LinkUp – Soziales Netzwerk

PROJEKT WEB-PROGRAMMIERUNG

Lina Groth, Klara Kulinna, Jennifer Tielke, Fabian Fehringer, Holger Theis, Marc Buddemeier
DUALE HOCHSCHUE BADEN-WÜRTTEMBERG MANNHEIM

Inhaltsverzeichnis

Einleitung

Das Ziel unseres Programmierprojektes war es, gemeinsam ein soziales Netzwerk zu entwickeln. In einer Zeit, in der digitale Kommunikation einen immer größeren Stellenwert einnimmt, wollten wir eine Plattform schaffen, die den Austausch von Informationen, Ideen und Meinungen ermöglicht und gleichzeitig eine unterhaltsame und ansprechende Erfahrung bietet.

Links:
Website: https://link-up-rho.vercel.app
Frontend Repository: https://github.com/linagrxth/LinkUpFront
Backend Repository: https://github.com/marcbudd/linkup-service
Swagger: https://linkup-api.de/swagger

Anforderungen

1. Benutzerregistrierung und sichere Authentifizierung:
    - Benutzer sollen sich mit ihrer E-Mail und einem Passwort registrieren können.
    - Die E-Mail-Adresse wird mit einem Token, der per Mail versandt wird, verifiziert.
2. Benutzerprofile und Landingpages:
    - Jeder Benutzer soll ein individuelles Profil haben, das Informationen wie Name, Benutzername, Geburtstag und eine kurze Biografie enthält.
    - Eine eigene Landingpage für jeden Benutzer, auf der seine Beiträge und Informationen angezeigt werden.
3. Timeline und Beiträge:
    - Eine Timeline, die alle Beiträge der Benutzer in chronologischer Reihenfolge anzeigt.
    - Benutzer sollen in der Lage sein, eigene Beiträge zu erstellen, die dann sowohl auf ihrer eigenen Seite als auch in der Timeline anderer Benutzer angezeigt werden.
    - Benutzer sollen sich gegenseitig folgen können, um die anderen Beiträge zu sehen sowie Beiträge liken und kommentieren können.
4. Nachrichtenfunktion:
    - Benutzer sollen in der Lage sein, sich gegenseitig private Nachrichten zu senden.
    - Die Nachrichten sollten in einem Postfach oder einer Inbox gespeichert und chronologisch sortiert werden.
5. Datenbank:
    - Eine Datenbank wird benötigt, um Benutzerkonten, Beiträge und Nachrichten zu speichern.
    - Der Datenbankzugriff erfolgt nur authentifiziert über die Backend-API, was die Datensicherheit gewährleistet.

Diese Anforderungen bildeten die Grundlage für die Entwicklung unseres sozialen Netzwerks. Außerdem sollte das Netzwerk offen für Erweiterungen bleiben, sodass die Funktionen auch in Zukunft noch ausgeweitet werden können.

