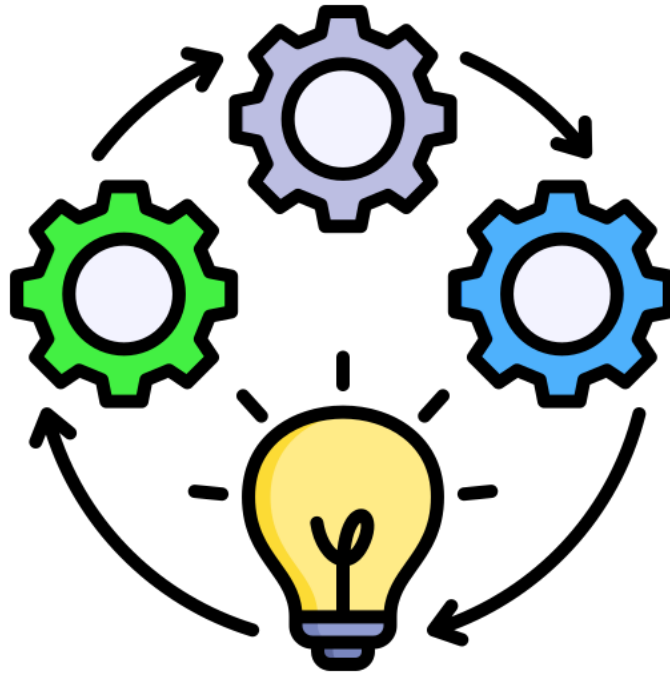


Realisierungskonzept



Verfasser:

Maurice Däppen, Lernender Informatiker
Patrick Aeschlimann, Lernender Informatiker
Timo Schulz, Lernender Informatiker
Mika Hannappel, Lernender Informatiker
Benjamin Raemy, Lernender Informatiker
Lenny Herren, Lernender Informatiker

Lehrperson:

Georg Ninck

Abgabetermin:

7. Dezember 2024

Inhaltsverzeichnis

| | | |
|-------|---|----|
| 1 | Ziele mit der Applikation | 2 |
| 1.1 | Vereinfachung der Bereitstellung von Gameservern | 2 |
| 1.2 | Sicherstellung von Sicherheit und Datenschutz | 2 |
| 1.3 | Optimierung der Benutzererfahrung | 2 |
| 2 | Systemarchitektur | 3 |
| 2.1 | Komponenten der Architektur und ihre Verbindungen | 3 |
| 2.1.1 | Frontend | 3 |
| 2.1.2 | Backend | 3 |
| 2.1.3 | Datenbank | 3 |
| 2.1.4 | Proxy und Netzwerk | 3 |
| 2.1.5 | Gameserver | 3 |
| 2.1.6 | Deployment-Mechanismus | 4 |
| 2.2 | Technologiestack | 4 |
| 2.3 | Deployment | 4 |
| 2.4 | Architektur | 4 |
| 3 | Build | 6 |
| 4 | Abhängigkeiten | 7 |
| 5 | Datenfluss und Datenmodel | 8 |
| 5.1 | Datenmodel | 8 |
| 5.2 | Datenfluss | 8 |
| 6 | Mockups | 10 |
| 7 | Sicherheitskonzept | 12 |
| 7.1 | Authentifizierung und Autorisierung | 12 |
| 7.2 | Datenverschlüsselung | 12 |
| 7.3 | Sicherheitsrichtlinien | 12 |
| 8 | Projektorganisation und Projektressourcen | 14 |
| 9 | Risiken | 15 |
| 10 | Wartung und Support | 16 |
| 10.1 | Wartungsplan | 16 |
| 10.2 | Supportstruktur | 16 |
| 10.3 | Release-Planung | 17 |

1 Ziele mit der Applikation

1.1 Vereinfachung der Bereitstellung von Gameservern

xServer zielt darauf ab, eine intuitive Plattform bereitzustellen, die es Nutzern ermöglicht, Gameserver einfach und effizient einzurichten, ohne dass tiefgehende technische Kenntnisse erforderlich sind. Durch die Automatisierung der Serverbereitstellung und -konfiguration können Nutzer ihre Server in wenigen Minuten individuell anpassen und betreiben, was die Zugänglichkeit für Einzelpersonen und kleine Communities deutlich erhöht.

1.2 Sicherstellung von Sicherheit und Datenschutz

Ein zentrales Ziel von xServer ist die Gewährleistung eines hohen Sicherheitsstandards. Dies umfasst den Schutz sensibler Nutzerdaten durch Massnahmen wie End-to-End-Verschlüsselung, Zugriffskontrollen und regelmässige Sicherheitsüberprüfungen. Gleichzeitig werden bewährte Praktiken der Datensicherung implementiert, um mögliche Risiken wie Datenverlust oder -missbrauch zu minimieren.

1.3 Optimierung der Benutzererfahrung

Die Plattform wird so gestaltet, dass sie eine benutzerfreundliche Oberfläche bietet, die den gesamten Prozess – von der Serverkonfiguration über die Verwaltung bis hin zur Zahlungsabwicklung – reibungslos und intuitiv gestaltet. Durch den Fokus auf Usability sollen sowohl technisch versierte Nutzer als auch Anfänger ein personalisiertes und nahtloses Nutzererlebnis geniessen können.

2 Systemarchitektur

Die Systemarchitektur von xServer ist in mehrere Schichten und Komponenten unterteilt, die effizient miteinander interagieren, um eine zuverlässige Bereitstellung und Verwaltung von Gameservern zu gewährleisten. Diese Architektur ist darauf ausgelegt, flexibel, skalierbar und sicher zu sein.

