Compute Unified Device Architecture
CNN	Convolutional Neural Network
SPA	Single Page Application
UI	Userinterface

8.2 Abbildungsverzeichnis

8.3 Tabellenverzeichnis

9 Literaturverzeichnis

Babu, S. C. (2019). *A 2019 guide to Human Pose Estimation with Deep Learning*. https://nanonets.com/blog/human-pose-estimation-2d-guide/

Bazarevsky, V., Grishchenko, I., Raveendran, K., Zhu, T., Zhang, F., & Grundmann, M. (2020). *BlazePose: On-device Real-time Body Pose tracking* (arXiv:2006.10204). arXiv. http://arxiv.org/abs/2006.10204

Cao, Z., Hidalgo, G., Simon, T., Wei, S.-E., & Sheikh, Y. (2019). *OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields* (arXiv:1812.08008). arXiv. http://arxiv.org/abs/1812.08008

Duende, D. (2023, May 28). BFF Security Framework. *BFF Security Framework*. https://docs.duendesoftware.com/identityserver/v6/bff/

Duende IdentityServer Terminology. (n.d.). Retrieved May 17, 2023, from https://docs.duendesoftware.com/identityserver/v6/overview/terminology/

Duende IdentityServer Terminology. (2023, May 17). https://docs.duendesoftware.com/identityserver/v6/overview/terminology/

Fang, H.-S., Li, J., Tang, H., Xu, C., Zhu, H., Xiu, Y., Li, Y.-L., & Lu, C. (2022). *AlphaPose: Whole-Body Regional Multi-Person Pose Estimation and Tracking in Real-Time* (arXiv:2211.03375). arXiv. http://arxiv.org/abs/2211.03375

Google. (2023). *What's Material?* https://m3.material.io/get-started

Güler, R. A., Neverova, N., & Kokkinos, I. (2018). *DensePose: Dense Human Pose Estimation In The Wild* (arXiv:1802.00434). arXiv. http://arxiv.org/abs/1802.00434

Kendall, A., Grimes, M., & Cipolla, R. (2016). *PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization* (arXiv:1505.07427). arXiv. http://arxiv.org/abs/1505.07427

Kocabas, M., Athanasiou, N., & Black, M. J. (2020). *VIBE: Video Inference for Human Body Pose and Shape Estimation* (arXiv:1912.05656). arXiv. http://arxiv.org/abs/1912.05656

London, L. (n.d.). Virtual Try-On Is More Than A Pandemic Trend And These Brands Are Reaping The Rewards. *Virtual Try-On Is More Than A Pandemic Trend And These Brands Are Reaping The Rewards*. Retrieved May 25, 2023, from https://www.forbes.com/sites/lelalondon/2021/05/20/virtual-try-on-is-more-than-a-pandemic-trendand-these-brands-are-reaping-the-rewards/?sh=4e33bdb26c82

Matos, L. (2023). *Git Commit Best Practices*. https://gist.github.com/luismts/495d982e8c5b1a0ced4a57cf3d93cf60

Pavllo, D., Feichtenhofer, C., Grangier, D., & Auli, M. (2019). *3D human pose estimation in video with temporal convolutions and semi-supervised training* (arXiv:1811.11742). arXiv. http://arxiv.org/abs/1811.11742

Pishchulin, L., Insafutdinov, E., Tang, S., Andres, B., Andriluka, M., Gehler, P., & Schiele, B. (2016). *DeepCut: Joint Subset Partition and Labeling for Multi Person Pose Estimation* (arXiv:1511.06645). arXiv. http://arxiv.org/abs/1511.06645

Pullen, M., & Kirchhofer, C. (2023a). *Poseify Architecture*. https://github.com/fierc3/poseify/wiki/Architecture

Pullen, M., & Kirchhofer, C. (2023b, May 4). *Poseify GitHub Repository*. https://github.com/fierc3/poseify

Starreveld, A. (2022, June 23). What is a BFF? And how to build one? *What Is a BFF? And How to Build One?* https://abstarreveld.medium.com/what-is-a-bff-and-how-to-build-one-e2a2b78cfc43

Sun, K., Xiao, B., Liu, D., & Wang, J. (2019). *Deep High-Resolution Representation Learning for Human Pose Estimation* (arXiv:1902.09212). arXiv. http://arxiv.org/abs/1902.09212

