ENTWURF – Projekt „CollabCanvas“

Version: 1.0

Erstellt am: 10.05.2022

Letze Änderung: 16.05.2022

Projektgeber: StuV DHBW Mannheim

DOKUMENTENVERSION

Version Nr.	Datum	Autor	Art der Änderung
0.1	10.05.2022	Maximilian Brieger, Victor Cislari, Tim Hartmann, Sinan Ermis, David Schatz, Kai Herbst	Erstellung erster Diagramme
0.2	15.05.2022	siehe oben	Erstellung des Dokuments
1.0	16.05.2022	siehe oben	Finalisierung der Version 1.0
1.1	21.05.2022	Tim Hartmann, Maximilian Brieger	Anpassung Datenbank, Hinzufügen REST API-Endpoints

Inhaltsverzeichnis

1 Ziele der Architektur

Im Vordergrund der Anwendung steht die kreative Kollaboration zwischen den Studierenden. Um diese so angenehm wie möglich zu gestalten, ist es notwendig den verwendeten Canvas konstant fehlerfrei und auf dem neusten Stand anzuzeigen. Das Problem unterteilt sich dabei zum einen in die serverseitige Software, die Canvas-Daten vom Client entgegennimmt, verarbeitet und neueste Änderungen stets wieder an die Clients zurückgibt. Zum anderen in die clientseitige Anwendung, die für die fehlerfreie Darstellung der eingehenden Daten zuständig ist und Änderungsaufträge an den Server schickt. Die nachfolgende genaue technische Planung der Software dient dazu diese Ziele zu erreichen und eine strukturierte Implementierung zu ermöglichen.

2 Typische Anwendungsszenarien

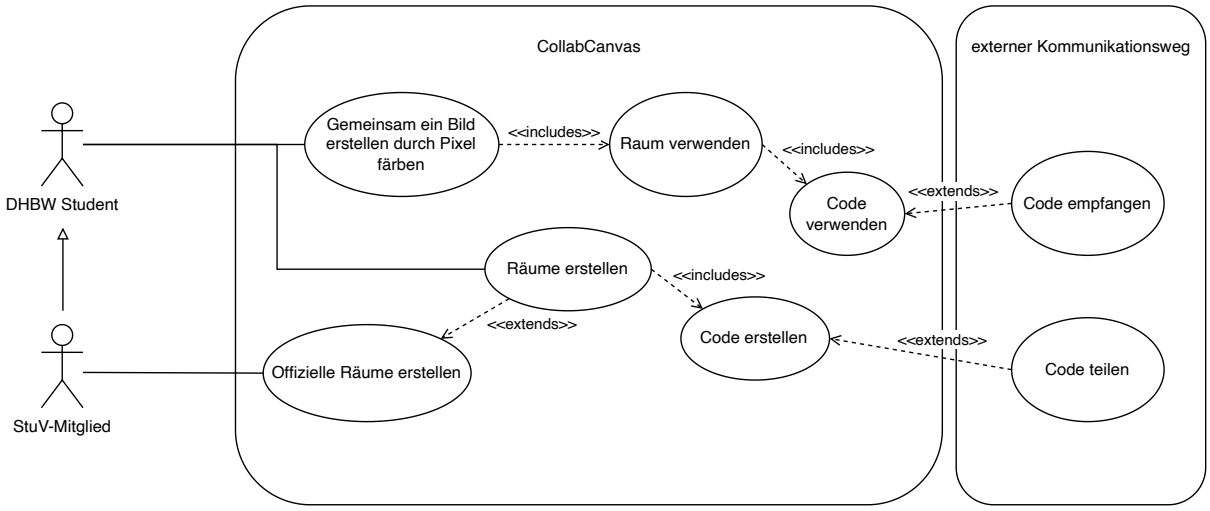

Abbildung 1: Darstellung von allgemeinen Use-Cases

Szenario 1 - Interaktion zwischen den Studiengängen:

Die StuV der DHBW Mannheim möchte die Kommunikation zwischen den verschiedenen Studiengängen stärken. Sie erstellt ein TeamCanvas mit einem Bereich für jeden Studiengang, auf welchem das gesamte Semester über gezeichnet werden kann. Die künstlerischen Ausführungen aller Kurse werden anschließend zusammengesetzt, sodass sich ein großes Bild ergibt.

Szenario 2 - Der gelangweilte Student:

Ein Studierender der DHBW Mannheim möchte in einer langweiligen Python-Vorlesung etwas Zeit totschlagen und mit seinen Freunden etwas zeichnen. Er erstellt einen Canvas und sie malen ein großes, erhabenes C.

