Todo list

Bei Abgabe: Anweisung nocite in Bachelorarbeit.tex entfernen

Bei Abgabe: In Bachelorarbeit.tex und Expose.tex Dokumentenoption overfullrule entfernen und die Option final eintragen

Bei Abgabe: In Packages.tex beim Package todonotes die Option disable eintragen, um Todos zu deaktivieren

Bei Abgabe: In Packages.tex beim TODO die größte Seitennummer eintragen

Wenn eine Version der Arbeit erstellt wird, die gedruckt werden soll in Packages.tex beim Package hyperref die Option urlcolor=blue entfernen

TECHNIK
HOCHSCHULE MAINZ
UNIVERSITY OF
APPLIED SCIENCES

Masterarbeit

Studiengang Geoinformatik M.Eng.

A standards-based, task-specifc telepresence application for contemporary dance

Hochschule Mainz

University of Applied Sciences

Fachbereich Technik

Vorgelegt von:	Vorname Nachname
	Straße
	PLZ Ort
	Matrikel-Nr. 123456
Vorgelegt bei:	Prof. Dr. Betreuer
Eingereicht am:	01.01.1970

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Masterarbeit

> **A standards-based, task-specifc telepresence application for contemporary dance**

selbstständig und ohne fremde Hilfe angefertigt habe. Ich habe dabei nur die in der Arbeit angegebenen Quellen und Hilfsmittel benutzt.

Zudem versichere ich, dass ich weder diese noch inhaltlich verwandte Arbeiten als Prüfungsleistung in anderen Fächern eingereicht habe oder einreichen werde.

Mainz, den 01.01.1970 Vorname Nachname

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit.

Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Contents

List of Figures

List of Tables

List of abbreviations

3D 3-dimensional

AJAX Asynchronous JavaScript and XML

API Application Programming Interface

CNCF Cloud Native Computing Foundation

CSS Cascading Style Sheets

ES6 ECMAScript 6

HTML HyperText Markup Language

HTTP Hypertext Transmission Protocol

IETF Internet Engineering Task Force

JSX JavaScript XML

JS JavaScript

NPM Node Package Manager

OCI Open Container Initiative

PWA Progressive Web Application

RFC Request For Comments

RTC Real-time Communication

SPA Single Page Application

TS TypeScript

UI User Interface

W3C World Wide Web Consortium

WebRTC Web Real-Time Communication

XR Mixed Reality

1. Introduction

1.1. Background

In recent years, remote collaboration has become increasingly important due to the rise of broader digitalisation strategies, especially since the pandemic. As a result, teleconferencing and telepresence platforms have become more pervasive in many work environments. These platforms allow people to work together remotely in real-time, usually focusing on streaming video and audio, document sharing or collaborative whiteboarding. While this is fine for most use cases and desktop-based workplaces, it lacks certain immersive qualities required for practices such as contemporary dance, where people relate to a shared space and physical presence. This became apparent in March 2020, when dancers could no longer rehearse and work together due to the lockdown. Despite this, there were attempts at using Zoom to stream and record collaborative rehearsals or dance classes. Still, these were confined to a screen-centric interface and limited to audio and video.

While commercial conferencing tools such as *Zoom*, *Google Meet* and *Microsoft Teams* dominate in popularity among conferencing applications (Brandl, 2023), there are both free and open-source variants such as *JitsiMeet* and *BigBlueButton*. However, these all focus on the most basic form of screen-based conferencing mentioned above. While there are various domain-specific solutions for specialised applications, mainly in telemedicine (as well as general industry and the military), that support more immersive and task-

specific remote collaboration, these are either unaffordable or unavailable to the general public.

As support for web standards is driven by key industry players (Davis et al., 2023), and with it the availability of a wide range of basic functionality, as well as access to display and sensor technology for deploying applications on desktop and mobile devices, there is an increased potential for smaller and more specific applications to be built and deployed with relative ease. This opens up new possibilities for niche cases of remote collaboration, such as dance, where collaborative functionality needs to be extended from the traditional paradigm of remotely viewing video streams to the creation of shared virtual environments that facilitate a sense of 'being there' together, if only at an abstract level (Skarbez et al., 2017).

