HTL Leonding
next level

HÖHERE TECHNISCHE
BUNDESLEHRANSTALT
LEONDING

Memoryland

DIPLOMARBEIT

verfasst im Rahmen der

Reife- und Diplomprüfung

an der

Höheren Abteilung für Informatik & Medientechnik

Eingereicht von:
Arwed Schnalzenberger, 5BHIF
Isabel Schnalzenberger, 5AHITM

Betreuer:
Prof. Dipl.-Ing. Christian Aberger

Projektpartner:
Keine

Leonding, 26. Februar 2025

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Die Arbeit wurde bisher in gleicher oder ähnlicher Weise keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Die vorliegende Diplomarbeit ist mit dem elektronisch übermittelten Textdokument identisch. Leonding, 26. Februar 2025

Arwed Schnalzenberger, 5BHIF

Isabel Schnalzenberger, 5AHITM

Danksagung

Als erstes möchten wir Prof. Christian Aberger für seine Unterstützung als Betreuungslehrer danken.

Wir möchten unseren Eltern herzlich danken für ihre Unterstützung während der Erstellung dieser Diplomarbeit. Ihr Beistand, ihre Geduld und ihre Ermutigung haben uns geholfen, diese Herausforderung erfolgreich zu bewältigen.

Abstract

Memories in the form of photos and videos are a valuable part of life, yet they are often lost or rarely revisited. To keep these memories alive, an appealing presentation and easy accessibility are essential.

The diploma project Memoryland was developed to address exactly this issue. It allows personal memories to be experienced in an interactive and animated form. As part of this project, a web application was created that transforms photos into engaging animations. Users can generate videos from these animations and share them with their friends.

A special focus was placed on creating an immersive experience. Memorylands enable users to relive their memories in virtual environments such as a forest or an island. Additionally, great care was taken to ensure that the functionalities are as intuitive and comfortable as possible.

Ultimately, this project aims to make it easier for people to preserve their memories in a creative and entertaining way while allowing them to rediscover them effortlessly.

Zusammenfassung

Erinnerungen in Form von Fotos und Videos sind ein wertvoller Bestandteil des Lebens und doch gehen sie oft verloren oder werden selten angesehen. Um diese Erinnerungen lebendig zu halten, sind daher eine ansprechende Präsentation und einfache Zugänglichkeit essenziell.

Die Diplomarbeit Memoryland wurde entwickelt, um genau dieses Problem zu lösen. Sie ermöglicht es, persönliche Erinnerungen in einer interaktiven und animierten Form zu erleben. Im Rahmen dieser Arbeit wurde eine Web-Anwendung erstellt, die Fotos in ansprechende Animationen umwandelt. Daraus können Nutzer:innen nun Videos generieren und an ihre Freunde weitergeben.

Besonderes wurde hierbei auf eine immersive Erfahrung geachtet. Memorylands ermöglichen es, Erinnerungen in einer virtuellen Umgebung, wie einem Wald oder einer Insel, zu erleben. Es wurde auch darauf geachtet, dass die Funktionalitäten so intuitiv und gemütlich wie möglich sind.

Schlussendlich soll unsere Arbeit es Menschen erleichtern, ihre Erinnerungen auf eine kreative und unterhaltsame Weise zu wahren und leicht wiederzuentdecken.

Inhaltsverzeichnis

1 Einleitung

1.1 Ursprüngliche Idee

1.2 Ausgangssituation

Herkömmliche Familien-/Fotoalben stehen normalerweise wegen ihres Gewichtes zuhause und falls man dann einmal einem Freund bei einer Party ein Foto schnell zeigen möchte, hat man eher das Handy als ein ganzes Fotoalbum dabei.

Zwar gibt es schon Tools, welche die Fotos nur präsentieren, aber wir wollen die Fotos zeitgemäß für jeden leicht verfügbar und transportabel animieren.

1.3 Untersuchungsanliegen

1.3.1 Arwed Schnalzenberger, 5BHIF

Die vorliegende Untersuchung zielt darauf ab, die effiziente Speicherung umfangreicher Mengen von Videos und Bildmaterial in Cloud-Umgebungen zu untersuchen sowie die Prozesse zur Erstellung und Bearbeitung von Videos auf der Backend-Ebene zu erforschen.

1.3.2 Isabel Schnalzenberger, 5AHITM

Die vorliegende Untersuchung zielt darauf ab, die potenzielle Steigerung der Akzeptanz von Online-Darstellungen durch die Integration von 3D-Visualisierungen einer Bildergalerie zu erforschen.

2 Umfeldanalyse

Bereits vor der Entwicklung von Memoryland, gab es bestehende Lösungen zur Verwaltung und Präsentation von Erinnerungen. Bevor mit dem Projekt begonnen wurde, wurden diese analysiert und die Erkenntnisse zu den Stärken und Schwächen dieser Systeme notiert. Daraus entstanden dann die Anforderungen für das finale System.

2.1 Analyse der vorhandenen Systeme

2.1.1 OneDrive

Vorteile: Das System bietet eine cloudbasierte Speicherung für Bilder und andere Medien. So können Nutzer:innen ihre Dateien zentral ablegen und von verschiedenen Geräten aus darauf zugreifen.

Nachteile: Eine Integration von Bildern in interaktive Formate ist nicht vorhanden. Dadurch bleibt die Nutzung der gespeicherten Bilder auf eine einfache Anzeige und Verwaltung beschränkt.

Zusammenfassung: OneDrive bietet eine sichere und cloudbasierte Speicherung von Bildern und anderen Medien. Es fehlen Funktionen zur Bildbearbeitung und zur Integration von immersiven Formaten. [1]

2.1.2 Google Photos

Vorteile: Das System stellt eine Oberfläche für eine Verwaltung und Anzeige von Bildern zur Verfügung. Dadurch können Nutzer:innen ihre Bilder organisieren und betrachten.

Nachteile: Eine Integration von Bildern in interaktive Formate ist nicht vorhanden. Somit ist es nicht möglich sie in erweiterte Präsentationsformen einzubinden und beschränkt die Nutzung der Bilder auf Verwaltungs- und Anzeigezwecke.