Szenario 3 - Kreatives Kennenlernen:

Eine Gruppe von Studierenden möchte sich im Rahmen einer Kennenlern-Veranstaltung in Zusammenarbeit üben und gleichzeitig etwas übereinander erfahren. Sie eröffnen einen TeamCanvas auf dem jeder Teilnehmer etwas über sich Zeichen kann. Die Wartezeit zwischen dem Färben der Pixel wird hierbei auf eine geringe Zeit reduziert. Anschließend können sie sich über die geschaffenen Werke unterhalten.

3 Server-Client Architektur

Da sich die Server-Client Architektur für eine verteilte Anwendung gut eignet, wird CollabCanvas in einer Server-Client Architektur umgesetzt. Der Serverteil nimmt hier eine Rolle als Dienstleister an, der nicht mit direkt mit dem Nutzer interagiert. Auf diesem wird das Persistieren der Daten vorgenommen und die „Geschäftslogik", also das Validieren und Verarbeiten der Daten. Auch das Versorgen der Clients mit Daten und Empfangen der Daten und Aktionen von den Clients gehört zu der Aufgabe des Servers. Die Kommunikation erfolgt verbindungslos über das Internet per HTTP(S)-Protokoll. Der zentrale Dienstanbieter wird benötigt, um den Bearbeitungsstand an mehr als einen Teilnehmer auszuspielen. Die zentrale Datenhaltung ermöglicht dem Nutzer / Client beliebiges Ein- und Ausschalten, ohne andere zu beeinflussen. Der Clientteil der Architektur wird auf dem Endgerät des Nutzers ausgeführt und ermöglicht die Bedienung des Systems.

4 Wesentliche Komponenten

4.1 Logische Struktur

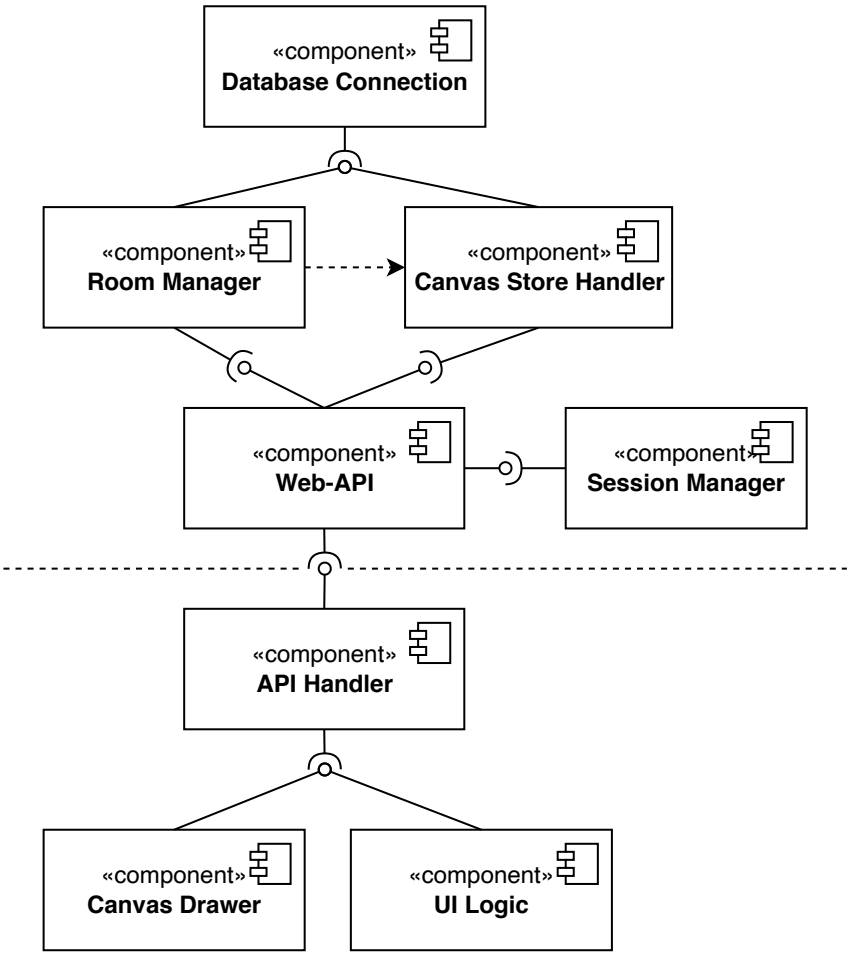

Abbildung 2: Überblick über die Komponenten und deren Schnittstellen

Die Anwendung ist geteilt in zwei logische und physisch durch Netzwerkkommunikation über getrennte Teile, das Frontend (unterhalb der gestrichelten Linie in Abbildung 2) welches als Schnittstelle zum Nutzer dient und das Backend (oberhalb der gestrichelten Linie in der Abbildung 2) welches für die Verteilung der Daten zuständig ist.