2.1 Komponenten der Architektur und ihre Verbindungen

2.1.1 Frontend

Entwickelt mit Angular, bildet das Frontend die Benutzerschnittstelle, über die Nutzer Gameserver erstellen, konfigurieren und verwalten können. Es kommuniziert mit dem Backend über eine REST-API und bietet eine intuitive Benutzererfahrung.

Die Komponente läuft als Docker-Image in einem Kubernetes-Cluster und ist über einen Proxy zugänglich.

2.1.2 Backend

Das Backend ist mit Express.js implementiert und fungiert als Vermittler zwischen dem Frontend und der Datenbank. Es verwaltet Nutzeranfragen, führt Validierungen durch und orchestriert die Bereitstellung der Server.

Es verwendet Kubernetes-APIs, um die Gameserver zu steuern, und ist für die Authentifizierung und Autorisierung der Nutzer verantwortlich.

2.1.3 Datenbank

MongoDB wird als Datenbanklösung verwendet, um Nutzerdaten, Konfigurationen und Serverinformationen zu speichern. Sie ist über eine ClusterIP isoliert und nur für das Backend erreichbar.

2.1.4 Proxy und Netzwerk

Der Proxy-Manager, basierend auf Traefik, sorgt für die Verwaltung eingehender Anfragen und die sichere Weiterleitung an die entsprechenden Komponenten. SSL-Zertifikate werden automatisch mit Let's Encrypt erstellt und erneuert.

2.1.5 Gameserver

Die Gameserver laufen in einem separaten Namespace innerhalb des Kubernetes-Clusters. Sie sind voneinander isoliert, um Sicherheit und Stabilität zu gewährleisten. Diese basieren auf Dockerimages.

2.1.6 Deployment-Mechanismus

Docker-Images werden automatisiert gebaut und in ein öffentliches Container-Registry gepusht. Das Deployment erfolgt manuell über Kubernetes-Konfigurationsdateien.

2.2 Technologiestack

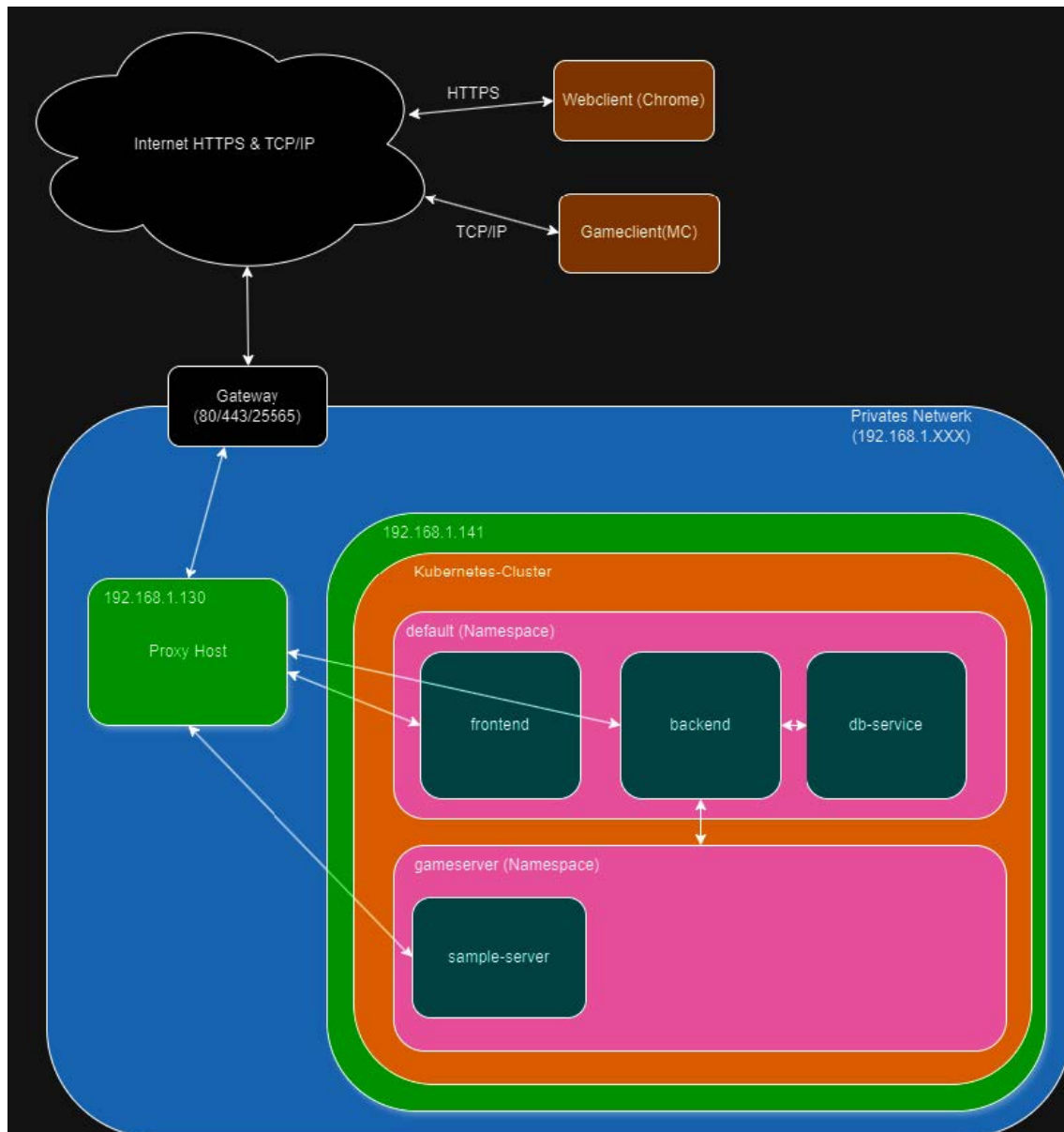
Der Technologiestack von xServer umfasst eine Vielzahl moderner Technologien, die eine robuste, skalierbare und benutzerfreundliche Plattform ermöglichen. Im Frontend kommt Angular zusammen mit Angular Material zum Einsatz, um eine reaktive und optisch ansprechende Benutzeroberfläche zu bieten. Das Backend basiert auf Express.js in Kombination mit Node.js und stellt die Geschäftslogik sowie die Schnittstelle zur Datenbank bereit. Als Datenbank wird MongoDB verwendet, die eine hohe Flexibilität und Skalierbarkeit für die Speicherung von Nutzerdaten, Serverkonfigurationen und weiteren Informationen bietet.