[1] Informationen zu OneDrive stammen von [1]

Zusammenfassung: Google Photos ermöglicht eine Verwaltung und Anzeige von Bildern, bietet jedoch keine Möglichkeit zur Umwandlung von Bildern in interaktive Formate. [2]

2.1.3 Animoto

Vorteile: Das System ermöglicht die Umwandlung von Bildern in animierte Diashows, wodurch Bilder in eine dynamische Präsentationsform dargestellt werden können.

Nachteile: Die erstellten Diashows enthalten jedoch keine immersiven Komponenten.

Zusammenfassung: Animoto bietet die Möglichkeit, Bilder in Diashows umzuwandeln. Diese Funktion gleicht einem animierten Fotoalbum und bietet keine immersiven Erlebnisse, wie sie in Memoryland vorgesehen sind. [3]

2.1.4 Zusammenfassung

Die Marktanalyse zeigt, dass zwar verschiedene Plattformen grundlegende Funktionen zur Speicherung und Präsentation von Bildern bieten, jedoch keine immersiven Erlebnisse ermöglichen. OneDrive und Google Photos kümmern sich um die sichere Speicherung und Verwaltung. Animoto ermöglicht die Erstellung von Diashows, jedoch ohne die Möglichkeit, Bilder in virtuelle Umgebungen zu integrieren.

2.2 Funktionale Anforderungen

2.2.1 Benutzerverwaltung

Das System soll eine Benutzerverwaltung bereitstellen, die eine Registrierung und Authentifizierung ermöglicht. Für eine sichere Authentifizierung erfolgt die Anmeldung über Azure AD B2C.

2.2.2 Bilder-Upload

Nutzer:innen sollen die Möglichkeit haben, Bilder hochzuladen und in Alben zu organisieren. Um Erinnerungen gut organisieren zu können sollen hochgeladene Bilder und

[2]Informationen zu Google Photos stammen von [2]
[3]Informationen zu Animoto stammen von [3]

Alben umbenannt werden können. Darüber hinaus soll das System eine Suchfunktion bieten, damit Nutzer:innen ihre Bilder schnell wiederfinden können. Um den Upload-Prozess zu vereinfachen, soll es möglich sein, mehrere Bilder auf einmal hochzuladen. Falls große Mengen an Bildern übertragen werden, soll ein Resumable Upload verwendet werden können, sodass unterbrochene Uploads fortgesetzt werden können.

2.2.3 Präsentation der Erinnerungen

Das System soll es ermöglichen, mehrere Memorylands zu erstellen, die dem gleichen oder unterschiedlichen Typen angehören können. Ein Typ definiert dabei eine eigene Szene, zum Beispiel eine Wald- oder Inselumgebung. Nutzer:innen sollen in einem Memoryland ihre Bilder an bestimmten Stellen platzieren können, um eine immersive Darstellung ihrer Erinnerungen zu ermöglichen.

2.2.4 Sharing-Funktion

Memorylands sollen mit anderen Personen geteilt werden können. Dies soll es Nutzern/-Nutzerinnen ermöglichen, ihre Erinnerungen mit Familie und Freunden zu teilen.

2.2.5 Security

Die hochgeladenen Bilder und Alben sollen ausschließlich für den jeweiligen Nutzer/die jeweilige Nutzerin verfügbar sein. Andere Nutzer sollen keinen Zugriff auf fremde Bilder erhalten. Wenn Memorylands mit anderen geteilt werden, soll deshalb eine deutliche Warnung darauf hinweisen, dass die Inhalte für andere sichtbar werden. Zudem soll es jederzeit möglich sein, eine Freigabe wieder zurückzuziehen, sodass geteilte Memorylands ungültig werden und somit nicht mehr aufrufbar sind.

2.3 Nicht funktionale Anforderungen

2.3.1 Security

Die Benutzeranmeldung erfolgt ausschließlich über Azure AD B2C, um eine sichere Authentifizierung zu gewährleisten. Hochgeladene Bilder und Memorylands dürfen nur für den jeweiligen Nutzer/die jeweilige Nutzerin zugänglich sein. Die Datenübertragung erfolgt über TLS-verschlüsselte Verbindungen (HTTPS). Zudem sollen Nutzer:innen

jederzeit die Möglichkeit haben, ihre Daten zu löschen („Recht auf Vergessenwerden“).

2.3.2 Benutzerfreundlichkeit & Design

Die Webanwendung muss eine intuitive Benutzeroberfläche bieten, sodass Nutzer:innen ihre Erinnerungen möglichst einfach hochladen, verwalten und präsentieren können. Suchleisten und das Sortieren der Daten erleichtert den schnellen Zugriff auf unterschiedliche Inhalte.

3 Architektur

3.1 Architekturdiagramm

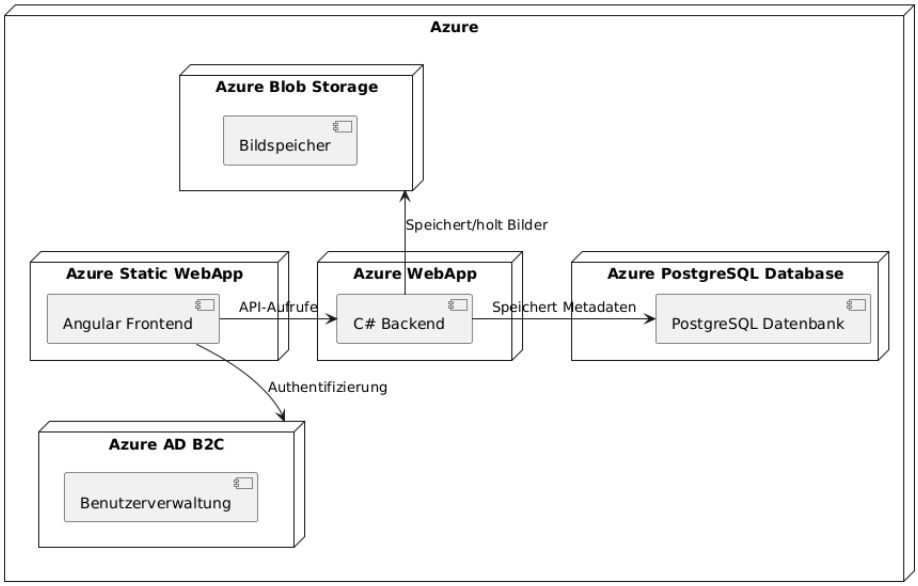

Abbildung 1: Architekturdiagramm

Das oben stehende Diagramm 1 zeigt die Architektur von Memoryland, welches auf der Azure Cloud läuft. Hierbei gibt es mehrere Bestandteile, einschließlich eines Frontends, eines C# Backends, einer Benutzerverwaltung über Azure AD B2C und abschließend einer PostgreSQL-Datenbank und einem Blob-Storage für die Datenhaltung.