Die gestrichelte Linie stellt die Netzwerkkommunikation mit dem HTTP-Protokoll dar. Hier wird es mehrere HTTP-Endpoints geben, die nach dem Prinzip von REST aufgebaut sind, wie in Abbildung 3 zu sehen.

4.2 Datenfluss

Abbildung 3 ist eine Übersicht über den Datenfluss zwischen Komponenten in der Software, die verschiedenen hierarchischen Schichten zugeordnet sind.

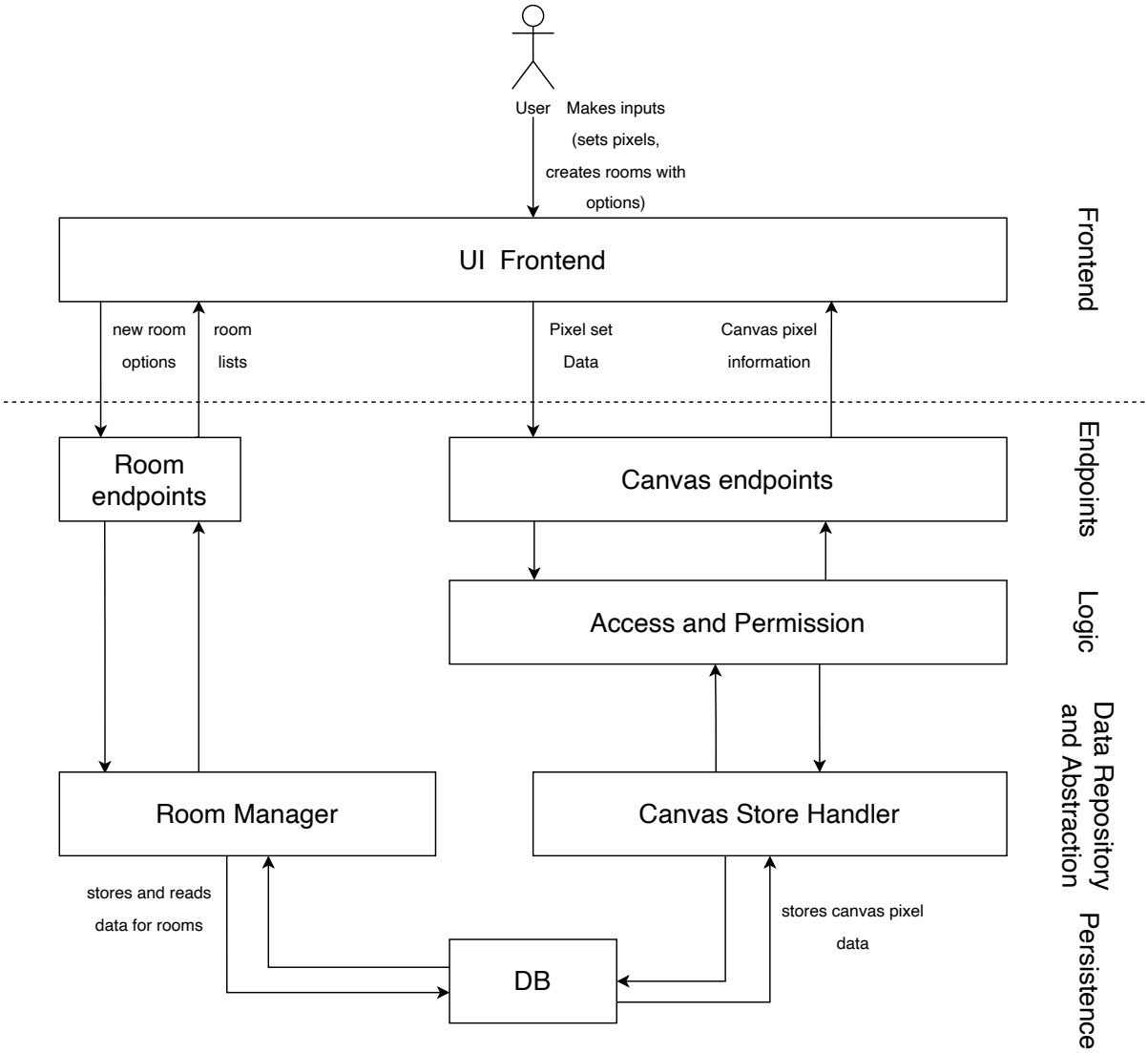

Abbildung 3: Übersicht über den hierarchischen Datenfluss in der Applikation

4.3 REST API-Endpoints

Die Kommunikation zwischen Frontend und dem Backend (Server) geschieht mit API-Endpoints, die an das REST-Paradigma angelehnt sind. Grundsätzlich werden Daten im JSON-Format übertragen. Es folgt eine Übersicht von geplanten Endpoints und deren Inhalte.

A. PUT /api/room/

Erstellung eines Raums:

Parameter: Name, Größe (Breite u. Höhe), CollabCanvas/TeamCanvas (=Anzahl RoomDivisions), Offizielle Markierung (true/false) (Hier wird true nur akzeptiert, wenn der Nutzer sich mit dem Master Code authentifiziert hat. Siehe auch PUT /api/system/auth)

Es wird eine Liste der Erstellten Codes zurückgegeben. Mögliche Fehler: Name schon vergeben, Größe oder Anzahl RoomDivisons nicht möglich oder offizieller Raum ist true, aber Nutzer kein Master.

```
{
    "admin": "<code>",
    "divisions": [
        "<code>",
        "<code>",

        ...
    ]
}
```

B. GET /api/room/<roomid>

Gibt alle Informationen über den Raum zurück, inklusive der Divisions.

C. GET /api/room/<roomid>/canvas/imagedata

Gibt die kompletten aktuellen Bilddaten für das Canvas im Speicherformat wie in 6.4 zurück. Zusätzlich wird am Ende der Bilddaten eine 64bit lange Update Identifikationsnummer übertragen.

D. GET /api/room/<roomid>/canvas/imageupdates?after=<updateid>

Gibt alle Pixelupdates zurück, die nach der gegebenen Update Identifikationsnummer übertragen wurden.

```
[
    {
        "updateid": <serverstartTS << 32 + counter>,
        "x": <x>,
        "y": <y>,
        "r": <r>,
        "g": <g>,
        "b": <b>,
        "lastEditedBy": <canvasdisplaynameid>
    },
]
```