Volet, G. (2022). *Formcheck mit Pose-Estimation*. HSLU. https://portfoliodb.hslu.ch/entries/0e4dc200-ea13-4909-86a0-6e38da8eb65e

Yiannakides, A., Aristidou, A., & Chrysanthou, Y. (2019). Real-time 3D human pose and motion reconstruction from monocular RGB videos. *Computer Animation and Virtual Worlds*, *30*(3–4). https://doi.org/10.1002/cav.1887

Zheng, C., Wu, W., Chen, C., Yang, T., Zhu, S., Shen, J., Kehtarnavaz, N., & Shah, M. (2022). *Deep Learning-Based Human Pose Estimation: A Survey* (arXiv:2012.13392). arXiv. http://arxiv.org/abs/2012.13392

A. Anhänge

A.1. Ausgangslage und Problemstellung

Die Bestimmung der 3D Haltung und Bewegung von Menschen ist eine wichtige Funktionalität in Anwendungen der Computeranimation, Sportanalyse oder Bewegungstherapie. Jedoch sind bisher hierzu kostspielige Motion Capture Systeme notwendig, die die allgemeine Verfügbarkeit stark einschränken. Auf der anderen Seite sind neueste AI Algorithmen in der Lage, die 3D Pose von Menschen aus einfachen Videosequenzen zu schätzen (z.B. VideoPose3D), wobei die Genauigkeit noch nicht jene von professionellen Motion Capture Systemen erreicht. Es besteht somit Forschungsbedarf, um diese Lücke zu schliessen. Zur Unterstützung solcher Forschung soll in diesem Projekt ein Server entwickelt werden, der 3D Pose Estimation in einfacher Weise zur Verfügung stellt.

A.1.1 Ziel der Arbeit und erwartete Resultate

- Entwicklung eines Servers zur 3D Pose Estimation
- Implementierung als Webapplikation
- Dokumentation und kritische Beurteilung

A.1.2 Gewünschte Methoden, Vorgehen

- Einarbeitung und Test von VideoPose3D
- Design der Serverarchitektur als Webapplikation
- Implementierung und Optimierung des Systems
- Ausführliche Tests mit eigenen Videodaten
- Auswertung und Dokumentation

A.1.3 Kreativität, Varianten, Innovation

- Konzeption und Design des Servers
- Experimente mit Cutting Edge Technologie

https://home.jointcreate.com/en_us/ventures/601/

A.2. Libraries, Frameworks und Dependencies

A.2.1 Core

Library/Framework Name	**Version**
.NET (Framework)	**7**
Microsoft.AspNetCore.Authentication.JwtBearer	7.0.5
Microsoft.EntityFrameworkCore.InMemory	7.0.3
RavenDB.Client	5.4.101
Newtonsoft.Json	13.0.2
Swashbuckle.AspNetCore	6.5.0

A.2.2 BFFWeb

Library/Framework Name	**Version**
.NET Core Framework	**7**
Duende.BFF	2.0.0
Duende.BFF.Yarp	2.0.0
Microsoft.AspNetCore.Authentication.OpenIdConnect	7.0.5
Microsoft.AspNetCore.SpaProxy	7.0.3
Microsoft.AspNetCore.SpaServices	3.1.32

A.2.3 ClientApp

Library/Framework Name	Version
React	18.2.0
emotion/react	11.10.6
emotion/styled	11.10.6
fontsource/public-sans	4.5.12
fontsource/roboto	4.5.8
mui/icons-material	5.11.16
mui/joy	5.0.0-alpha.76
mui/material	5.12.1
mui/x-data-grid	6.2.1
testing-library/jest-dom	5.16.5
testing-library/react	13.4.0
testing-library/user-event	13.5.0
types/jest	27.5.2
types/node	16.18.12
types/react	18.0.28
types/react-dom	18.0.11
axios	1.3.4
bootstrap	5.2.0
http-proxy-middleware	2.0.6
jquery	3.6.0
merge	2.1.1
oidc-client	1.11.5
react-dom	18.2.0
react-moment	1.1.3
react-query	3.39.3
react-router-dom	6.8.1
react-scripts	5.0.1
reactstrap	9.1.3
rimraf	3.0.2
web-vitals	2.1.4
workbox-background-sync	6.5.4
workbox-broadcast-update	6.5.4
workbox-cacheable-response	6.5.4
workbox-core	6.5.4
workbox-expiration	6.5.4
workbox-google-analytics	6.5.4
workbox-navigation-preload	6.5.4
workbox-precaching	6.5.4
workbox-range-requests	6.5.4
workbox-routing	6.5.4
workbox-strategies	6.5.4
workbox-streams	6.5.4
ajv	8.11.0
cross-env	7.0.3
eslint	8.22.0
eslint-config-react-app	7.0.1
eslint-plugin-flowtype	8.0.3
eslint-plugin-import	2.26.0
eslint-plugin-jsx-a11y	6.6.1
eslint-plugin-react	7.30.1
eslint-plugin-react-hooks	4.6.0