3.2 Datenhaltung

In unserer Datenverwaltung kommen zwei Hauptkomponenten zum Einsatz, welche sich um die Speicherung und Verwaltung der Bilddaten kümmern. Ein Azure Blob Storage und eine Azure PostgreSQL Datenbank.

3.2.1 Azure Blob Storage

Hier werden die Bilder der Benutzer:innen gespeichert. Diese werden ohne eine Struktur abgelegt, außer dass für jede/n Benutzer:in ein separater Container erstellt wird. Somit

wird der kostenpflichtige Zugriff auf den Blob Storage minimiert und die Verwaltung so einfach wie möglich gehalten. [4] [5]

3.2.2 Azure PostgreSQL Datenbank

Diese Datenbank speichert die Metadaten zu den Bildern. Sie hält zum Beispiel Informationen darüber, in welchen Alben sich welche Erinnerungen befinden und wem sie gehören neben anderen Details. Zusätzlich verwaltet die PostgreSQL-Datenbank die Struktur der Memorylands. Hier stehen unter anderem Informationen, welches Bild zu welchem Memoryland gehört, und wo diese im Memoryland platziert werden soll. [6] [7]

3.3 Benutzerverwaltung

Die Benutzerverwaltung erfolgt über Azure AD B2C. Die Benutzer:innen authentifizieren sich im Frontend mithilfe von MSAL (Microsoft Authentication Library), während das Backend ihre Berechtigungen gegen Azure AD B2C überprüft. [8] [9]

3.4 Backend

Das Backend dient dazu, die Daten von Benutzern und Benutzerinnen mit den erforderlichen Berechtigungen, zu verwalten. Zu diesen Daten gehören unter anderem Memorylands, Alben und Bilder. [10]

3.5 Frontend

Das Frontend dient als Schnittstelle zum Benutzer/zur Benutzerin und bietet eine grafische Oberfläche zur Verwaltung ihrer Daten. Zudem ermöglicht es eine Darstellung der gespeicherten Bilder und Memorylands. [11]

[4] Mehr Info dazu im Kapitel: 5.5.1
[5] Was ist *Azure Blob Storage* und warum verwenden wir es: Kapitel 5.2.5
[6] Mehr Info dazu im Kapitel: 4
[7] Was ist *Azure PostgreSQL Datenbank* und warum verwenden wir es: Kapitel 5.2.4
[8] Mehr Info zu Azure AD B2C im Kapitel: 5.2.5
[9] Mehr Info zu MSAL im Kapitel: 5.2.5
[10] Mehr Info dazu im Kapitel: 5
[11] Mehr Info dazu im Kapitel: 6

4 Datenmodell

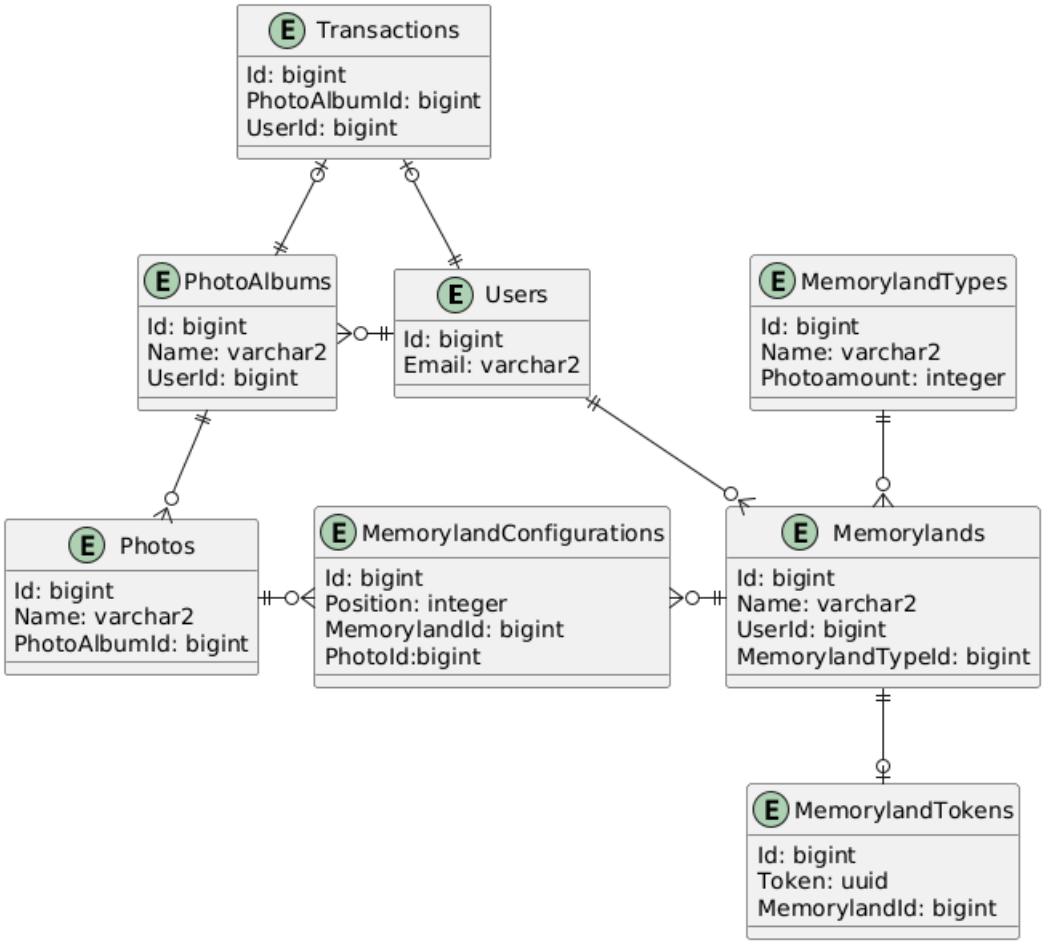

Abbildung 2: Datenmodell

Die Datenbank besteht aus insgesamt acht Tabellen, die das Backend für die Verwaltung von Benutzern, Fotoalben, deren Fotos und der Memorylands benötigt.