Alternativ wird ein Fehler, falls das Update „after" nicht mehr im Speicher liegt und somit nicht mehr alle Updates zurückgegeben werden können. Um die Erkennung von diesem Sachverhalt zu gewährleisten, enthält die Update Identifikationsnummer neben einem fortlaufenden Zähler `counter` auch die Startzeit des Servers `serverstartTS`, da die Updates nur im Arbeitsspeicher gehalten werden.

E. GET /api/room/<roomid>/canvas/displaynames

Gibt eine Liste der Anzeigenamen für den aktuellen Canvas zurück.

F. GET /api/room/<roomid>/canvas/displayname/<displaynameid>

Gibt den Anzeigenamen mit der angegebenen displaynameid zurück.

G. PUT /api/room/<roomid>/canvas/pixel

Parameter: x, y, r, g, b
Setzt einen Pixel an der Angegebenen Position mit der Angegeben Farbe. Mögliche Fehler sind hier ein noch nicht abgelaufenen Pixelsetzverzögerung oder kein Zugriff auf den Canvas bzw. Bereich im Canvas.

H. POST /api/room/<roomid>/auth

Dies kann ein Admin-Code für den Raum sein, ein Code für den ganzen Canvas (bei nur einer RoomDivision) oder ein Code für einen Room. Auch kann der Master Code gesendet werden.

I. POST /api/system/auth

Hier kann der Master Code gesendet werden. Dies ermöglicht dem aktuellen Nutzer offizielle Räume zu erstellen.

4.4 Kontrollfluss

4.4.1 Einem Canvas beitreten

Der Beitritt in einen Canvas beginnt zunächst mit der Eingabe des dafür benötigten Zugangscodes. Die StuV hat mit ihrem Mastercode die Möglichkeit als Administrator beizutreten. Ist dieser richtig wird anschließend ein Benutzername gewählt, der beim Färben des Pixels gespeichert wird. Anschließend wird ein Session-Cookie erstellt, um den Benutzer identifizieren zu können. Parallel dazu wird zunächst die GUI, bzw. die Webseite geladen. Ist dies vollendet werden die Daten für den Canvas abgerufen und im Canvas-Element angezeigt. Als nächstes kann der Benutzer ein Pixel färben. Der Unterschied zwischen den beiden Modi CollabCanvas und TeamCanvas ist beim Beitritt folgender: Bei einem TeamCanvas wird der spezifische Code für einen der jeweiligen Teambereiche des Canvases eingegeben. Beim Beitritt in einem CollabCanvas wird der raumspezifische Code eingegeben, da es nur einen einzelnen Bereich gibt, in welchem Pixel gefärbt werden können.