nan	2.16.0
typescript	4.9.5

A.2.4 Identity Server

Library/Framework Name	**Version**
.NET (Framework)	**7**
Duende.IdentityServer.AspNetIdentity	6.2.0
Microsoft.AspNetCore.Authentication.Google	6.0.0
Microsoft.EntityFrameworkCore.Design	7.0.5
Microsoft.EntityFrameworkCore.SqlServer	7.0.5
Microsoft.NET.Build.Containers	0.4.0
Serilog.AspNetCore	6.0.0
Microsoft.AspNetCore.Diagnostics.EntityFrameworkCore	6.0.0
Microsoft.AspNetCore.Identity.EntityFrameworkCore	6.0.0
Microsoft.AspNetCore.Identity.UI	6.0.0
Microsoft.EntityFrameworkCore.Sqlite	7.0.5
Microsoft.EntityFrameworkCore.Tools	7.0.5

A.3. Arbeitsjournal

SW01-SW03	
Mike	- Projektstruktur, Projekt Setup - Git Repo, Setup - Architektur Diagramm - Testenvironment organisieren - Test Kommunikation Front & Backend - Projektmanagement Setup - Architektur mit Senior Developers besprechen - Definition of Done - ProblemDetails Errorhandling
Corsin	- Evaluation *VideoPose3D* - Installationsscript *VideoPose3D* - *VideoPose3D* Runner Script
Mike & Corsin	- Update *.Net 7* - Architektur - Libraries und Frameworks entscheiden - Dokumentation - Lizenzen Dokumentieren - Datenbank Setup und Test

SW03-SW05	
Mike	- Frontend Navigation - Homepage - Settings Page
Corsin	- *VideoPose3D* Runner Script & Integration
Mike & Corsin	- *BFF* Setup / Authentication - Login / Logout - Datenbank Konzept

SW05-SW011	
Mike	- Dashboard Frontend - Fileuploader - Resultat View

	- Download - Endpoints API - Datenbank Erweiterung
Corsin	- *Estimation* Script Service - Endpoints *API* - Controllers *API/BFF* - *IdentityServer* - Dokumentation
Mike & Corsin	- Controllers *API/BFF* - Endpoints *API* - IdentityServer finalisierung - Testing, Evaluation

SW011-SW013	
Mike	- Upload Limit - Test Deployment - Dockerisation - Debugging / Cleanup
Corsin	- Dokumentation
Mike & Corsin	- Security Trimming *API* Calls - Feedback Formular erstellen

SW013-SW014	
Mike	- Dokumentation - Usertesting Monitoring - Bug fixing - Feedback von Usertests implementieren - Demo aufnehmen / uploaden
Corsin	- Dokumentation
Mike & Corsin	- Dokumentation und verbesserungen - Auswertung Feedback