The standard for Real-time Communication (RTC) in Browsers or Web Real-Time Communication (WebRTC) ('WebRTC: Real-Time Communication in Browsers', 2023) was first proposed by *Google* in 2011 and became an official World Wide Web Consortium (W3C) standard in 2021 (Couriol, 2021). It is already being used in a wide range of applications, such as some of the conferencing tools mentioned above, media streaming servers such as Wowza or Ant, or real-time frameworks and servers such as *Mediasoup*, *Janus* or *LiveKit*. In its most basic form, WebRTC establishes peer-to-peer connections between different devices, allowing low-latency exchange of media streams and arbitrary messages on so-called data channels.

1.2. Proposal

I propose a feasibility study for a reference implementation of a domain-specific telepresence platform based entirely on web standards and open-source components. The platform allows streaming of sensor data in addition to audio and video. This means that remote collaborators can share a wide range of data from multiple sources, such as motion capture, wearables and more. As the data flows through the WebRTC data

channels alongside the usual video and audio streams, it can be analysed and rendered as required for a specific task on the client device.

A reference implementation will be created to demonstrate the platform's capabilities to simulate essential aspects of presence in a shared virtual environment (Skarbez et al., 2017). The software implementation, written entirely in JavaScript (JS), should rely on existing open-source libraries and frameworks as much as possible and add as little custom code as necessary. Only the minimal viable product will be built to observe its basic functionality, leaving issues of robustness, interface design, security and scalability outside the scope of this study.

For dancers to communicate while moving in remote rehearsal spaces, a video feed is less important as they need to be free from looking at a screen or typing on a keyboard. A virtual or augmented reality headset is also ineffective because it hinders vision and movement in the physical space. Here, spatial sonification seems to be the optimal alternative to create a virtual environment without interfering with the dancers' ability to move freely. The sound generation code should be kept as minimal as possible.

For practical implementation testing, two remote rehearsal spaces of similar size will need to be set up with a motion capture system (TDB) that generates spatial data from the dancers' movements. Both rooms will be within the university network to provide a known and controllable quality of service. The two dancers will use the system in each location. The interaction will be via wireless in-ear sports headsets, commonly used for exercise and providing basic verbal communication via WebRTC's audio channels.

To interact with the spatial dimension of a shared virtual environment consisting of both rehearsal rooms, sonification is used as a means of orientation in the virtual space. The *Web Audio Application Programming Interface (API)* provides functionalities for generating, mixing and positioning sound sources in spatial audio ('Web Audio API', 2021), which will be used to render each participant's microphone sound alongside their

distinct signature sound (one percussive and the other sustained) at their tracked position in space, modulated by two basic movement qualities, the contraction index and the quantity of movement, based on an existing proposal for qualitative movement analysis (Volpe, 2003) and derived from the motion capture data. Each participant can only hear the other to avoid confusion. The spatial audio representation should provide a basic sense of position, while the sound characteristic should indicate the type of movement being performed. In addition, verbal cues can be exchanged.

A separate client implementation uses the same data to visualise the dancers' motion capture data as basic 3-dimensional (3D) images. It adds the combined sonification, which can then be viewed by a third party using the Mixed Reality (XR) Device API for the web ('WebXR Device API', 2023), allowing viewers to evaluate aesthetic aspects of the combined interaction result. In addition, a recording of the session's streams can later be replayed and reviewed by the dancers using this client implementation.

2. Concepts

2.1. Telepresence

The term *Telepresence* first appears in an article by Marvin Minsky in which the author roughly defines it as a form of remote robotic operation, that 'emphasizes the importance of high-quality sensory feedback' and that its development's biggest challenge is 'achieving that sense of "being there."' (Minsky, 1980). Minsky was arguing from a standpoint that was concerned mostly with robotic *manipulators* that performed labour that was either mediated over a distance or enhanced both the abilities and the safety of the operator.

The current spectrum of telepresence is much more diverse. While there are applications of remote robotic control in industry, telemedicine and the military, the most common instance has become the teleconferencing application relaying video and audio streams, as well as allowing chat and collaborative whiteboards.

In this study, the term telepresence is used to explicitly describe a form of virtual or augmented reality, that allows multiple people to experience a form of presence and immersion.

2.2. Motion Capture

The positional tracking of specific keypoints on a moving body over time is referred to as motion capture.