2.3 Deployment

Für das Deployment der Applikation wird eine Kombination aus Docker und Kubernetes genutzt. Insbesondere Kubernetes (in der Variante microk8s) übernimmt die Orchestrierung der Services und gewährleistet eine effiziente Ressourcennutzung sowie Skalierbarkeit. Der Proxy-Dienst Traefik ist für die Verwaltung von Netzwerkanfragen zuständig und sorgt dafür, dass diese sicher an die richtigen Komponenten weitergeleitet werden. Sicherheitsmechanismen wie bcrypt und JWT bieten Authentifizierungs- und Verschlüsselungsfunktionen, während SSL/TLS für die durchgängige Absicherung der Datenübertragung sorgt. Darüber hinaus kommen Terraform und CloudInit als Infrastrukturtools zum Einsatz, um die Bereitstellung und Konfiguration der Server zu automatisieren.

2.4 Architektur

Die Architektur von xServer basiert auf einer klaren Trennung der Aufgaben. Das Frontend ist für die Präsentationslogik und die Interaktion mit den Nutzern verantwortlich. Es kommuniziert mit dem Backend, das die zentrale Geschäftslogik enthält und als Vermittler zwischen Frontend, Datenbank und Gameservern fungiert. Die Datenbank stellt die persistente Datenquelle für alle notwendigen Informationen dar. Kubernetes orchestriert die verschiedenen Services und sorgt dafür, dass diese effizient zusammenarbeiten und bei Bedarf skaliert werden können.



Die Interaktion zwischen den Komponenten erfolgt nahtlos. Nutzeranfragen, die über das Frontend eingegeben werden, werden an das Backend weitergeleitet, wo sie verarbeitet werden. Das Backend greift bei Bedarf auf die Datenbank zu, um Informationen zu speichern oder abzurufen, und übernimmt die Orchestrierung der Gameserver. Gleichzeitig verwaltet der Proxy die Netzwerkanfragen und leitet sie basierend auf der Subdomain an die entsprechenden Instanzen weiter. Schliesslich verbinden sich die Spieleclients direkt über spezifizierte Ports mit den Gameservern, wodurch eine stabile und performante Verbindung gewährleistet wird.

3 Build

Der Bau der Applikation xServer erfolgt in einer strukturierten Pipeline, die auf Automatisierung und Effizienz ausgelegt ist. Der Entwicklungsprozess beginnt mit der Erstellung und Verwaltung des Quellcodes durch die Entwickler, der in einem Versionskontrollsystem wie Git gespeichert wird. Änderungen am Code werden über Pull Requests geprüft und in den Hauptentwicklungszweig integriert, sobald sie die festgelegten Standards und Qualitätskontrollen bestehen.

Sobald ein neuer Code in den Hauptzweig gemergt wird, startet eine automatisierte Build-Pipeline. Diese verwendet Docker, um Images der Applikation zu erstellen. Für jede Komponente der Applikation – wie Frontend, Backend und Datenbank – wird ein separates Docker-Image generiert. Diese Images enthalten alle notwendigen Abhängigkeiten und Konfigurationen, um die Applikation in einer isolierten und konsistenten Umgebung auszuführen.

Die erzeugten Docker-Images werden anschliessend in eine Container-Registry hochgeladen, wo sie für die Bereitstellung im Kubernetes-Cluster bereitstehen. Kubernetes sorgt für die Orchestrierung dieser Container, wobei die Konfigurationsdateien den Umfang der Ressourcen, die Netzwerkeinstellungen und die Interaktionen zwischen den Komponenten definieren. Diese Konfigurationsdateien werden in der Entwicklung und im Deployment eingesetzt, um sicherzustellen, dass die Applikation in verschiedenen Umgebungen – von der lokalen Entwicklung bis hin zur Produktionsumgebung – identisch ausgeführt werden kann.

Für die Bereitstellung in der Produktionsumgebung wird eine spezielle Konfiguration verwendet, die mithilfe von Tools wie Terraform und CloudInit angepasst wird. Diese Tools helfen dabei, die Infrastruktur als Code zu definieren, wodurch Änderungen an der Infrastruktur reproduzierbar und nachvollziehbar sind. Die gesamte Applikation wird schliesslich in einem Kubernetes-Cluster ausgeführt, der die Skalierbarkeit und Verfügbarkeit der Dienste sicherstellt. Mit diesem Ansatz können neue Funktionen oder Updates reibungslos integriert werden, ohne den Betrieb zu unterbrechen, wodurch die Applikation kontinuierlich verbessert wird.

4 Abhängigkeiten

Die Applikation basiert auf mehreren externen Abhängigkeiten, die sowohl für die Funktionalität als auch für die Infrastruktur essenziell sind. Eine der zentralen Abhängigkeiten ist die Nutzung von Docker und Kubernetes. Docker wird verwendet, um die Applikation in Container zu verpacken, während Kubernetes die Orchestrierung dieser Container übernimmt und sicherstellt, dass die Services nahtlos zusammenarbeiten und skalierbar sind.