A.4. Github Repositories

A.4.1 Poseify Backend, BFFWeb, Frontend, Docs

https://github.com/fierc3/poseify

A.4.2 Poseify Identity Server

https://github.com/MadMeister/IdentityServer_Poseify

A.4.3 Detectron 2 für Python 3.7

https://github.com/fierc3/detectron2-python-37

A.4.4 VidePose3D für Poseify

https://github.com/MadMeister/VideoPose3D_poseify

A.5. VideoPose3d Install Helper Script

```
################################################
# VidePose3d and requirements install script   #
################################################

[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12

$git_url                        =                       "https://github.com/git-for-
windows/git/releases/download/v2.39.2.windows.1/Git-2.39.2-64-bit.exe"
$vp3d_url = "https://github.com/MadMeister/VideoPose3D_poseify.git"
$msbuild_url = "https://aka.ms/vs/17/release/vs_buildtools.exe"
$detectron_url = "git+https://github.com/fierc3/detectron2-python-37"
$pretrained_model_url         =         "https://dl.fbaipublicfiles.com/video-pose-
3d/pretrained_h36m_detectron_coco.bin"

$git_dest = "C:\Program Files\Git"
$vp3d_dest = Split-Path -parent $PSCommandPath
$vp3d_dest = $vp3d_dest+"\vp3d"
$msbuild_dest       =       "C:\Program       Files       (x86)\Microsoft       Visual
Studio\2022\BuildTools\MSBuild\Current\Bin"
$pretrained_model_dest                                                          =
"$vp3d_dest\checkpoint\pretrained_h36m_detectron_coco.bin"


"This Script will install the following things:
    - git in $git_dest
    - videopose3d in $vp3d_dest
    - msbuildtools c++ in $msbuild_dest
    - detectron and various python libraries

if you would like to change any paths please edit the paths in this powershell
script"


$git_conf = Read-Host "do you want to install git to $git_dest ? (y/n)"

if ($git_conf -eq 'y') {
    if (-Not(Test-Path $git_dest)) {
        try {
            "installing git"
            Invoke-WebRequest -Uri $git_url -OutFile $git_dest
            $gitExePath = "$git_dest\git.exe"
            Start-Process  $gitExePath  -Wait  -ArgumentList  '/NORESTART  /NOCANCEL
/SP-                    /CLOSEAPPLICATIONS                    /RESTARTAPPLICATIONS
/COMPONENTS="icons,ext\reg\shellhere,assoc,assoc_sh"            /LOG="C:\git-for-
windows.log"'
        }
        catch {
            Write-Host "error while downloading and installing git"
            Write-Host $_
        }
        else {
            "git path already exists, skipping installation"
        }
    }
}

$msbuild_conf = Read-Host "do you want to install msbuildtools to $msbuild_dest ?
(y/n)"

if ($msbuild_conf -eq 'y') {
    if (-Not(Test-Path $msbuild_dest)) {
        try {
```

```
            "installing msbuildtools"
            $msbuildExePath = "$msbuild_dest\vs_buildtools.exe"
            Invoke-WebRequest -Uri $msbuild_url -OutFile  $msbuild_dest

            $buildtoolargs    =    "    --quiet    --norestart    --wait    --add
Microsoft.VisualStudio.Workload.MSBuildTools                                    --add
Microsoft.VisualStudio.Component.VC.Tools.x86.x64                               --add
Microsoft.VisualStudio.Component.VC.CMake.Project                               --add
Microsoft.VisualStudio.Component.VC.ASAN"
            Start-Process  $msbuildExePath  -ArgumentList  $buildtoolargs  -Wait  -
PassThru -WorkingDirectory $msbuild_dest -NoNewWindow
        }
        catch {
            Write-Host "error while downloading and installing msbuildtools"
            Write-Host $_
        }
    }
    else {
        "msbuildtools path already exists, skipping installation"
    }
}

$vp3d_conf = Read-Host "do you want to install videopose3d to $vp3d_dest ? (y/n)"

if ($vp3d_conf -eq 'y') {
    if (-Not(Test-Path $vp3d_dest)) {
        try {
            "installing vp3d"
            git clone $vp3d_url $vp3d_dest
        }
        catch {
            Write-Host "error while downloading and installing vp3d"
            Write-Host $_
        }
    }
    else {
        "vp3d path already exists, skipping installation"
    }
 $CheckPointFolder = "$vp3d_dest\checkpoint"
 if (Test-Path -Path $CheckPointFolder) {
     "Checkpoint folder exists, no reason to create"
 } else {
     "Checkpoint folder is missing, creating it."
     mkdir "$vp3d_dest\checkpoint"
 }

}

$model_conf = Read-Host "do you want to download the pretrained model for vp3d?
(y/n)"

if ($model_conf -eq 'y') {
    try {
        "downloading pretrained vp3d model"
        Invoke-WebRequest        -Uri        $pretrained_model_url        -OutFile
$pretrained_model_dest
    }
    catch {
        Write-Host "error while downloading and installing vp3d model"
        Write-Host $_
    }
}
```

```
$req_conf = Read-Host "do you want to install detectron2 and all python requirements
for videopose3d? as specified in .\requirements.txt ? (y/n)"

if ($req_conf -eq 'y') {
    try {
        "installing requirements"
        python -m pip install $detectron_url
        python -m pip install -r .\requirements.txt
    }
    catch {
        Write-Host "error while downloading and installing detectron2 and or
requirements"
        Write-Host $_
    }
}

$ffmpeg_conf = Read-Host "do you want to install ffmpeg for windows (c:\ffmpeg)?
(y/n)"

if ($ffmpeg_conf -eq 'y') {
    try {
        "installing ffmpeg"
        New-Item -Type Directory -Path C:\ffmpeg ; Set-Location C:\ffmpeg
        curl.exe      -L      'https://www.gyan.dev/ffmpeg/builds/ffmpeg-release-
essentials.zip' -o 'ffmpeg.zip'
        Expand-Archive .\ffmpeg.zip -Force -Verbose
        # Move the executable (*.exe) files to the top folder
        Get-ChildItem -Recurse `
            -Path .\ffmpeg\ -Filter *.exe |
        ForEach-Object {
            Write-Output $_
            Move-Item $_.FullName -Destination C:\ffmpeg -Verbose
        }
        Remove-Item .\ffmpeg.zip
        # List the directory contents
        Get-ChildItem
        # Prepend the FFmpeg folder path to the system path variable
        [System.Environment]::SetEnvironmentVariable(
            "PATH",

"C:\ffmpeg\;$([System.Environment]::GetEnvironmentVariable('PATH','MACHINE'))",
            "Machine"
        )
    }
    catch {
        Write-Host "error while downloading and installing ffmpeg on windows"
        Write-Host $_
    }
}

"IM WELL AND TRULY FINISHED"
Read-Host
```