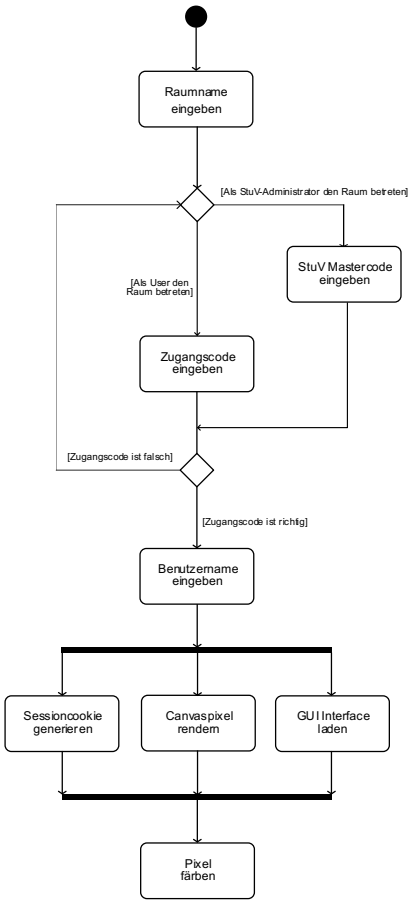

Abbildung 4: Ablauf bei Raumbeitritt

4.4.2 Raum erstellen

Um einen Raum zu erstellen, muss zunächst der Name des Raums und der gewünschte Modus gewählt werden. Anschließend wird entweder im CollabCanvas-Modus die Größe des gesamten Canvas festgelegt oder im Team-Canvas Modus die Größe und Anzahl der einzelnen Abschnitte. Die StuV hat mit Hilfe ihres Mastercodes außerdem noch die Möglichkeit den Raum als offiziell zu markieren. Danach wird automatisch der Code generiert, welcher für den Beitritt in den Raum verwendet werden kann.