Eine weitere wichtige Abhängigkeit ist MongoDB, die als Datenbank für die Speicherung von Nutzerdaten, Serverkonfigurationen und anderen persistenten Informationen dient. MongoDB wird über ein offizielles Docker-Image eingebunden und ist zentraler Bestandteil der Datenhaltung.

Für die Proxy-Funktionalität und das Netzwerkmanagement setzt die Applikation auf Traefik. Dieser Dienst ist verantwortlich für die Verarbeitung eingehender Netzwerkanfragen, das Routing zu den entsprechenden Komponenten sowie die Verwaltung und automatische Erneuerung von SSL-Zertifikaten durch die Integration von Let's Encrypt.

Im Bereich Sicherheit nutzt die Applikation Bibliotheken wie bcrypt zur Passwortverschlüsselung und jsonwebtoken (JWT) zur Token-basierten Authentifizierung. Diese gewährleisten, dass Nutzerdaten sicher gespeichert und übertragen werden.

Darüber hinaus sind für die Infrastruktur Automatisierungstools wie Terraform und CloudInit entscheidend. Terraform wird verwendet, um die Infrastruktur als Code zu verwalten, während CloudInit bei der Initialkonfiguration von Servern unterstützt. Beide Tools ermöglichen es, die Bereitstellung von Serverressourcen auf Cloud-Plattformen effizient zu automatisieren.

Schliesslich gibt es Abhängigkeiten von Bibliotheken und Frameworks, die innerhalb der Applikation verwendet werden. Im Frontend kommen Angular und Angular Material für die Benutzeroberfläche zum Einsatz, während das Backend auf Express.js und Node.js basiert. Diese Abhängigkeiten ermöglichen eine schnelle und effiziente Entwicklung der jeweiligen Komponenten und sorgen für eine einheitliche Codebasis.

5 Datenfluss und Datenmodell

5.1 Datenmodell

Das Datenmodell von xServer ist so gestaltet, dass es die effiziente Speicherung und Verarbeitung aller relevanten Informationen ermöglicht. Es basiert auf einer dokumentenorientierten Datenbankstruktur, die durch MongoDB realisiert wird. Die zentralen Entitäten des Datenmodells umfassen Nutzer, Serverkonfigurationen und Transaktionen. Diese werden als JSON-ähnliche Dokumente gespeichert, was eine flexible und skalierbare Handhabung von Daten erlaubt.

Die Entität Nutzer speichert Informationen wie Name, E-Mail-Adresse, Passwort (verschlüsselt mit bcrypt) und Authentifizierungstokens (JWT). Die Entität Serverkonfiguration enthält Details zur Hardwareausstattung der Gameserver (wie CPU, RAM und Speicher), den Spieltyp, spezifische Einstellungen und Verknüpfungen zu den jeweiligen Nutzern. Die Entität Transaktionen dokumentiert Zahlungsdetails, einschliesslich der genutzten Zahlungsmethode und des Transaktionsstatus.

5.2 Datenfluss

Der Datenfluss in xServer ist klar strukturiert und basiert auf einer client-server-architektur. Der Prozess beginnt, wenn ein Nutzer über das Frontend eine Aktion ausführt, wie beispielsweise das Anlegen eines neuen Gameservers. Die Eingaben des Nutzers werden vom Frontend validiert und über die REST-API an das Backend gesendet.

Das Backend verarbeitet die Anfrage, prüft die Authentifizierung und Autorisierung des Nutzers anhand der gespeicherten JWTs und greift anschliessend auf die Datenbank zu. Dabei werden die Daten der Anfrage entweder gespeichert (wie beim Anlegen eines neuen Servers) oder abgerufen (wie bei der Anzeige bestehender Server). Nach der Verarbeitung sendet das Backend die Ergebnisse in Form einer JSON-Antwort zurück an das Frontend, das diese Informationen dann darstellt.

Bei Aktionen, die eine externe Interaktion erfordern, wie die Bereitstellung eines neuen Servers, orchestriert das Backend über die Kubernetes-API den Prozess. Es erstellt basierend auf den Serverkonfigurationen des Nutzers die notwendigen Kubernetes-Ressourcen und sorgt dafür, dass der neue Server innerhalb des Clusters verfügbar ist.

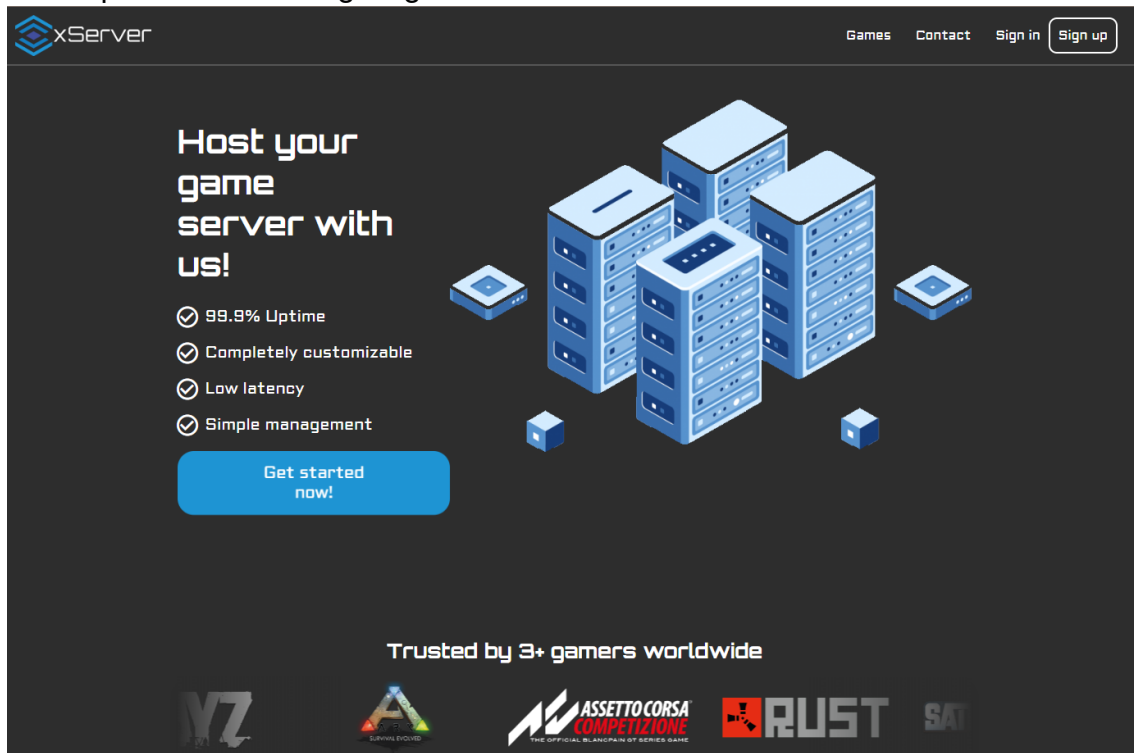
Für sicherheitskritische Daten, wie die Zahlungsabwicklung, wird der Datenfluss durch die Integration von externen Zahlungsanbietern wie Twint ergänzt. Hierbei werden Zahlungsinformationen sicher verschlüsselt übertragen, und das Backend verarbeitet nur die minimal erforderlichen Daten, um Transaktionen abzuschliessen und zu speichern.