A.6. Setup.SQL for Identity Server

```
IF OBJECT_ID(N'[__EFMigrationsHistory]') IS NULL
BEGIN
    CREATE TABLE [__EFMigrationsHistory] (
        [MigrationId] nvarchar(150) NOT NULL,
        [ProductVersion] nvarchar(32) NOT NULL,
        CONSTRAINT [PK___EFMigrationsHistory] PRIMARY KEY ([MigrationId])
    );
END;
GO

BEGIN TRANSACTION;
GO

CREATE TABLE [AspNetRoles] (
    [Id] nvarchar(450) NOT NULL,
    [Name] nvarchar(256) NULL,
    [NormalizedName] nvarchar(256) NULL,
    [ConcurrencyStamp] nvarchar(max) NULL,
    CONSTRAINT [PK_AspNetRoles] PRIMARY KEY ([Id])
);
GO

CREATE TABLE [AspNetUsers] (
    [Id] nvarchar(450) NOT NULL,
    [UserName] nvarchar(256) NULL,
    [NormalizedUserName] nvarchar(256) NULL,
    [Email] nvarchar(256) NULL,
    [NormalizedEmail] nvarchar(256) NULL,
    [EmailConfirmed] bit NOT NULL,
    [PasswordHash] nvarchar(max) NULL,
    [SecurityStamp] nvarchar(max) NULL,
    [ConcurrencyStamp] nvarchar(max) NULL,
    [PhoneNumber] nvarchar(max) NULL,
```

```
    [PhoneNumberConfirmed] bit NOT NULL,
    [TwoFactorEnabled] bit NOT NULL,
    [LockoutEnd] datetimeoffset NULL,
    [LockoutEnabled] bit NOT NULL,
    [AccessFailedCount] int NOT NULL,
    CONSTRAINT [PK_AspNetUsers] PRIMARY KEY ([Id])
);
GO

CREATE TABLE [AspNetRoleClaims] (
    [Id] int NOT NULL IDENTITY,
    [RoleId] nvarchar(450) NOT NULL,
    [ClaimType] nvarchar(max) NULL,
    [ClaimValue] nvarchar(max) NULL,
    CONSTRAINT [PK_AspNetRoleClaims] PRIMARY KEY ([Id]),
    CONSTRAINT [FK_AspNetRoleClaims_AspNetRoles_RoleId] FOREIGN KEY ([RoleId]) REFERENCES [AspNetRoles] ([Id]) ON DELETE CASCADE
);
GO

CREATE TABLE [AspNetUserClaims] (
    [Id] int NOT NULL IDENTITY,
    [UserId] nvarchar(450) NOT NULL,
    [ClaimType] nvarchar(max) NULL,
    [ClaimValue] nvarchar(max) NULL,
    CONSTRAINT [PK_AspNetUserClaims] PRIMARY KEY ([Id]),
    CONSTRAINT [FK_AspNetUserClaims_AspNetUsers_UserId] FOREIGN KEY ([UserId]) REFERENCES [AspNetUsers] ([Id]) ON DELETE CASCADE
);
GO

CREATE TABLE [AspNetUserLogins] (
    [LoginProvider] nvarchar(450) NOT NULL,
    [ProviderKey] nvarchar(450) NOT NULL,
    [ProviderDisplayName] nvarchar(max) NULL,
```

```
    [UserId] nvarchar(450) NOT NULL,
    CONSTRAINT [PK_AspNetUserLogins] PRIMARY KEY ([LoginProvider], [ProviderKey]),
    CONSTRAINT [FK_AspNetUserLogins_AspNetUsers_UserId] FOREIGN KEY ([UserId]) REFERENCES [AspNetUsers] ([Id]) ON DELETE CASCADE
);
GO

CREATE TABLE [AspNetUserRoles] (
    [UserId] nvarchar(450) NOT NULL,
    [RoleId] nvarchar(450) NOT NULL,
    CONSTRAINT [PK_AspNetUserRoles] PRIMARY KEY ([UserId], [RoleId]),
    CONSTRAINT [FK_AspNetUserRoles_AspNetRoles_RoleId] FOREIGN KEY ([RoleId]) REFERENCES [AspNetRoles] ([Id]) ON DELETE CASCADE,
    CONSTRAINT [FK_AspNetUserRoles_AspNetUsers_UserId] FOREIGN KEY ([UserId]) REFERENCES [AspNetUsers] ([Id]) ON DELETE CASCADE
);
GO

CREATE TABLE [AspNetUserTokens] (
    [UserId] nvarchar(450) NOT NULL,
    [LoginProvider] nvarchar(450) NOT NULL,
    [Name] nvarchar(450) NOT NULL,
    [Value] nvarchar(max) NULL,
    CONSTRAINT [PK_AspNetUserTokens] PRIMARY KEY ([UserId], [LoginProvider], [Name]),
    CONSTRAINT [FK_AspNetUserTokens_AspNetUsers_UserId] FOREIGN KEY ([UserId]) REFERENCES [AspNetUsers] ([Id]) ON DELETE CASCADE
);
GO

CREATE INDEX [IX_AspNetRoleClaims_RoleId] ON [AspNetRoleClaims] ([RoleId]);
GO

CREATE UNIQUE INDEX [RoleNameIndex] ON [AspNetRoles] ([NormalizedName]) WHERE [NormalizedName] IS NOT NULL;
GO
```

```
CREATE INDEX [IX_AspNetUserClaims_UserId] ON [AspNetUserClaims] ([UserId]);

GO


CREATE INDEX [IX_AspNetUserLogins_UserId] ON [AspNetUserLogins] ([UserId]);

GO


CREATE INDEX [IX_AspNetUserRoles_RoleId] ON [AspNetUserRoles] ([RoleId]);

GO


CREATE INDEX [EmailIndex] ON [AspNetUsers] ([NormalizedEmail]);

GO


CREATE UNIQUE INDEX [UserNameIndex] ON [AspNetUsers] ([NormalizedUserName]) WHERE [NormalizedUserName] IS NOT NULL;

GO


INSERT INTO [__EFMigrationsHistory] ([MigrationId], [ProductVersion])

VALUES (N'20230505084030_InitialCreate', N'7.0.5');

GO


COMMIT;

GO
```