Motion capture technology is often used in CGI, enabling puppeteering 3D avatars for motion picture productions, character animation in games. For these purposes, a high accuracy is required and the technological and financial entry-barriers are relatively high. These applications make use of systems by *Vicon* or *OptiTrack*, which use visual markers to track movement in space and require a studio environment to be deployed. Another markerless optical system is Captury Live, which uses a 360° camera setup to track humanoid moving actors.

In the performance field, the preferred method are IMU-based tracking systems like the *SmartSuit* by Rokoko or the Perception Neuron sensor kit. These operate over radio and are independent from the lighting conditions, but tend to produce less accurate data or are subject to interference.

The grassroots setup for motion capture is the Kinect, developed by Microsoft in 2010, featuring an infrared time-of-flight measurement system, that produces a depth image from which a 3D pose can then be extracted using '3D pose estimation' (=poseEstimationPaper)
The Kinect was frequently used among creative coders, although being originally developed for games. In 2024, the Kinect, now called Azure Kinect, is supposed to be officially discontinued. However, there are other low-cost 3D-cameras on the market like the Oak-D with an integrated processing engine or the Orbbec Femto Bolt. These systems produce rather sub-par accuracy, but can be used to analyse more general dynamics in the movement data.

Deep learning models for motion capture like PoseNet or BlazePose have also become available and, while mostly used on 2D (surveillance) footage, can be extended into 3D if combined with the right calibration data (e.g. depth images). These models are fast

and can be run on a normal webcam, but also tend to produce rather coarse movement data.

2.3. Movement Data Sonification

The sonification of movement data is used in health and therapeutic research to offer an acoustic interface to experience dynamics in movement properties. This can be used for rehabilitation and stabilising movement practice or as a guiding signal within an excercise.

The basic principle for movement sonification is the same as for any data sonification. It requires certain data points to be tied to acoustic properties. This can be a direct value connection from one property to the other (e.g. velocity to loudness, altitude to pitch), but can also be achieved using indirect logical constraints (e.g. if multiple thresholds are crossed, a single signal is triggered).

2.4. Application Design Paradigms

2.4.1. Single Page Applications

The idea of a Single Page Application (SPA) originated around the beginning of the 2000s with the concepts 'Inner-Browsing' (Galli et al., 2003) and Asynchronous JavaScript and XML (AJAX) (Garrett, 2005). It breaks with the traditional way of moving from one page to another in favour of asynchronous loading and replacing parts of the current page. This allows for a website to evoke the look and feel of a regular desktop application.

2.4.2. Progressive Web Applications

The term Progressive Web Application (PWA) was initially coined in 2015 by two *Google* employees in an online Article (Russell & Berriman, 2015). At its core, it describes the process of a website "progressively" evolving into a true device application by adding offline functionality and blending with the operating system functionality. It is often built atop the concept of an SPA and can be perceived by the user as an application they own instead of just accessed at a remote location.

2.4.3. Real-time Web Applications

A real-time web application enhances the user experience by relaying relevant changes on the server to the server as they happen. This can be a simple chat application or a more complex collaborative multi-user environment. While real-time updates can happen on any multi-page website, they can also be a beneficial feature of an SPA or a PWA. Instantaneous updates are commonly realised using WebSockets, a transmission protocol that was standardised as Request For Comments (RFC) 6455 by the Internet Engineering Task Force (IETF) in 2011 (Fette & Melnikov, 2011). It allows full-duplex communication between client and server, running on the same ports and in the same transport layer as the half-duplex Hypertext Transmission Protocol (HTTP) protocol, thus being compatible with existing web infrastructure. It allows for updates to be pushed to the client whenever a resource on the server changes.

2.5. Application Deployment

2.5.1. Containerisation

Containerisation, in the context of computing infrastructure, refers to the 'packaging of software code with just the operating system (OS) libraries and dependencies required to run the code to create a single lightweight executable—called a container—that runs consistently on any infrastructure.' (IBM, 2024b) It was popularised through the release of the *Docker Engine*, an open-source project devoted to creating an industry standard for application containerisation (Barbier, 2014). The *Docker* team eventually launched the Open Container Initiative (OCI) in 2015, which serves as 'a lightweight, open governance structure (project), formed under the auspices of the Linux Foundation, for the express purpose of creating open industry standards around container formats and runtimes.' It subsequently received *Docker*'s container runtime and format as a donation, which was released as *runC* version 1.0 in 2020 (Linux Foundation, 2024). In recent years it has become the defacto standard for packaging and delivering applications in the web development field and beyond. *GitHub* reports that 'in 2023, 4.3 million public and private repositories used Dockerfiles — and more than 1 million public repositories used Dockerfiles for creating containers.' (Daigle and GitHub, 2023)