4.1 Users

In der Tabelle “Users” werden Benutzer von Memoryland gespeichert, die sich über Azure AD B2C registriert haben. Dabei wird ein Benutzer wird mit deren Email automatisch gespeichert, sobald er oder sie eine Aktion ausführt, die eine Speicherung im Backend erfordert.

Der Benutzername wird nicht gespeichert, da er im Backend nicht benötigt wird und eine regelmäßige Aktualisierung bei Namensänderungen unnötigen Aufwand verursachen würde. Die E-Mail spielt in Memoryland nur als eindeutige Identifikation eines Benutzers eine Rolle.

4.2 PhotoAlbums

Diese Tabelle speichert die von Benutzern erstellten Fotoalben. Jedes Album ist eindeutig einem Benutzer zugeordnet, wodurch sichergestellt wird, dass ausschließlich der jeweilige Besitzer Zugriff darauf hat. Andere Benutzer können weder auf die Alben noch auf deren Inhalte zugreifen.

Die Alben selbst existieren als Einträge in der Datenbank, während die darin enthaltenen Fotos im Azure Blob Storage gespeichert werden. Das Konzept der Fotoalben wird also nur in der Datenbank abgebildet und hat keine direkte Entsprechung in der Speicherstruktur des Blob Storage. [12] Diese Struktur dient dazu, den direkten Zugriff auf den Azure Blob Storage zu minimieren und zu vereinfachen. Gleichzeitig ermöglicht sie dem Frontend eine sortierte und strukturierte Anzeige der Fotoalben.

Jeder Benutzer kann mehrere Fotoalben anlegen, wobei ein Album beliebig viele Fotos enthalten kann – von keinem bis zu einer unbegrenzten Anzahl.

4.3 Photos

In dieser Tabelle werden die einzelnen Fotos mit Referenzen auf ihre jeweiligen Alben gespeichert. Das verwalten einzelner Fotos ist ausschließlich dem Benutzer vorbehalten, dem dieses gehört. Allerdings können Fotos anderen Benutzern innerhalb von Memorylands durch die Nutzung von SAS-Tokens angezeigt werden.

Jedes Foto gehört genau einem Album an, und innerhalb eines Albums darf kein weiteres Foto denselben Namen tragen. Dies gewährleistet eine eindeutige Zuordnung der Dateien für die Benutzer.

Im Azure Blob Storage werden zur Identifikation nicht die Namen der Fotos, sondern die eindeutigen Identifikationsnummern verwendet. Dies ermöglicht es, dass ein

[12] Mehr Informationen zu “Azure Blob Storage” im Kapitel: 5.5.1

Benutzer mehrere Fotos mit identischen Namen besitzen kann, solange diese sich in unterschiedlichen Alben befinden. [13]

4.4 Memorylands

Ein Memoryland beinhaltet eine Sammlung von Fotos und ist immer einem bestimmten Memoryland-Typ zugeordnet. Der Typ eines Memorylands bestimmt unter anderem die maximale Anzahl an Fotos, die darin enthalten sein können.

Alle Benutzer:innen können mehrere Memorylands besitzen, die jeweils eine festgelegte Anzahl von Fotos an bestimmten Positionen enthalten. Die Anzahl der erlaubten Fotos pro Memoryland hängt jeweils vom Memoryland-Typ ab.

4.5 MemorylandTypes

Diese Tabelle definiert verschiedene Typen von Memorylands. Ein Memoryland-Typ entspricht einer Szene in Unity und legt die maximale Anzahl an Fotos fest, die in einem Memoryland gespeichert werden können.

Da die Typen von den Unity-Szenen abhängen, die nur durch ein Deployment geändert werden können, und es keinen anderen direkten Weg gibt, festzustellen, wie viele Fotos in welcher Szene benötigt werden, ist es erforderlich, die Memoryland-Typen manuell in die Datenbank einzutragen.

4.6 MemorylandConfigurations

Diese Tabelle speichert die spezifischen Konfigurationen eines Memorylands, das heißt, sie definiert, welche Fotos an welchen Positionen innerhalb eines Memorylands angezeigt werden.

Ein Memoryland kann eine beliebige Anzahl von Konfigurationen haben, wobei im Backend überprüft wird, ob die angegebene Position tatsächlich existiert. Zudem wird sichergestellt, dass jede Position innerhalb eines Memorylands nur einmal belegt wird. Es ist jedoch möglich, dass ein Foto in einem Memoryland mehrfach vorkommt.

[13] Siehe die Fußnote: 12

Diese Flexibilität ermöglicht es, Fotos an verschiedenen Stellen im Memoryland anzuzeigen, ohne das die Reihenfolge der Speicherung oder anderes wichtig wäre.

4.7 MemorylandTokens

Diese Tabelle speichert die Zugriffs-Tokens eines Memorylands. Pro Memoryland kann immer nur ein Token gleichzeitig existieren. Benutzer:innen können mithilfe dieses Tokens auf ein Memoryland zugreifen und es anzeigen lassen, egal ob ihnen das Memoryland gehört oder nicht.

Die Tokens sind von der Datenbank generierte GUIDs. Eine separate Tabelle für die Tokens wurde aus zwei wesentlichen Gründen angelegt: Zum einen ist diese Struktur eine Lösung die eine automatische Erstellung der Tokens durch die Datenbank ermöglicht, was den Aufwand im Backend reduziert und eine sichere sowie eindeutige Identifikation gewährleistet. Zum anderen folgt die Trennung der Tokens in eine eigene Tabelle dem Prinzip der "Separation of Concerns".