A.7. Conda Package List for CUDA and VideoPose3d

```
# packages in environment at C:\conda38:
#
# Name                    Version                   Build  Channel
absl-py                   1.4.0                     pypi_0    pypi
antlr4-python3-runtime    4.9.3                     pypi_0    pypi
black                     23.3.0                    pypi_0    pypi
blas                      1.0                       mkl
brotlipy                  0.7.0           py38h2bbff1b_1003
ca-certificates           2023.5.7          h56e8100_0    conda-forge
cachetools                5.3.0                     pypi_0    pypi
certifi                   2023.5.7          pyhd8ed1ab_0    conda-forge
cffi                      1.15.1          py38h2bbff1b_3
charset-normalizer        2.0.4             pyhd3eb1b0_0
click                     8.1.3                     pypi_0    pypi
cloudpickle               2.2.1                     pypi_0    pypi
colorama                  0.4.6             pyhd8ed1ab_0    conda-forge
conda                     4.14.0          py38haa244fe_0    conda-forge
conda-package-handling    2.0.2             pyh38be061_0    conda-forge
conda-package-streaming   0.7.0             pyhd8ed1ab_1    conda-forge
conda-tree                1.1.0             pyhd8ed1ab_2    conda-forge
contourpy                 1.0.7                     pypi_0    pypi
cryptography              39.0.1          py38h21b164f_0
cuda-cccl                 12.1.109                  0    nvidia
cuda-cudart               11.7.99                   0    nvidia
cuda-cudart-dev           11.7.99                   0    nvidia
cuda-cupti                11.7.101                  0    nvidia
cuda-libraries            11.7.1                    0    nvidia
cuda-libraries-dev        11.7.1                    0    nvidia
cuda-nvrtc                11.7.99                   0    nvidia
cuda-nvrtc-dev            11.7.99                   0    nvidia
cuda-nvtx                 11.7.91                   0    nvidia
cuda-runtime              11.7.1                    0    nvidia
cycler                    0.11.0                    pypi_0    pypi
detectron2                0.6                       dev_0    <develop>
ffmpeg                    1.4                       pypi_0    pypi
ffmpeg-python             0.2.0                     py_0    conda-forge
filelock                  3.9.0           py38haa95532_0
fonttools                 4.39.4                    pypi_0    pypi
freetype                  2.12.1            ha860e81_0
future                    0.18.3          py38haa95532_0
fvcore                    0.1.5.post20221221        pypi_0    pypi
giflib                    5.2.1             h8cc25b3_3
google-auth               2.18.1                    pypi_0    pypi
google-auth-oauthlib      1.0.0                     pypi_0    pypi
grpcio                    1.54.2                    pypi_0    pypi
hydra-core                1.3.2                     pypi_0    pypi
idna                      3.4             py38haa95532_0
importlib-metadata        6.6.0                     pypi_0    pypi
importlib-resources       5.12.0                    pypi_0    pypi
intel-openmp              2023.1.0        h59b6b97_46319
iopath                    0.1.9                     pypi_0    pypi
jinja2                    3.1.2           py38haa95532_0
jpeg                      9e                h2bbff1b_1
kiwisolver                1.4.4                     pypi_0    pypi
```