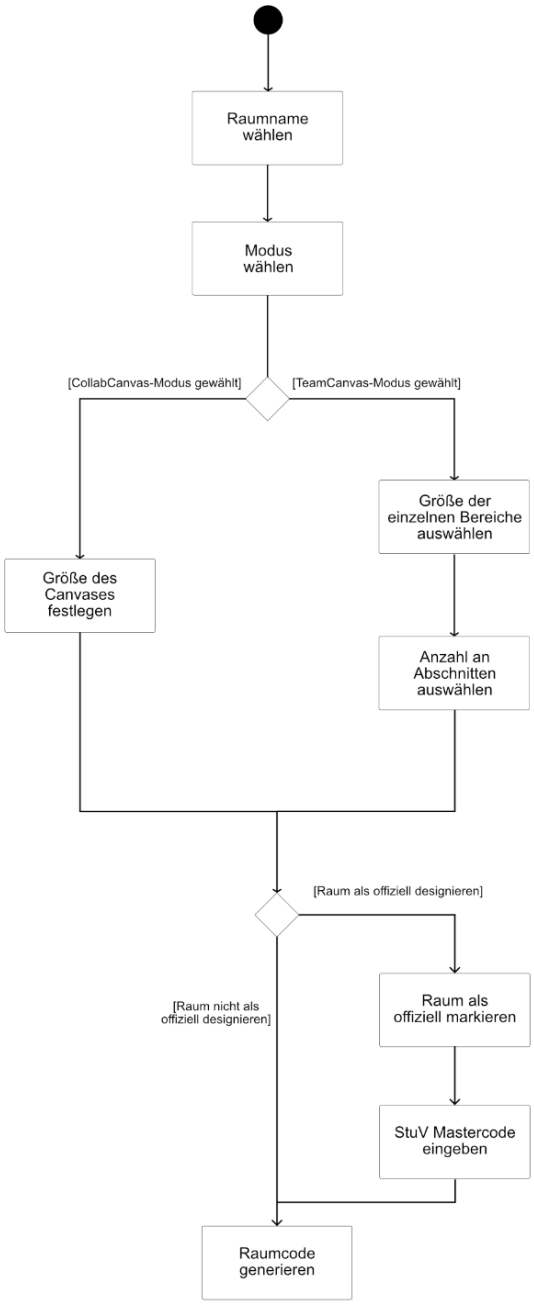

Abbildung 5: Ablauf bei Raumerstellung

4.5 Physikalische Sicht

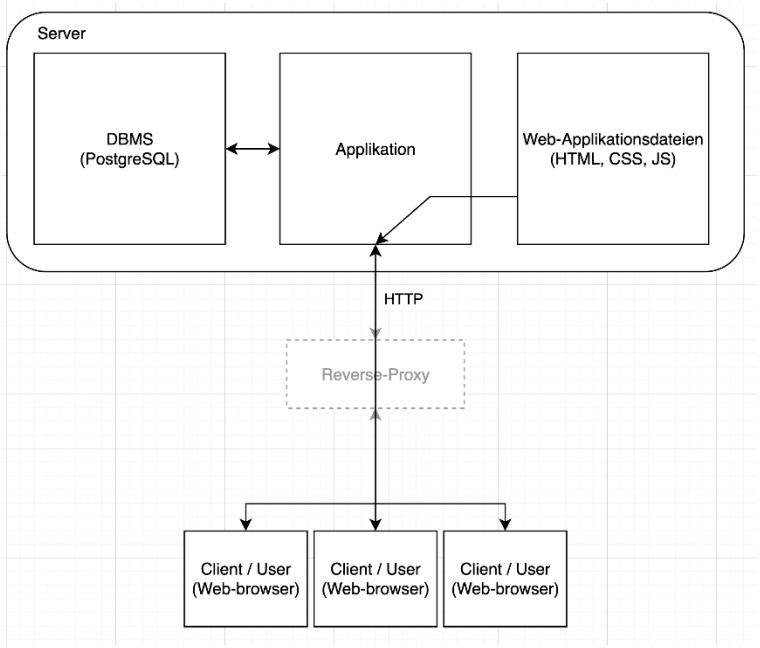

Abbildung 6: Überblick über das Deployment

Um ein einfacheres Deployment zu ermöglichen, wird die (statische) Web-Applikation über den gleichen HTTP Server, der für die Übertragung der Daten benutzt wird, übertragen.

Der HTTP Server selbst läuft direkt im Applikationsprozess. Dies macht das Deployment einfacher, da die Applikation nach außen über einen Port die Website (also den Website-Code mit den HTML, CSS und JS Dateien) übertragt zusammen mit den dynamischen Daten und Anfragen (z.B. Pixel setzen).

4.6 Anforderungen an die Hardware

In der physischen Schicht muss ein Client zur Verfügung stehen, welcher in der Lage ist, einen der unterstützten Web-Browser auszuführen. Für den Betrieb von PHP und PostgreSQL sehen wir mindestens 8 CPU-Kerne aus der 7. Prozessorgeneration von Intel und 4 GB RAM (exkl. Betriebssystem) vor. Passend zur Anzahl an parallele Nutzer muss die Speicherkapazität skaliert werden.

5 User-Interface

Das Anzeigen der UI wird, wie bereits beschrieben, im Browser stattfinden. Für die Anzeige des eigentlichen Canvases innerhalb der Webseite wird das <canvas>-HTML Tag verwendet. Dieses ist speziell für das Zeichnen von 2D-Anwendungen vorgesehen und erfüllt alle Anforderungen, die für das Rendern des Canvases benötigt werden.

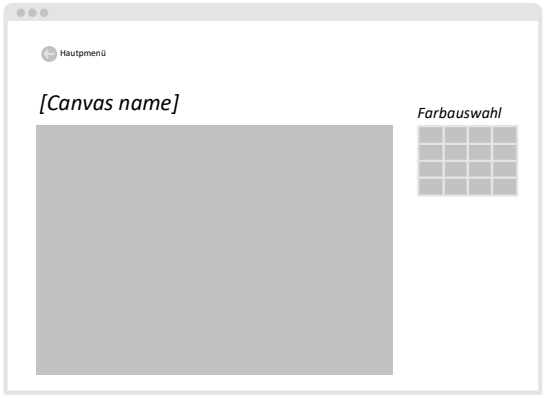

Abbildung 7: UI nach Einwählen in ein Canvas

5.1 Rendern des Canvas

Wie das Rendern im genauen umgesetzt wird, wird nun beschrieben. Zunächst muss das HTML-Canvas-Element in einzelne Rechtecke, die jeweils ein Pixel des Canvas anzeigen, unterteilt werden. Die Anzahl der Pixel, die im Canvas-Element angezeigt werden, hängt von der derzeit ausgewählten Zoom-Stufe ab. Umso größer die Zoom-Stufe umso weniger Pixel werden angezeigt, beziehungsweise ein kleinerer Ausschnitt des Canvases. Um den richtigen Ausschnitt des Canvas anzuzeigen, muss die Position des Users überwacht werden. Anhand dieser Position und der Größe des Ausschnitts werden anschließend die benötigten Daten ausgewählt und im Canvas-Element angezeigt.