2.5.2. Container Orchestration

'Container orchestration automates the provisioning, deployment, networking, scaling, availability, and lifecycle management of containers.' (IBM, 2024a) The concept first gained popularity as *Docker Swarm*, which is a functionality of the *Docker* software, but its most successful instance so far is as the software package *Kubernetes*, which originated at *Google* in late 2013 (Burns, 2018) and went on to be included in the Cloud Native Computing Foundation (CNCF), a project by the *Linux Foundation*, that 'aims to advance the state-of-the-art for building cloud native applications and services' (Linux

Foundation, 2015). It can be extended, highly customised and deployed on anything from an embedded device to a large-scale cloud infrastructure, providing a versatile deployment and management tool for many kinds of application infrastructures.

3. Tools

3.1. Web Development Languages

Table 3.1.: Ranking among the most used languages on GitHub

Language	Rank
JavaScript	1
Python	2
TypeScript	3
C++	6

3.1.1. JavaScript

A scripting language created by Brendan Eich in 1995 as part of the release of the *Netscape 2* browser (Netscape, 1995), then was officially standardised by the swiss standards body *Ecma* in 1997 as *ECMA-262* or *ECMAScript*, as it is known today. This standard later was the basis for *JScript* by *Microsoft* and *ActionScript* as part of *Macromedia Flash*. The version currently being supported by all browsers (except Internet Explorer 11) is ECMAScript 6 (ES6) (W3Schools, 2024).

JS is an object-oriented, weakly-typed programming language that allows multiple programming paradigms to be applied. It is primarily used in the browser to add extra functionality to web pages. The underlying *ECMAScript* standard does not define any

input or output methods, which means that this is provided by the specific environment it is being used in (e.g. desktop or mobile browsers).

3.1.2. TypeScript

TypeScript (TS) was released by Microsoft in 2012 'to accommodate an increasing number of developers who are interested in using JavaScript to build large-scale Web applications to run in a browser, rather than on the desktop.' (Jackson, 2012) It is compliant with the underlying Ecma scripting standard and is designed as a superset of JS, adding static typing. It uses a compiler to generate regular JS code.

3.2. Native Application Development

3.2.1. Node JS

Originally released by developer Ryan Dahl in 2009 a server-side JS environment, Node.js runs standard ECMAScript in Google's V8 engine, allowing multithreading and native code integration.

Node.js uses the Node Package Manager (NPM) to package code as modules, which can be used as dependencies. These modules can also integrate native C++ code, enabling bindings to most existing open-source libraries in the linux ecosystem.

It can be used to develop APIs or other server-side applications, but is also used to support local web development processes like preprocessing or packaging and deployment.

3.2.2. Python

The Python programming language was created by Guido van Rossum in 1990 (Python Software Foundation, 2024). It is a multi-paradigm language that is both dynamically and strongly typed (van Rossum, 2008). It relies heavily on indentation and whitespace to structure the code.

The language uses a standard library and the surrounding ecosystem of available modules and applications based on Python makes it a good choice for data processing and science.

There is a native code interface that allows extending python with bindings to native code like with Node.js.

3.2.3. C/C++

C++ originated as an extension to C in 1985. It is a multi-paradigm, statically typed and object oriented programming language.

It can be used to develop code for embedded platforms like Arduino, is used to extend both Node.js and Python and more generally provides access to direct interaction with the operating system and its APIs.

3.3. Frontend Frameworks and Libraries

There is a wide range of available JS frameworks to build dynamic frontends for SPAs and PWAs. The three libraries currently dominating the landscape are *React*, developed by *Facebook* in 2013, and *Vue.js*, created by Evan You in 2014. These libraries can be used with frameworks to offer complete routing and state management solutions. Another

popular framework is *Angular*, initially released by *Google* in 2010 and re-released in 2016.