lerc 3.0 hd77b12b_0
libblas 3.9.0 1_h8933c1f_netlib conda-forge
libcblas 3.9.0 5_hd5c7e75_netlib conda-forge
libcublas 11.10.3.66 0 nvidia
libcublas-dev 11.10.3.66 0 nvidia
libcufft 10.7.2.124 0 nvidia
libcufft-dev 10.7.2.124 0 nvidia
libcurand 10.3.2.106 0 nvidia
libcurand-dev 10.3.2.106 0 nvidia
libcusolver 11.4.0.1 0 nvidia
libcusolver-dev 11.4.0.1 0 nvidia
libcusparse 11.7.4.91 0 nvidia
libcusparse-dev 11.7.4.91 0 nvidia
libdeflate 1.17 h2bbff1b_0
liblapack 3.9.0 5_hd5c7e75_netlib conda-forge
libnpp 11.7.4.75 0 nvidia
libnpp-dev 11.7.4.75 0 nvidia
libnvjpeg 11.8.0.2 0 nvidia
libnvjpeg-dev 11.8.0.2 0 nvidia
libpng 1.6.39 h8cc25b3_0
libtiff 4.5.0 h6c2663c_2
libuv 1.44.2 h2bbff1b_0
libwebp 1.2.4 hbc33d0d_1
libwebp-base 1.2.4 h2bbff1b_1
lz4-c 1.9.4 h2bbff1b_0
m2w64-gcc-libgfortran 5.3.0 6 conda-forge
m2w64-gcc-libs 5.3.0 7 conda-forge
m2w64-gcc-libs-core 5.3.0 7 conda-forge
m2w64-gmp 6.1.0 2 conda-forge
m2w64-libwinpthread-git 5.0.0.4634.697f757 2 conda-forge
markdown 3.4.3 pypi_0 pypi
markupsafe 2.1.1 py38h2bbff1b_0
matplotlib 3.7.1 pypi_0 pypi
menuinst 1.4.19 py38haa244fe_1 conda-forge
mkl 2023.1.0 h8bd8f75_46356
mkl-service 2.4.0 py38h2bbff1b_1
mkl_fft 1.3.6 py38hf11a4ad_1
mkl_random 1.2.2 py38hf11a4ad_1
mpmath 1.2.1 py38haa95532_0
msys2-conda-epoch 20160418 1 conda-forge
mypy-extensions 1.0.0 pypi_0 pypi
networkx 2.8.4 py38haa95532_1
numpy 1.21.6 py38h5ed9b9d_0 conda-forge
oauthlib 3.2.2 pypi_0 pypi
omegaconf 2.3.0 pypi_0 pypi
openssl 1.1.1l h8ffe710_0 conda-forge
packaging 23.1 pypi_0 pypi
pathspec 0.11.1 pypi_0 pypi
pillow 9.4.0 py38hd77b12b_0
pip 23.0.1 py38haa95532_0
platformdirs 3.5.1 pypi_0 pypi
portalocker 2.7.0 pypi_0 pypi
protobuf 4.23.1 pypi_0 pypi
pyasn1 0.5.0 pypi_0 pypi
pyasn1-modules 0.3.0 pypi_0 pypi
pycocotools 2.0.6 pypi_0 pypi