"*Separation of Concerns*" ist ein Prinzip der Softwareentwicklung, bei dem verschiedene Aspekte eines Systems oder einer Anwendung klar voneinander getrennt werden. In diesem Fall enthält die Tabelle der Memorylands allgemeine Informationen über die erstellten Memorylands, während die Tabelle der Tokens die separate Verantwortung für die Tokens übernimmt. Durch diese Trennung ist es einfacher neue Sicherheitsmechanismen oder Berechtigungsstufen oder anderes hinzuzufügen oder zu entfernen. [14]

4.8 Transactions

Diese Tabelle dient der Verwaltung von Transaktionen, die verwendet werden, um das Hochladen großer Fotoalben effizienter zu gestalten. Besonders bei umfangreichen Bildersammlungen, wie sie beispielsweise nach einer Reise oder einem Event anfallen, kann der Upload-Prozess durch verschiedene Faktoren unterbrochen werden – sei es durch eine instabile Internetverbindung, einen Verbindungsabbruch oder anderem. Um zu verhindern, dass Benutzer:innen in solchen Fällen den gesamten Upload erneut starten müssen, wird ein Mechanismus für „Resumable Uploads“ verwendet.

[14] Alle Informationen zu "Separation of Concerns" stammen von: [4]

In dieser Diplomarbeit werden solche Upload-Vorgänge synonym als „Transaktionen“ und „Resumable Uploads“ bezeichnet. Jede:r Benutzer:in kann zu einem bestimmten Zeitpunkt genau eine aktive Transaktion haben. In der Tabelle wird außerdem eine Referenz auf das jeweilige Fotoalbum gespeichert, in das die Bilder hochgeladen werden. Dies ermöglicht es, einen begonnenen Upload gezielt einem bestimmten Album zuzuordnen und den Prozess genau dort fortzusetzen, wo er unterbrochen wurde, was Benutzer:innen dabei helfen kann, sich daran zu erinnern, wofür der Upload gedacht war.

4.8.1 Probleme bei der Implementierung

Ursprünglich war vorgesehen, zusätzlich den lokalen Dateipfad der hochgeladenen Dateien in der Datenbank zu speichern. Dieses Feature konnte jedoch aufgrund von Sicherheitsrichtlinien nicht umgesetzt werden, da das Speichern vollständiger Dateipfade potenzielle Risiken im Hinblick auf Datenschutz und Zugriffsbeschränkungen mit sich gebracht hätte.

Für den Upload ganzer Fotoalben wurde die HTML5-Funktion *webkitdirectory* genutzt, die es ermöglicht, komplette Verzeichnisse auszuwählen und deren Inhalte hochzuladen. Allerdings erlaubt *webkitdirectory* aus Sicherheitsgründen keinen Zugriff auf absolute Dateipfade. Stattdessen werden nur relative Pfade bereitgestellt, wodurch die ursprüngliche Idee, vollständige Dateipfade in der Datenbank zu speichern, technisch nicht umsetzbar war. [15]

[15] Alle Informationen zu dem “webkitdirectory” stammen von: [5]

5 Backend-Umsetzung

5.1 Einrichtung

5.1.1 Einrichtung des Azure Blob Storage

Damit alles funktioniert und damit der häufige Zugriff auf die Bilder möglichst effizient ist, wurden folgende Einstellungen im Azure Portal konfiguriert. [16]

SAS-Tokens

Um den Zugriff auf Bilder über Shared Access Signatures (SAS) zu ermöglichen, muss in die Konfiguration "Allow storage account key access" aktiviert werden.

SAS-Tokens ermöglichen es uns, bestimmte Zugriffsrechte zu definieren, wie zum Beispiel nur ein Lesezugriff. Das ermöglicht es unserem System, mittels Unity direkt auf die gespeicherten Bilder zuzugreifen.

Mithilfe von SAS-Tokens die im Backend generiert und an Unity übergeben werden, kann Unity nun auf die benötigten Bilder zugreifen, ohne vollständige Zugriffsdaten des Storage Accounts zu benötigen. [17]

Blob Access Tier

Um den häufigen Zugriff auf Bilder effizienter zu gestalten, sollte in der Konfiguration des Blob Storage der Access Tier die Standardstufe auf "Hot" gesetzt werden.

Ein Azure Blob Storage bietet verschiedene Speicherstufen, wie "Hot", "Cool", "Cold", oder "Archive".Cool ist zum Beispiel Geeignet für weniger häufige Zugriffe, und hat niedrigere Speicherkosten.

[16] Infos dazu, was Azure Blob Storage ist und anderes stammen von hier: [6]
[17] Alle Informationen zu SAS-Tokens stammen von: [7]

Für uns ist Hot geeignet, da mit dieser Einstellung die Bilder für den häufigen Zugriff mit schnelleren Abrufzeiten optimiert werden. Dies führt allerdings zu höheren Speicherkosten. [18]

CORS-Header

CORS (Cross-Origin Resource Sharing) ist eine Sicherheitsfunktion moderner Webbrowser, die den Zugriff auf Ressourcen zwischen verschiedenen Ursprüngen (Domains) einschränkt. Da Memoryland und Unity Zugriff auf Bilder aus dem Azure Blob Storage benötigen, müssen die folgenden CORS-Einstellungen für "Blob-Services" benötigt.

Es müssen die Spalten "Allowed Origins", "Allowed Headers" und "Exposed Headers" mit dem "*" Zeichen gefüllt werden, und Als "Allowed Methods", sollen "Head" und "GET" angegeben werden. Dies erlaubt den Zugriff von beliebigen Ursprüngen, mit beliebigen Headern in der Anfrage. Weiters wird mit "Exposed Headers" eingestellt, dass alle Antwort-Header für den Client sichtbar sind und die "Allowed Methods" beschränkt die zulässigen HTTP-Methoden auf das Abrufen von Metadaten (HEAD) und das Laden von Bildern (GET). "Max Age: 0", verhindert das Caching der CORS-Vorgaben durch den Browser, sodass jede Anfrage die aktuellen Regeln berücksichtigt.