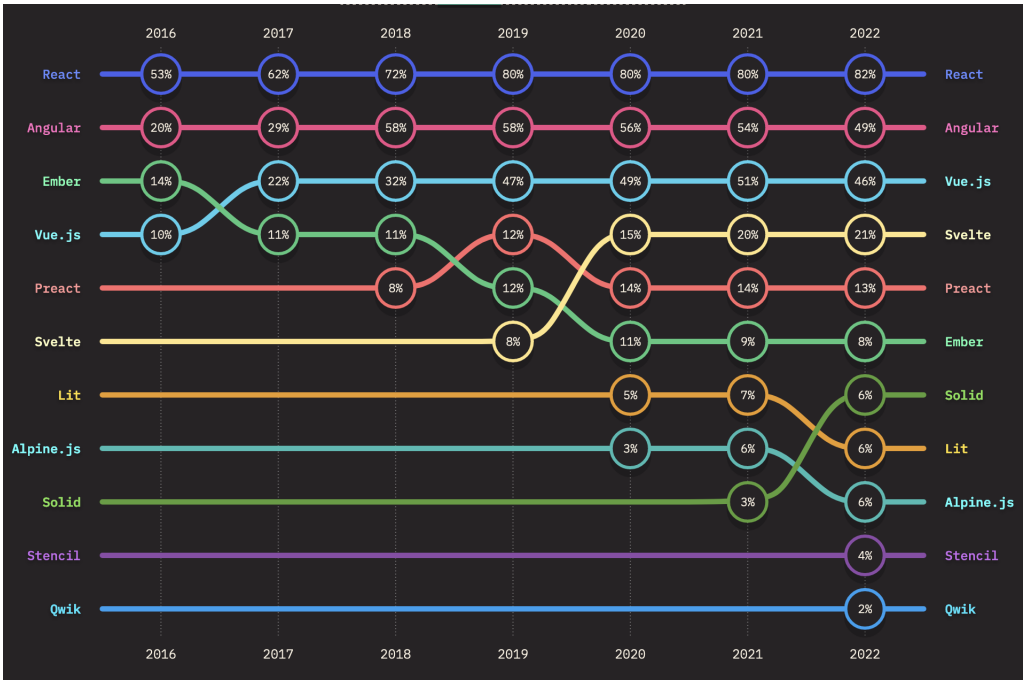

Figure 3.1.: State of JS: Most used frontend frameworks in 2022[1]

3.3.1. React

React (https://react.dev/), developed by *Facebook* and maintained by its successor *Meta*, has become the most widely used tool for building SPAs and is steadily leading the rankings for most used frontend frameworks both in the *StackOverflow* (Overflow, 2023) and the *State Of JS* (Greif & Burel, 2023b) polls. By definition, it is not a framework but a User Interface (UI) library that builds on other extensions to support state management, routing and deployment functionality. Although it is not a framework itself, there are existing frameworks like *Next.js* (https://nextjs.org/) for the web and *ReactNative* (https://reactnative.dev/) for building mobile apps using native functionality. React makes use of JavaScript XML (JSX), which allows directly mixing inline HyperText Markup Language (HTML) with the JS or TS code structure.

[1] Greif and Burel, 2023b

3.3.2. Vue.js

Vue.js (https://vuejs.org/) was developed by Evan You and is maintained by an international team of individuals. It had a relatively marginal presence in the first years after its inception. This can be partially attributed to its origin in China, and most of its supporting modules were localised in Chinese. Over the years, it grew in popularity and received much more international support, eventually overcoming the language barrier. Unlike *React*, it is billed as a "progressive framework" that provides fundamental functionality for building reactive components but also accommodates more complex use-cases (You, 2021). *Vue.js* builds on standard JS or TS, HTML and Cascading Style Sheets (CSS) to build components, recommending a simple template mechanism mixed with reactive substitutions. However, it also supports using JSX for specifying inline HTML within JS. As with *React*, there are extensions and frameworks like *Quasar* (https://quasar.dev/) and *Nuxt* (https://nuxt.com/) that enable even more sophisticated workflows for application development and deployment.