Insgesamt gewährleistet der strukturierte Datenfluss eine reibungslose Kommunikation zwischen den Komponenten der Applikation und ermöglicht eine effiziente

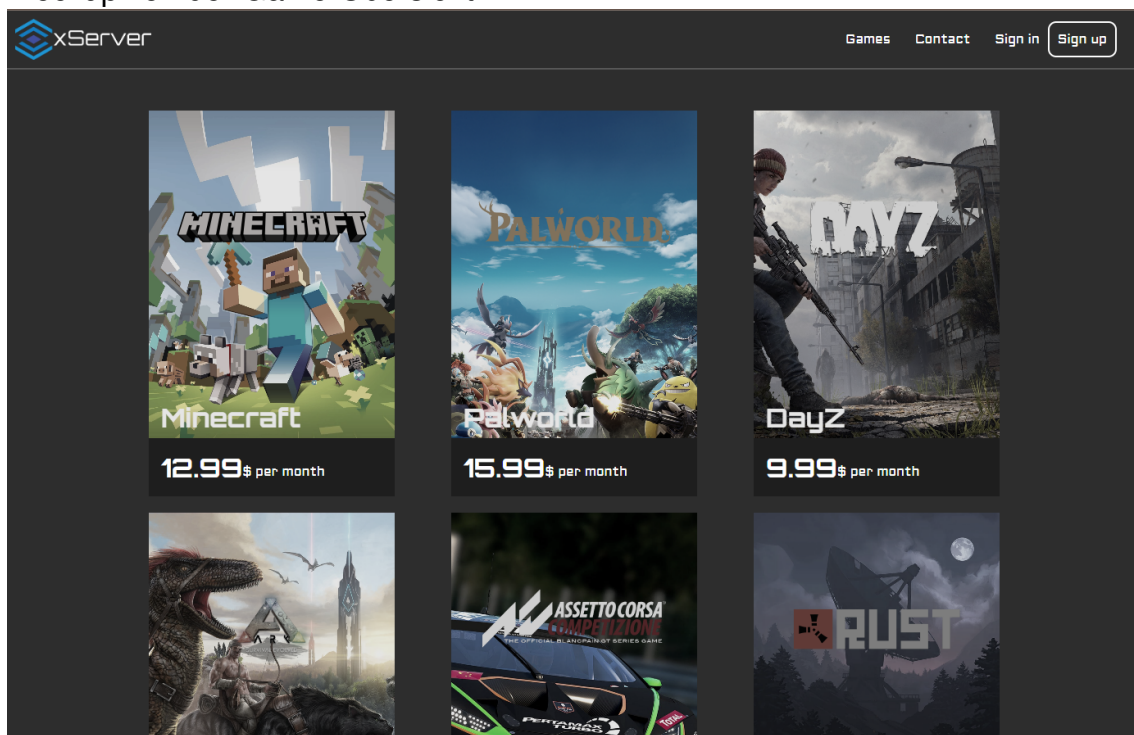
Verarbeitung und Bereitstellung von Daten.