Wenn alles eingestellt wurde, können nun Bilder aus dem Blob Storage ohne Einschränkungen in Memoryland angezeigt werden. [19]

5.1.2 Einrichtung von Azure PostgreSQL DB

"Azure Database for PostgreSQL" ist ein Datenbankdienst von Microsoft Azure, der auf dem Open-Source-Datenbanksystem PostgreSQL basiert [20]. Er bietet eine Lösung für Anwendungen, die eine relationale Datenbank benötigen.

Die PostgreSQL Datenbank war eine schlüssige Entscheidung aufgrund der Erfahrung der Entwickler. [21]

Flexible Server haben gegenüber Single Server Unterschiede. Der wichtigste Grund für die Entscheidung war, dass Flexible Server auf Linux basiert, während Single Server Windows verwendet. [22]

[18] Alle Informationen zu Blob Access Tiers stammen von [8]
[19] Alle Informationen zu CORS für Azure BLob Storage stammen von [9]
[20] Weiter Infos zu Azure PostgreSQL: [10]
[21] Flexible Server vs. Single Server: [11]
[22] Warum Linux wollten: [12]

Tabelle 1: Vergleich: Flexible Server vs. Single Server

Feature	Single Server	Flexible Server
Betriebssystem	Windows	Linux
Maximale Speichergröße	Bis zu 16 TB	Bis zu 64 TB
Hochverfügbarkeit	Nein	Ja (Zonenredundant)
Kosten	1x	2x (Compute + Storage)

Außerdem besitzen Flexible Servers eine größere maximale Speichergröße, Zonenredundante Hochverfügbarkeit, wodurch Ausfälle besser abgefangen werden können. Obwohl die Kosten etwas höher sein können, sind Flexible Servers aufgrund dieser Punkte die bessere Wahl für unser Projekt.

5.1.3 Einrichtung von Azure AD B2C

Die Einrichtung von Azure AD B2C wurde anhand von Beispiel im folgenden MSAL-Beispiel für Angular durchgeführt [13]. Zusätzlich diente wurde die folgende Website verwendet [14]. [23]

5.1.4 Einrichtung von Azure WebApp

Die Einrichtung der Azure WebApp wurde mithilfe der folgenden Website erledigt [15]. [24]

5.1.5 Einrichtung von Azure Static WebApp

Die Azure WebApp wurde nicht speziell konfiguriert, es gibt aber einige wichtige Aspekte, die vor dem Deployment der Angular Single Page Application beachtet werden müssen.

Damit die Static WebApp die Routen korrekt erkennt und an die Single Page Application weiterleitet, anstatt "*404 Not Found*"-Fehler auszugeben, muss die folgende Konfiguration 1 in der Datei "*staticwebapp.config.json*" hinterlegt werden. Diese Datei muss sich im Verzeichnis des Angular-Projekts befinden – in dieser Diplomarbeit also im Ordner "*memoryland*". [25] [26]

[23] Was ist *Azure AD B2C* und warum verwenden wir es: Kapitel 5.2.5
[24] Was ist *Azure WebApp* und warum verwenden wir es: Kapitel 5.2.5
[25] Infos zu dem Code 1 gibt es in diesem Video: [16]
[26] Was ist *Azure Static WebApp* und warum verwenden wir es: Kapitel 5.2.5

Listing 1: staticwebapp.config.json

```
{
    "navigationFallback": {
        "rewrite": "/",
        "exclude": [
            "*.{png,jpg,jpeg,gif,ico,css,scss,svg,js}"
        ]
    }
}
```

5.1.6 Einrichtung des Backends

Die Einrichtung des Backends für die Diplomarbeit erfordert die Konfiguration der "*appsettings.json*" Datei. In dieser Datei stehen mehrere Einstellungen, damit Memoryland auf die Unterschiedlichen Dienste zugreifen kann. Die Konfiguration umfasst drei Hauptbereiche: Authentifizierung, Datenbank und den Azure Blob Storage.

Authentifizierung

Für die Authentifizierung wird Azure AD B2C genutzt, um die Benutzerverwaltung zu ermöglichen. In der "*appsettings.json*" Datei sind die entsprechenden Einstellungen wie in dem folgendem Beispiel 2 einzutragen. Hier werden unter anderem die Instance, ClientId und Domain des Azure AD B2C Verzeichnisses angegeben. Der SignedOutCallbackPath gibt den Pfad an, der nach der Abmeldung des Benutzers/der Benutzerin aufgerufen wird, und SignUpSignInPolicyId enthält den Namen ebenjener. [27]

Listing 2: appsettings.json

```
{
    "AzureAdB2C": {
        "Instance": "",
        "ClientId": "",
        "Domain": "",
        "SignedOutCallbackPath": "/signout/<SignUpSignInPolicyId>",
        "SignUpSignInPolicyId": ""
    }
}
```

Datenbank

Für die Konfiguration der Datenbank sind die entsprechenden Einstellungen wie in dem folgendem Beispiel 3 ebenso in der Datei "*appsettings.json*" einzutragen. Es gibt sowohl eine Einstellung für den Deployment-Datenbankanschluss (Default) als auch eine für den lokalen Anschluss (DefaultLocal), der für Test-Zwecke verwendet wird.