3.3.3. Angular

Angular was initially released by *Google* in 2010 as *AngularJS* and officially discontinued in 2022 (https://angularjs.org/). A completely overhauled and currently used version 2 was released in 2016 and maintained by *Google*. It is different from *React* and *Vue.js* in that it is a complete framework that contains everything required to build and deploy an application, and it explicitly recommends TS as a programming language. The framework is also less flexible in that it is opinionated and has its own set of best practices baked into the framework's structure.

3.4. Backend Libraries

3.4.1. Express

3.4.2. Koa

3.4.3. Fastify

3.5. Backend Frameworks

3.5.1. Nest JS

3.5.2. Meteor

3.5.3. Feathers

4. Methodology

4.1. Quantitative Analysis

4.1.1. Performance Testing

4.1.2. Time spent

4.2. Qualitative Analysis

4.2.1. Code Quality

4.2.2. Critical reflection

5. Implementation

5.1. Architecture

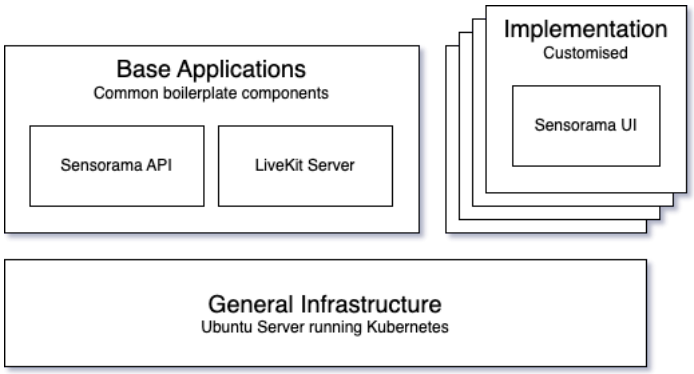

Figure 5.1.: The main components comprising the application architecture

5.1.1. Hardware

5.1.2. Software

5.2. Application infrastructure

While all the frameworks represented in 3.1 could be used to build an application as envisioned in this study, Vue is selected as the tool of choice, both due to the relatively high acceptance and the comparably steep learning curve. Additionally, it is already used in a number of applications designed by the author.

5.2.1. WebRTC Server

5.2.2. Database

5.3. Design Paradigms

5.3.1. Application Partitioning

5.3.2. Coding Style

5.3.3. Testing

5.4. Application Components

5.4.1. Web Frontend

5.4.2. API Backend

5.4.3. Native Utilities

6. Conclusion

6.1. Evaluation Results

6.2. Critical reflection

6.3. Outlook

Bibliography

Barbier, J. (2014, June 9). *It's here: Docker 1.0*. Retrieved January 15, 2024, from https://web.archive.org/web/20220518024454/https://www.docker.com/blog/its-here-docker-1-0/

Brandl, R. (2023, January 4). *The Most Popular Video Call Conferencing Platforms Worldwide*. Retrieved April 3, 2023, from https://www.emailtooltester.com/en/blog/video-conferencing-market-share/

Burns, B. (2018, July 20). *The history of kubernetes & the community behind it*. Retrieved January 15, 2024, from https://kubernetes.io/blog/2018/07/20/the-history-of-kubernetes-the-community-behind-it/

Couriol, B. (2021, April 7). *10 Years after Inception, WebRTC Becomes an Official Web Standard*. Retrieved April 3, 2023, from https://www.infoq.com/news/2021/04/webrtc-official-web-standard/

Daigle, K., & GitHub. (2023, November 8). *Octoverse: The state of open source and rise of ai in 2023*. Retrieved January 14, 2024, from https://github.blog/2023-11-08-the-state-of-open-source-and-ai/

Davis, J., Nguyen, T., & Simmons, J. (2023, February 1). *Interop 2023: Pushing interoperability forward*. Retrieved April 3, 2023, from https://webkit.org/blog/13706/interop-2023/

Fette, I., & Melnikov, A. (2011, November 1). *The websocket protocol*. Retrieved January 14, 2024, from https://datatracker.ietf.org/doc/html/rfc6455

Galli, M., Soares, R., & Oeschger, I. (2003, May 16). *Inner-browsing: Extending web browsing the navigation paradigm*. Retrieved January 11, 2024, from https://web.archive.org/web/20030810102320/http://devedge.netscape.com/viewsource/2003/inner-browsing/

Garrett, J. J. (2005, February 18). *Ajax: A new approach to web applications*. Retrieved January 11, 2024, from https://web.archive.org/web/20150910072359/http://adaptivepath.org/ideas/ajax-new-approach-web-applications/