6 Mockups

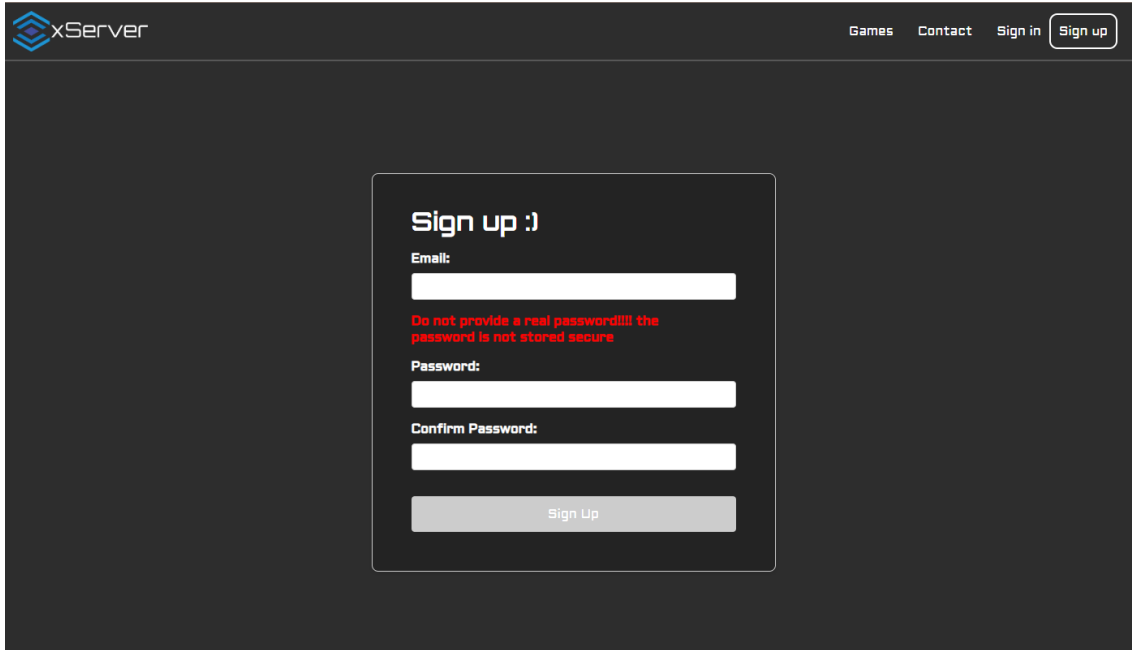
Mockup von der Landing-Page:



Mockup von der Game-Übersicht:

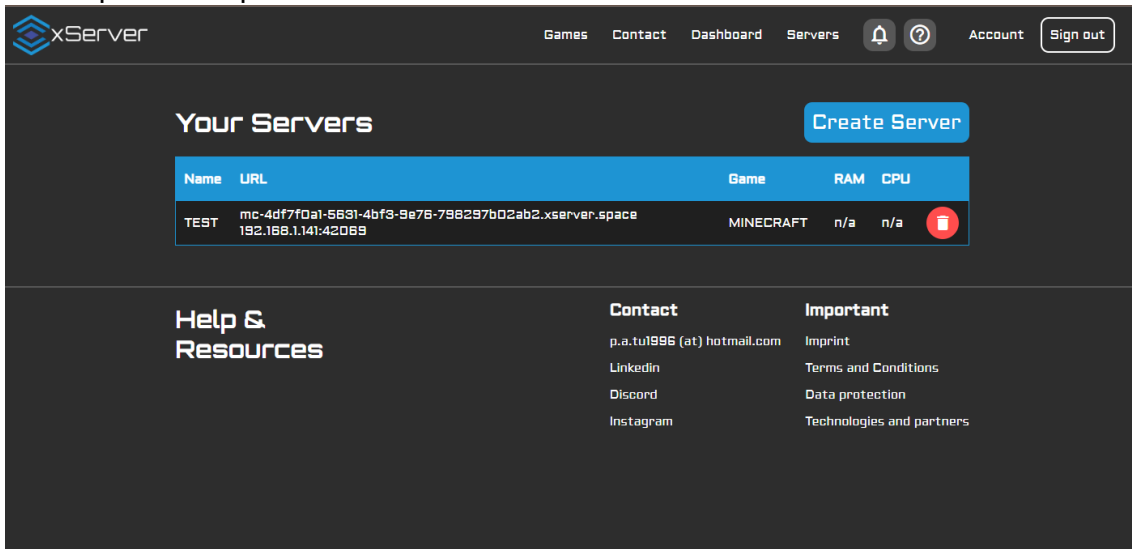


Mockup von der Sign-Up Page (Sign in im gleichen Design):



The image shows a web interface for xServer with a dark theme. At the top, there is a navigation bar with the xServer logo on the left and links for Games, Contact, Sign in, and a Sign up button on the right. The main content area features a central 'Sign up :)' form. This form includes an 'Email:' label and input field, a red warning message 'Do not provide a real password!!!! the password is not stored secure', a 'Password:' label and input field, a 'Confirm Password:' label and input field, and a 'Sign Up' button at the bottom.

Mockup von der persönlichen Serverübersicht:



The image shows a web interface for xServer with a dark theme, displaying a personal server overview. The top navigation bar includes the xServer logo, links for Games, Contact, Dashboard, Servers, a notification bell, a help icon, an Account link, and a Sign out button. The main content area is titled 'Your Servers' and features a 'Create Server' button. Below the title is a table with the following data:

| Name | URL | Game | RAM | CPU |
|------|--|-----------|-----|-----|
| TEST | mc-4df7f0a1-5631-4bf3-9e76-798297b02ab2.xserver.space 192.168.1.141:42069 | MINECRAFT | n/a | n/a |

Below the table, there are three columns of links: 'Help & Resources', 'Contact' (with links to p.a.tu1996 (at) hotmail.com, LinkedIn, Discord, and Instagram), and 'Important' (with links to Imprint, Terms and Conditions, Data protection, and Technologies and partners).

7 Sicherheitskonzept

Das Sicherheitskonzept von xServer ist auf den Schutz der Daten und Systeme vor unbefugtem Zugriff, Verlust und Missbrauch ausgerichtet. Es basiert auf modernen Sicherheitsstandards und deckt die Bereiche Authentifizierung und Autorisierung, Datenverschlüsselung sowie proaktive Sicherheitsrichtlinien ab.

7.1 Authentifizierung und Autorisierung

Die Authentifizierung erfolgt durch die Verwendung von JSON Web Tokens (JWT). Bei der Anmeldung eines Nutzers generiert das Backend ein signiertes JWT, das die Identität des Nutzers bestätigt und dessen Berechtigungen beinhaltet. Dieses Token wird an das Frontend gesendet und bei jeder Anfrage an das Backend mitgeschickt, um die Authentifizierung sicherzustellen. Durch die Überprüfung der Signatur des Tokens garantiert das System, dass es nur von einer autorisierten Quelle stammt.