pycosat	0.6.4	py38h2bbff1b_0	
pycparser	2.21	pyhd3eb1b0_0	
pyee	9.1.0	pypi_0	pypi
pyopenssl	23.0.0	py38haa95532_0	
pyparsing	3.0.9	pypi_0	pypi
pysocks	1.7.1	py38haa95532_0	
python	3.8.0	hff0d562_2	
python-dateutil	2.8.2	pypi_0	pypi
python-ffmpeg	2.0.4	pypi_0	pypi
python_abi	3.8	2_cp38	conda-forge
pytorch	2.0.1	py3.8_cuda11.7_cudnn8_0	pytorch
pytorch-cuda	11.7	h16d0643_5	pytorch
pytorch-mutex	1.0	cuda	pytorch
pywin32	306	pypi_0	pypi
pyyaml	6.0	pypi_0	pypi
requests	2.29.0	py38haa95532_0	
requests-oauthlib	1.3.1	pypi_0	pypi
rsa	4.9	pypi_0	pypi
ruamel_yaml	0.15.80	py38h294d835_1007	conda-forge
setuptools	66.0.0	py38haa95532_0	
six	1.16.0	pypi_0	pypi
sqlite	3.41.2	h2bbff1b_0	
sympy	1.11.1	py38haa95532_0	
tabulate	0.9.0	pypi_0	pypi
tbb	2021.8.0	h59b6b97_0	
tensorboard	2.13.0	pypi_0	pypi
tensorboard-data-server	0.7.0	pypi_0	pypi
termcolor	2.3.0	pypi_0	pypi
tk	8.6.12	h2bbff1b_0	
tomli	2.0.1	pypi_0	pypi
toolz	0.12.0	pyhd8ed1ab_0	conda-forge
torchaudio	2.0.2	pypi_0	pypi
torchvision	0.15.2	pypi_0	pypi
tqdm	4.65.0	pypi_0	pypi
typing_extensions	4.5.0	py38haa95532_0	
urllib3	1.26.15	py38haa95532_0	
vc	14.2	h21ff451_1	
vs2015_runtime	14.27.29016	h5e58377_2	
werkzeug	2.3.4	pypi_0	pypi
wheel	0.38.4	py38haa95532_0	
win_inet_pton	1.1.0	py38haa95532_0	
xz	5.4.2	h8cc25b3_0	
yacs	0.1.8	pypi_0	pypi
yaml	0.2.5	h8ffe710_2	conda-forge
zipp	3.15.0	pypi_0	pypi
zlib	1.2.13	h8cc25b3_0	
zstandard	0.19.0	py38h2bbff1b_0	
zstd	1.5.5	hd43e919_0	