[27] Alle Informationen zu den Einstellungen der Authentifizierung sind hier zu finden: [17]

Die UseLocalDb Einstellung gibt an, ob eine lokale Datenbank verwendet werden soll, wobei *false* bedeutet, dass die Deployment Datenbank verwendet werden soll. [28]

Listing 3: appsettings.json

```
{
    "UseLocalDb": false,
    "ConnectionStrings": {
        "Default": "",
        "DefaultLocal": ""
    }
}
```

Azure Blob Storage

Für die Konfiguration des Azure Blob Storage sind die entsprechenden Einstellungen wie in dem folgendem Beispiel 4 ebenso in der Datei "*appsettings.json*" einzutragen. Diese Einstellung ist notwendig, um die Anwendung mit dem Azure Blob Storage zu verbinden und den Zugriff auf Bilder und andere Dateien sicherzustellen. [29]

Listing 4: appsettings.json

```
{
    "ConnectionStrings": {
        "BlobStorageDefault": "",
    }
}
```

5.1.7 Einrichtung des Frontends

Auch für das Frontend gibt es wichtige Einstellungen, damit die Single Page Application funktioniert. Hierbei steht die Konfiguration in der Datei "*environment.ts*". Die Konfiguration ist für die Integration der Azure AD B2C Authentifizierung sowie die Verbindung zum Backend und den APIs. Die Datei enthält alle wichtigen Daten für die Authentifizierung und die Verbindung zum Backend.

Im Folgenden ist die Konfiguration beschrieben und in zwei Hauptkapitel unterteilt: *environment.ts*-Konfiguration und Deployment.

environment.ts-Konfiguration

Für die Authentifizierung wird Azure AD B2C verwendet. Die Konfiguration umfasst die folgende Definition 5 von Login- und Profil-Bearbeitungs-Flows. [30]

[28] Alle Informationen zu den Einstellungen der Azure Datenbank sind hier zu finden: [18]
[29] Alle Informationen zu den Einstellungen des Azure Blob Storage sind hier zu finden: [19]
[30] Die Informationen zu den Einstellungen stammen alle aus den folgenden Links: [13] [14]

Userflows in Azure AD B2C sind vordefinierte, anpassbare Prozesse für häufige Benutzeraktionen wie Anmeldung, Registrierung, Profilbearbeitung oder Passwortzurücksetzung. Sie ermöglichen eine schnelle Konfiguration dieser Schritte, sodass Benutzer:innen nach Abschluss Zugriff auf die Anwendung erhalten. [31]

Tabelle 2: Erklärung der Werte von "*environment.ts*"

Wert	**Erklärung**
signUpSignIn	Der Name des Sign-Up/Sign-In Flows
editProfile	Der Name des Profilbearbeitungs-Flows
authority	URL der Autorität, die für Authentifizierung genutzt wird
authorityDomain	Der Domainname des B2C-Tenants
clientId	Die Client-ID der registrierten Anwendung
redirectUri	URI, wohin nach erfolgreicher Authentifizierung weitergeleitet wird
scopes	Liste von Berechtigungen, die die Anwendung benötigt
uri	Die URI des Backend-Servers

Listing 5: environment.ts

```
const b2cPolicies = {
  names: {
    signUpSignIn: "<flow-name>",
    editProfile: "<flow-name>"
  },
  authorities: {
    signUpSignIn: {
      authority: "https://<tenant-name>.b2clogin.com/
        <tenant-name>.onmicrosoft.com/<flow-name>"
    },
    editProfile: {
      authority: "https://<tenant-name>.b2clogin.com/
        <tenant-name>.onmicrosoft.com/<flow-name>"
    }
  },
  authorityDomain: "<tenant-name>.b2clogin.com"
};

export const environment = {
  production: false,
  b2cPolicies: b2cPolicies,
  msalConfig: {
    auth: {
      clientId: "<your-client-id>",
      authority: b2cPolicies.authorities.signUpSignIn.authority,
      knownAuthorities: [b2cPolicies.authorityDomain],
      redirectUri: '/',
    }
  },
  apiConfig: {
    scopes: [
      "https://<tenant-name>.onmicrosoft.com/<app-name>/<scope-name>",
    ],
    uri: "<backend-uri>"
  }
};
```

Diese Konfiguration ermöglicht es, die Anwendung mit den Azure AD B2C Flows zu verbinden und eine sichere Kommunikation mit dem Backend sicherzustellen.

[31] Die Informationen zu Userflows stammen alle aus dem folgenden Link: [20]

Deployment

All diese Einstellungen werden auch in GitHub (https://github.com/) benötigt. Mithilfe von GitHub Actions [32] wird unser Frontend bei einem Push oder einer Pull-Request auf den Branch Main, deployed. Dafür müssen jedoch die folgenden Variablen 3 mittels GitHub Secrets im GitHub Repo eingetragen werden. [33]

Tabelle 3: GitHub Secrets für das Deployment des Frontends

GitHub Secret	**Beschreibung**
`B2C_SIGNUP_SIGNIN_FLOW`	Name des Sign-Up/Sign-In Flows
`B2C_EDIT_PROFILE_FLOW`	Name des Profilbearbeitungs-Flows
`B2C_SIGNUP_SIGNIN_AUTHORITY`	Autoritäts-URL für den Sign-Up/Sign-In Flow
`B2C_EDIT_PROFILE_AUTHORITY`	Autoritäts-URL für den Profilbearbeitungs-Flow
`B2C_AUTHORITY_DOMAIN`	Domain des B2C-Tenants
`CLIENT_ID`	Client-ID der registrierten Anwendung
`SCOPE_READ`	API-Berechtigung für Lesezugriff
`SCOPE_WRITE`	API-Berechtigung für Schreibzugriff
`BACKEND_URI`	URI des Backend-Servers

Tabelle 4: Verwendete GitHub Secrets für die Konfiguration

5.2 Technologien

5.2.1 .NET C#

5.2.2 MSAL

5.2.3 REST

5.2.4 Postgres-DB

5.2.5 Azure

Blob Storage

AD B2C

https://learn.microsoft.com/en-us/azure/active-directory-b2c/overview

https://learn.microsoft.com/en-us/azure/active-directory-b2c/

[32] Alle Informationen zu GitHub stammen von hier: [21]
[33] Alle Informationen zu GitHub Secrets stammen von hier: [22]

WebApp

Static WebApp

5.2.6 Rider

5.2.7 GitHub Actions

5.3 API-Endpoints

5.4 Integration von Azure Blob Storage

5.5 Uploads

- 2 Teile: Postgres und BlogStorage - BlogStorage - Pro User 1 Container - Bilder werden mit ihren Ids gespeichert + nur admin hat zugriff - Zugriff auf bilder erfolgt mit sas-tokens (was sind sas tokens)