Die Autorisierung wird ebenfalls durch JWTs umgesetzt. Diese enthalten rollenbasierte Informationen, die festlegen, welche Aktionen ein Nutzer ausführen darf. So wird sichergestellt, dass nur berechtigte Nutzer Zugriff auf sensible Daten oder administrative Funktionen erhalten.

7.2 Datenverschlüsselung

Datenverschlüsselung wird sowohl für Daten in Ruhe als auch für Daten während der Übertragung angewendet. In der Datenbank werden alle sensiblen Informationen, wie Passwörter, mithilfe des bcrypt-Algorithmus verschlüsselt gespeichert. Dieser Ansatz gewährleistet, dass selbst im Falle eines Datenlecks keine Klartext-Passwörter kompromittiert werden können.

Für Datenübertragungen wird SSL/TLS verwendet, um eine sichere Verbindung zwischen den Komponenten der Applikation zu gewährleisten. Alle API-Aufrufe, sowohl zwischen Frontend und Backend als auch bei externen Diensten wie Zahlungsanbietern, sind durch HTTPS abgesichert. Zusätzlich schützt Traefik als Proxy die Kommunikation, indem es automatisch SSL-Zertifikate über Let's Encrypt verwaltet und erneuert.

7.3 Sicherheitsrichtlinien

Das Sicherheitskonzept von xServer beinhaltet proaktive Massnahmen zur Bedrohungsabwehr, **Schwachstellenmanagement und Notfallplanung**: Proaktive Sicherheitsmassnahmen: Regelmässige Updates der genutzten Frameworks und Bibliotheken minimieren Risiken durch bekannte Schwachstellen. Automatisierte Sicherheits-Scans prüfen den Code auf potenzielle Sicherheitslücken. Darüber hinaus werden Firewalls und ein restriktives Rollen- und Berechtigungssystem verwendet, um die Angriffsfläche zu reduzieren.

Schwachstellenmanagement: Alle Systemkomponenten werden regelmässig auf Schwachstellen untersucht. Falls eine Sicherheitslücke entdeckt wird, folgt eine sofortige Analyse und ein schneller Patch, um potenzielle Schäden zu minimieren.

Notfallplanung: Es existiert ein Backup- und Wiederherstellungskonzept, das regelmässige Sicherungen aller kritischen Daten vorsieht. Diese Backups werden sicher gespeichert und getestet, um im Ernstfall eine schnelle Wiederherstellung zu ermöglichen. Darüber hinaus sorgt eine durchdachte Logging- und Monitoring-Strategie dafür, dass Angriffe frühzeitig erkannt und dokumentiert werden, um geeignete Gegenmassnahmen einzuleiten.

8 Projektorganisation und Projektressourcen

Die Projektorganisation von xServer ist klar strukturiert und basiert auf einer präzisen Rollenverteilung, um einen reibungslosen Ablauf zu gewährleisten. Der Projektleiter, Maurice Däppen, ist für die Koordination des gesamten Projekts verantwortlich. Seine Aufgaben umfassen die Überwachung des Fortschritts, die Sicherstellung der Qualität und die Kommunikation mit dem Auftraggeber Valve (Steam). Das Entwicklerteam, bestehend aus Patrick Aeschlimann, Lenny Herren und Mika Hannappel, trägt die Verantwortung für die technische Umsetzung der Plattform. Sie arbeiten an der Entwicklung des Frontends, Backends sowie der Integration aller systemrelevanten Komponenten. Als Auftraggeber definiert Valve die Projektziele und Anforderungen und prüft die Ergebnisse der verschiedenen Projektphasen. Ergänzt wird die Organisation durch Stakeholder wie potenzielle Nutzer und Testpersonen, die wertvolles Feedback geben und zur Optimierung der Plattform beitragen.

Das Team arbeitet eng zusammen und nutzt agile Methoden, um flexibel auf Herausforderungen reagieren zu können. Regelmässige Besprechungen und klare Kommunikationswege, vor allem über Tools wie Microsoft Teams, sorgen dafür, dass alle Mitglieder stets über den aktuellen Stand informiert sind. Dadurch bleibt das Projekt auch bei komplexeren Aufgaben gut steuerbar und transparent.

Für die Umsetzung des Projekts stehen spezifische technische und personelle Ressourcen zur Verfügung, die die Grundlage für die erfolgreiche Entwicklung bilden. Jeder Entwickler nutzt einen Laptop mit einer standardisierten Entwicklungsumgebung, die alle notwendigen Tools wie Visual Studio Code, Docker und Git umfasst. Die technische Infrastruktur basiert auf Kubernetes (microk8s) für die Orchestrierung von Containern sowie auf AWS-Diensten, die für das Hosting und die Skalierung der Plattform verwendet werden. MongoDB dient als zentrale Datenbank und wird in der Cloud bereitgestellt, um eine skalierbare und flexible Datenverwaltung zu ermöglichen.

Die Zeitressourcen des Teams sind eng getaktet, mit regelmässigen Entwicklungs- und Besprechungsphasen während der Woche. Um den Fortschritt zu beschleunigen, wird zudem die Freizeit der Mitglieder genutzt, wobei eine effiziente Planung sicherstellt, dass die verfügbaren Stunden optimal ausgeschöpft werden. Die finanziellen Mittel sind begrenzt, decken jedoch die wesentlichen Ausgaben für Infrastruktur, Softwarelizenzen und andere benötigte Ressourcen. Durch den Einsatz von Open-Source-Technologien und Automatisierungswerkzeugen wird der kosteneffiziente Einsatz der Mittel gewährleistet.