Greif, S., & Burel, E. (2023a, January 11). *The state of js: Most used backend frameworks in 2022*. Retrieved January 11, 2024, from https://2022.stateofjs.com/en-US/other-tools/#backend_frameworks

Greif, S., & Burel, E. (2023b, January 11). *The state of js: Most used frontend frameworks in 2022*. Retrieved January 11, 2024, from https://2022.stateofjs.com/en-US/libraries/front-end-frameworks/

IBM. (2024a, January 15). *What is container orchestration?* Retrieved January 15, 2024, from https://www.ibm.com/topics/container-orchestration

IBM. (2024b, January 15). *What is containerization?* Retrieved January 15, 2024, from https://www.ibm.com/topics/containerization

Jackson, J. (2012, October 1). *Microsoft augments javascript for large-scale development*. Retrieved January 15, 2024, from https://web.archive.org/web/20131217223751/http://www.cio.com/article/717679/Microsoft_Augments_Javascript_for_Large_scale_Development

Linux Foundation. (2015, July 21). *New cloud native computing foundation to drive alignment among container technologies*. Retrieved January 15, 2024, from https://www.cncf.io/announcements/2015/06/21/new-cloud-native-computing-foundation-to-drive-alignment-among-container-technologies/

Linux Foundation. (2024, January 15). *About the open container initiative*. Retrieved January 15, 2024, from https://opencontainers.org/about/overview/

Minsky, M. (1980, June 1). *Telepresence*. Retrieved January 15, 2024, from https://web.media.mit.edu/~minsky/papers/Telepresence.html

Netscape. (1995, December 4). *Netscape and sun announce javascript*. Retrieved January 15, 2024, from https://web.archive.org/web/20070916144913/https://wp.netscape.com/newsref/pr/newsrelease67.html

Overflow, S. (2023, June 13). *2023 stack overflow developer survey: Web frameworks and technologies*. Retrieved January 13, 2024, from https://survey.stackoverflow.co/2023/#section-most-popular-technologies-web-frameworks-and-technologies

Python Software Foundation. (2024, January 15). *History of the software*. Retrieved January 15, 2024, from https://docs.python.org/3/license.html

Russell, A., & Berriman, F. (2015, October 8). *Progressive Web Apps: Escaping Tabs Without Losing Our Soul*. Retrieved January 11, 2024, from https://medium.com/@slightlylate/progressive-apps-escaping-tabs-without-losing-our-soul-3b93a8561955

Skarbez, R., Brooks Jr., F. P., & Whitton, M. C. (2017). A Survey of Presence and Related Concepts. *ACM Computing Surveys*, *50*(96), 1–39. https://doi.org/10.1145/3134301

van Rossum, G. (2008, August 11). *Why is python a dynamic language and also a strongly typed language?* Retrieved January 15, 2024, from https://wiki.python.org/moin/Why%20is%20Python%20a%20dynamic%20language%20and%20also%20a%20strongly%20typed%20language

Volpe, G. (2003, April 22). *Computational models of expressive gesture in multimedia systems*. Retrieved April 4, 2023, from https://theses.eurasip.org/theses/157/computational-models-of-expressive-gesture-in/download/

W3Schools. (2024, January 15). *Javascript history*. Retrieved January 15, 2024, from https://www.w3schools.com/js/js_history.asp

Web Audio API. (2021, June 17). Retrieved April 4, 2023, from https://www.w3.org/TR/webaudio/

WebRTC: Real-Time Communication in Browsers. (2023, March 6). Retrieved April 3, 2023, from https://www.w3.org/TR/webrtc/

WebXR Device API. (2023, March 3). Retrieved April 4, 2023, from https://www.w3.org/TR/webxr/

Ye, M., Wang, X., Yang, R., Ren, L., & Pollefeys, M. (2011). Accurate 3d pose estimation from a single depth image. *2011 International Conference on Computer Vision*, 731–738. https://doi.org/10.1109/ICCV.2011.6126310

You, E. (2021, September 29). *Introduction: The progressive framework*. Retrieved January 14, 2024, from https://vuejs.org/guide/introduction.html#the-progressive-framework

Appendix Listing

A. Appendix