5.5.1 Azure Blob-Storage Datenhaltung

5.6 Authentifizierung

6 Frontend-Umsetzung

6.1 Technologien

6.1.1 Angular

6.1.2 WebStorm

6.2 Home

6.3 About

6.4 Explore Worlds

6.5 All Worlds

6.6 Memory Store

7 Unity-Umsetzung

7.1 Technologien

7.1.1 Unity

7.1.2 Visual Studio 2022

7.2 Einrichtung von Unity

7.3 Erstellung von neuen Memoryland-Typen

7.4 Einfügen von Images

8 Zusammenfassung

Glossar

GUID Globally Unique Identifier

Literaturverzeichnis

[1] Microsoft Corporation, „OneDrive," zuletzt besucht: 21. Februar 2025. Online verfügbar: https://www.microsoft.com/de-at/microsoft-365/onedrive/online-cloud-storage

[2] Google Ireland Limited, „Google Photos," zuletzt besucht: 21. Februar 2025. Online verfügbar: https://www.google.com/photos/about/

[3] Animoto, „Animoto," zuletzt besucht: 21. Februar 2025. Online verfügbar: https://animoto.com/

[4] V. Kulkarni und S. Reddy, „Separation of concerns in model-driven development," *IEEE software*, Vol. 20, Nr. 5, S. 64–69, 2003, zuletzt besucht: 24. Februar 2025.

[5] Mozilla Foundation, „File: webkitRelativePath property," zuletzt besucht: 24. Februar 2025. Online verfügbar: https://developer.mozilla.org/en-US/docs/Web/API/File/webkitRelativePath

[6] Microsoft Corporation, „Azure Blob Storage - Training," zuletzt besucht: 23. Februar 2025. Online verfügbar: https://learn.microsoft.com/en-us/training/modules/configure-blob-storage/

[7] ——, „Azure Blob Storage - SAS-Tokens," zuletzt besucht: 23. Februar 2025. Online verfügbar: https://learn.microsoft.com/en-us/azure/storage/common/shared-key-authorization-prevent?tabs=portal

[8] ——, „Azure Blob Storage - Access Tiers," zuletzt besucht: 23. Februar 2025. Online verfügbar: https://learn.microsoft.com/en-us/azure/storage/blobs/access-tiers-overview

[9] ——, „Azure Blob Storage - CORS," zuletzt besucht: 23. Februar 2025. Online verfügbar: https://learn.microsoft.com/en-us/rest/api/storageservices/cross-origin-resource-sharing--cors--support-for-the-azure-storage-services

[10] ——, „Azure PostgreSQL DB," zuletzt besucht: 23. Februar 2025. Online verfügbar: https://learn.microsoft.com/en-us/training/paths/microsoft-learn-azure-database-for-postgresql/

[11] ——, „Azure PostgreSQL DB - Flexible Server vs. Single Server," zuletzt besucht: 23. Februar 2025. Online verfügbar: https://learn.microsoft.com/en-us/azure/postgresql/flexible-server/concepts-compare-single-server-flexible-server

[12] S. Hussain, F. Bahadur, F. Gul, A. Iqbal, G. Ashraf, und S. Nazeer, „Survey on Window and Linux as Server Operating System," *International Journal of Computer*, Vol. 18, Nr. 1, S. 1–6, 2015, zuletzt besucht: 23. Februar 2025.

[13] Microsoft Corporation, „MSAL Example GitHub," zuletzt besucht: 23. Februar 2025. Online verfügbar: https://github.com/AzureAD/microsoft-authentication-library-for-js/tree/dev/samples/msal-angular-samples/angular-b2c-sample

[14] ——, „Angular single-page application using MSAL Angular to sign-in users against Microsoft Entra External ID," zuletzt besucht: 23. Februar 2025. Online verfügbar: https://learn.microsoft.com/en-us/samples/azure-samples/ms-identity-ciam-javascript-tutorial/ms-identity-ciam-javascript-tutorial-2-sign-in-angular/

[15] ——, „Publish an ASP.NET Core app to Azure with Visual Studio Code," zuletzt besucht: 23. Februar 2025. Online verfügbar: https://learn.microsoft.com/en-us/aspnet/core/tutorials/publish-to-azure-webapp-using-vscode?view=aspnetcore-8.0

[16] ——, „How to configure routing in Azure Static Web Apps [6 of 22] | Azure Tips and Tricks: Static Web Apps," zuletzt besucht: 23. Februar 2025. Online verfügbar: https://learn.microsoft.com/en-us/shows/azure-tips-and-tricks-static-web-apps/how-to-configure-routing-in-azure-static-web-apps-6-of-16--azure-tips-and-tricks-static-web-apps

[17] ——, „Enable authentication in your own web API by using Azure AD B2C," zuletzt besucht: 23. Februar 2025. Online verfügbar: https://learn.microsoft.com/en-us/azure/active-directory-b2c/enable-authentication-web-api?tabs=csharpclient

[18] ——, „Connect to an existing Azure PostgreSQL flexible server," zuletzt besucht: 23. Februar 2025. Online verfügbar: https://learn.microsoft.com/en-us/dotnet/aspire/database/azure-postgresql-entity-framework-integration?tabs=dotnet-cli#connect-to-an-existing-azure-postgresql-flexible-server

[19] ——, „View account access keys," zuletzt besucht: 23. Februar 2025. Online verfügbar: https://learn.microsoft.com/en-us/azure/storage/common/storage-account-keys-manage?tabs=azure-portal#view-account-access-keys

[20] ——, „User flows and custom policies overview," zuletzt besucht: 23. Februar 2025. Online verfügbar: https://learn.microsoft.com/en-us/azure/active-directory-b2c/user-flow-overview

[21] GitHub, Inc., „Quickstart for GitHub Actions," zuletzt besucht: 23. Februar 2025. Online verfügbar: https://docs.github.com/en/actions/writing-workflows/quickstart

[22] ——, „Using secrets in GitHub Actions," zuletzt besucht: 23. Februar 2025. Online verfügbar: https://docs.github.com/en/actions/security-for-github-actions/security-guides/using-secrets-in-github-actions

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

Anhang