9 Risiken

Ein bedeutendes Risiko besteht in technologischen Herausforderungen, insbesondere bei der Integration verschiedener Technologien wie Kubernetes, Docker und MongoDB. Die Komplexität der Infrastruktur, kombiniert mit den spezifischen Anforderungen an die Skalierbarkeit und Sicherheit der Plattform, könnte zu Verzögerungen oder unerwarteten technischen Problemen führen. Auch die Verwendung externer Abhängigkeiten wie Traefik oder CloudInit birgt das Risiko, dass neue Versionen oder Konfigurationsänderungen unvorhergesehene Fehler oder Inkompatibilitäten mit sich bringen.

Ein weiteres potenzielles Risiko ist der Ressourcenmangel. Dies betrifft sowohl die zeitlichen als auch die personellen Ressourcen. Da das Team mit begrenzten Kapazitäten arbeitet, könnte es bei unvorhergesehenen Aufgaben oder Problemen zu Engpässen kommen, die den Projektzeitplan beeinträchtigen. Die finanzielle Begrenzung des Projekts erhöht zudem die Abhängigkeit von Open-Source-Software, die wiederum weniger Support und längere Lösungszeiten bei Problemen bedeuten kann.

Ein zusätzliches Risiko stellt der Schutz sensibler Nutzerdaten dar. Angesichts der Vielzahl an Cyberbedrohungen ist es unerlässlich, dass Sicherheitsmassnahmen wie Datenverschlüsselung, Zugriffskontrollen und regelmässige Schwachstellenanalysen konsequent umgesetzt werden. Jede Sicherheitslücke könnte nicht nur das Vertrauen der Nutzer beeinträchtigen, sondern auch rechtliche Konsequenzen nach sich ziehen.

Abschliessend ist auch die Nutzerakzeptanz ein potenzielles Risiko. Trotz einer benutzerfreundlichen Gestaltung des Frontends kann es passieren, dass bestimmte Zielgruppen Schwierigkeiten bei der Nutzung der Plattform haben oder dass ihre Erwartungen nicht vollständig erfüllt werden. Dieses Risiko kann durch umfassende Tests und Feedbackschleifen während der Entwicklungsphasen reduziert werden.

10 Wartung und Support

Das Wartungs- und Supportkonzept für die Applikation xServer stellt sicher, dass die Plattform nach dem Go-Live kontinuierlich gepflegt, verbessert und bei Bedarf unterstützt wird. Es gewährleistet, dass sowohl technische Anforderungen als auch Nutzerbedürfnisse langfristig erfüllt werden.

10.1 Wartungsplan

Nach dem Go-Live wird ein strukturierter Wartungsplan umgesetzt, um die Funktionsfähigkeit und Sicherheit der Applikation zu gewährleisten. Regelmässige Updates sind ein zentraler Bestandteil der Wartung. Dazu gehören sowohl Sicherheitsupdates für genutzte Bibliotheken und Frameworks als auch Funktionserweiterungen, um den steigenden Anforderungen der Nutzer gerecht zu werden. Weiterhin wird der Code in regelmässigen Abständen überprüft und refaktoriert, um technische Schulden zu minimieren und die langfristige Wartbarkeit der Applikation zu sichern.

Monitoring-Tools werden eingesetzt, um die Performance der Applikation zu überwachen und potenzielle Probleme frühzeitig zu erkennen. Basierend auf diesen Erkenntnissen werden proaktive Massnahmen ergriffen, beispielsweise zur Optimierung der Systemressourcen oder zur Beseitigung von Engpässen. Zudem sind geplante Wartungsfenster vorgesehen, in denen grössere Systemaktualisierungen oder Infrastrukturänderungen durchgeführt werden können. Diese werden vorab kommuniziert, um die Auswirkungen auf die Nutzer zu minimieren.

10.2 Supportstruktur

Für den laufenden Betrieb von xServer wird eine mehrstufige Supportstruktur eingerichtet. Im ersten Schritt können Nutzer über eine E-Mail-Support-Adresse ihre Anfragen oder Probleme melden. Dieser Basis-Support wird von den Entwicklern selbst angeboten, die die Plattform detailliert kennen und daher effizient auf technische Anfragen reagieren können.

Für dringende Fälle, die eine sofortige Lösung erfordern, wird ein On-Call-Support bereitgestellt, der auch ausserhalb der regulären Arbeitszeiten verfügbar ist. Der On-Call-Support umfasst insbesondere kritische Bereiche wie die Verfügbarkeit von Gameservern und die Sicherheit der Plattform. Service-Level-Agreements (SLAs) definieren klare Reaktions- und Lösungszeiten, um die Zufriedenheit der Nutzer sicherzustellen.

Langfristig ist geplant, ein dediziertes Supportteam aufzubauen, das die Anfragen professionell bearbeitet und umfangreichere Unterstützung leisten kann. Für Standardfragen wird eine Wissensdatenbank bereitgestellt, die häufige Probleme und deren Lösungen beschreibt.

10.3 Release-Planung

Die Releases der Applikation erfolgen in einem iterativen Prozess, der auf einem kontinuierlichen Integrations- und Bereitstellungsansatz (CI/CD) basiert. Kleinere Updates und Fehlerbehebungen werden in regelmässigen Intervallen bereitgestellt, um die Plattform stabil und aktuell zu halten. Diese sogenannten Minor Releases erfolgen alle zwei bis vier Wochen und umfassen in der Regel Bugfixes sowie kleine Verbesserungen.

Grössere Funktionserweiterungen und grundlegende Systemänderungen werden in Major Releases zusammengefasst, die quartalsweise geplant sind. Vor jedem Release durchläuft die Applikation eine umfassende Testphase, die sowohl automatische Tests als auch manuelle Qualitätsprüfungen umfasst. Die Bereitstellung erfolgt während vorab angekündigter Wartungsfenster, um Störungen für die Nutzer zu minimieren.