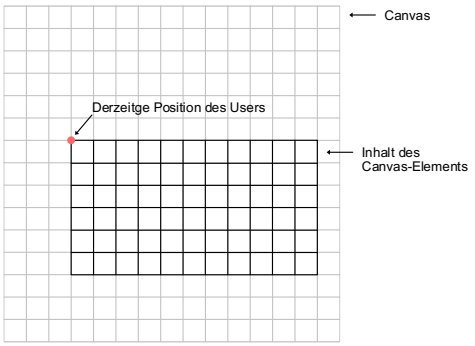

Abbildung 8: Rendering des Canvas

Bewegt sich der User innerhalb des Canvas muss so nur die Position aktualisiert und der daraus resultierende Canvas neu berechnet werden.

6 Verwendete Technologien

6.1 Betriebssystem

Die Anwendung ist sowohl auf Seite des Servers als auch auf der Seite des Clients vom Betriebssystem unabhängig. Da die Anwendung als Web-App realisiert wird, benötigt der Client demnach nur einen funktionierenden Browser. Wie auch in Abschnitt 4.6 genauer beschrieben, wird die serverseitige Software in der Programmiersprache PHP entwickelt - das Betriebssystem muss demnach in der Lage sein solchen Code zur Ausführung zu bringen.

6.2 Code Style

Viele Elemente im Code benötigen Namen. Diese müssen verständlich und eindeutig sein. Deswegen werden folgende Regeln festgelegt:

1. Englisch als Sprache
2. Klassennamen in PascalCase
3. Konstanten in UPPER_CASE_SNAKE_CASE
4. camelCase für alle Weiteren Namen

Für die Datenbank speziell:

1. Attribute in snake_case
2. Tabellennamen in Singular und PascalCase

6.3 Programmiersprachen und Bibliotheken

Für den Client werden die für Webanwendungen üblichen Sprachen benutzt: Die Auszeichungssprache HTML 5, Stylesheetsprache CSS3 und die Programmiersprache JS.

Für die Umsetzung des Backends wird die Programmiersprache PHP benutzt. Es wird zusätzlich das Framework Amp eingesetzt (https://amphp.org) um den HTTP Server direkt in dem PHP-Prozess zu ermöglichen und parallele Ausführung zu erlauben.

6.4 Datenbanken

Die Datenbanken werden mit dem Datenbankmanagementsystem (DBMS) PostgreSQL verwaltet. PostgreSQL ist ein freies, Open-Source objekt-relationales DBMS. (https://www.postgresql.org/).

Das Datenbankschema aus Abbildung 9 wird in diesem DBMS als Datenbank implementiert.

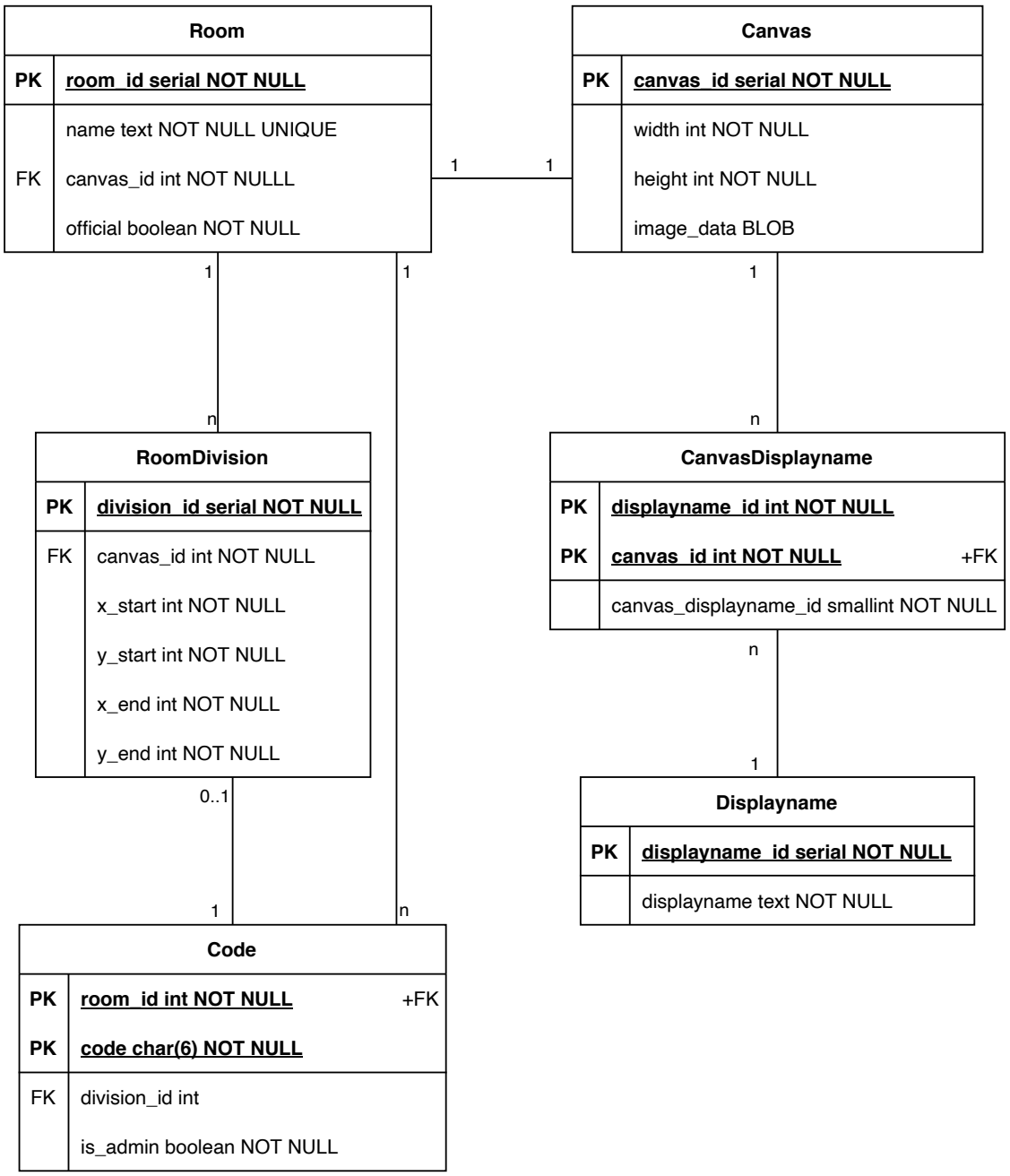

Abbildung 9: Übersicht über die Tabellen in der Datenbank

Die Inhalte von Canvas.image_data sind wie folgt strukturiert:

Für jeden Pixel wird folgende Bytesequenz abgelegt:

R uint (1 byte)	G uint (1 byte)	B uint (1 byte)	LastEditedBy uint (1 byte)

- Die Pixel werden zeilenweise (width-Richtung) abgelegt. Dabei wird bei x=0 begonnen, bis x=width jeweils um eins inkrementiert. Bei x=width wird x auf 0 gesetzt und y um eins inkrementiert. Dies wird so lange wiederholt bis y=height und x=width ist.

LastEditedBy enthält die canvas_displayname_id für den Nutzer, der diesen Pixel zuletzt editiert hat. Die Tabelle CanvasDisplayname enthält die Zuweisung von Displayname zu Canvas. Sie enthält auch canvas_displayname_id, eine Canvas-spezifische ID für den jeweiligen Displayname, um diese klein genug zu halten, um in das LastEditedBy Feld jedes Pixels zu passen. Die Displayname-Tabelle ist ausgelagert, um die Größe dieser der CanvasDisplayname Tabelle zu reduzieren.

7 Werkzeuge

Entwicklungsumgebung:

Um eine einfache Zusammenarbeit zu gewährleisten, wird die Entwicklungsumgebung PHPStorm von JetBrains genutzt.

Versionsverwaltung:

Zur zentralen Verwaltung und Versionierung des Projektes wird das Versionierungs-System GIT unter Verwendung des Cloud-Dienstleisters GitHub eingesetzt.

Docker:

Um die Plattformunabhängigkeit der Anwendung zu gewährleisten wird die serverseitige Software mit Hilfe des Containerisierungstools Docker in einem Container isoliert. Im Rahmen der Entwicklung werden Docker-Instanzen eingesetzt.

8 Abbildungsverzeichnis