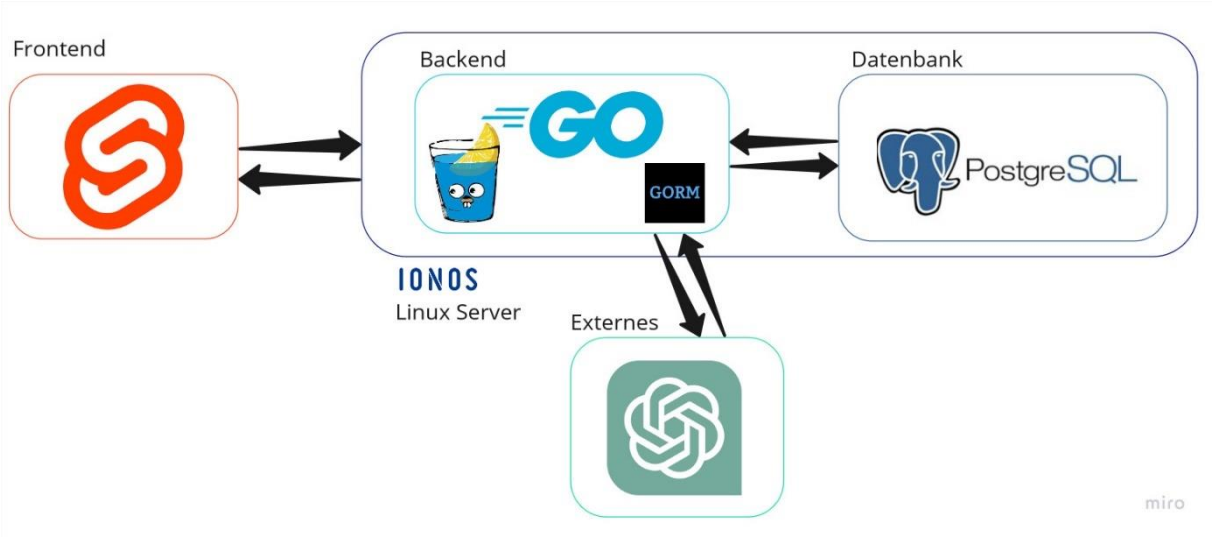

Abbildung 1: Technologien

Backend

Datenbank

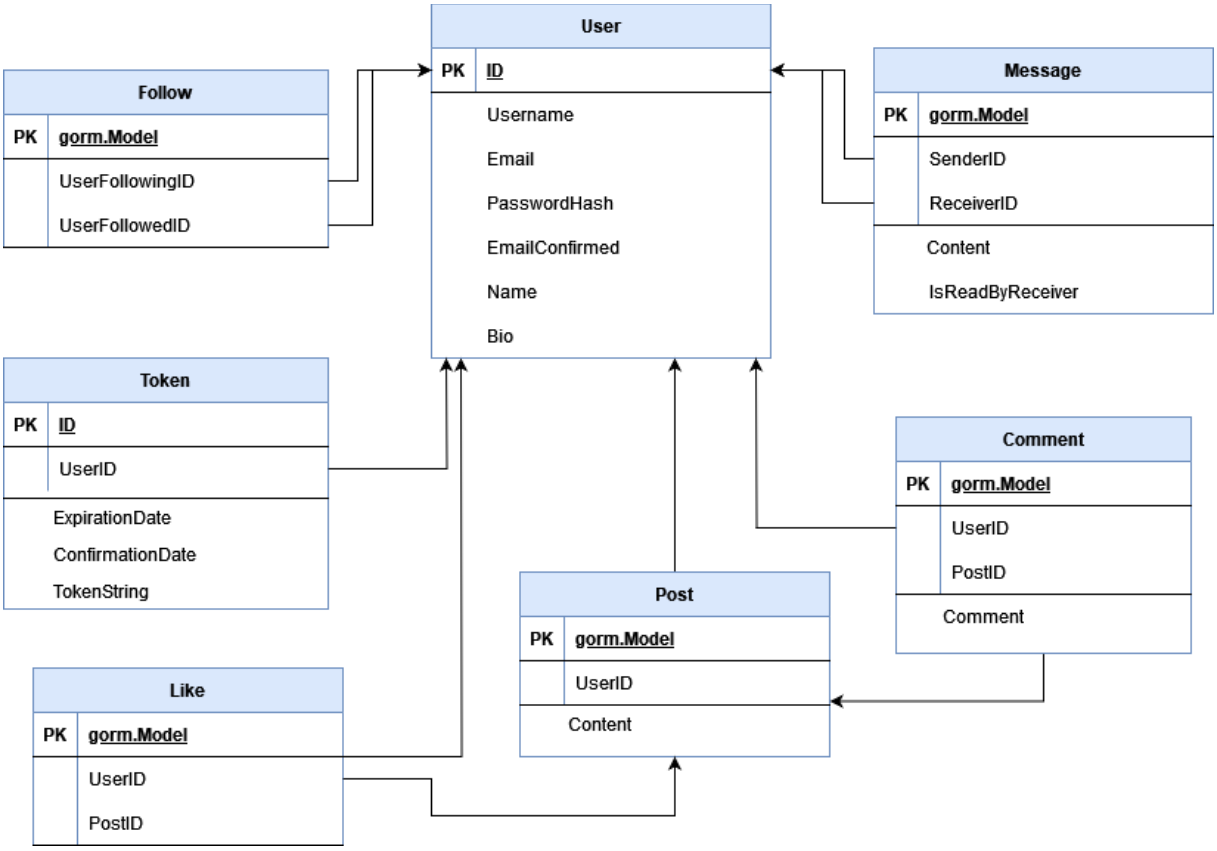

Abbildung 2: ER-Modell

Im Mittelpunkt des Datenmodells steht die Entität "User". Hier werden sämtliche Informationen zu einem Benutzer gespeichert, wie beispielsweise die E-Mail-Adresse, das gehashte Passwort und weitere relevante Informationen.

Darüber hinaus spielen die Entitäten "Post" und "Comment" eine wichtige Rolle. Diese ermöglichen die Speicherung von individuellen Beiträgen sowie den dazugehörigen Kommentaren, wobei jedem Beitrag und Kommentar ein Benutzer zugeordnet wird.

Die Entitäten "Follow" und "Like" dienen der Verfolgung der Benutzerinteraktionen. "Follow" ermöglicht es einem Benutzer, einem anderen Benutzer zu folgen und dessen Aktivitäten zu verfolgen. "Like" wiederum speichert die Informationen darüber, welcher Benutzer welchen Beitrag oder welches Bild mit einem "Like" markiert hat.

Die Kommunikation zwischen den Benutzern wird in der Entität "Message" gespeichert. Hierbei werden die Nachrichten zwischen den Benutzern archiviert und können in chronologischer Reihenfolge abgerufen werden.

Um die E-Mail-Adresse eines Benutzers zu verifizieren, nutzen wir die Entität "Token". Dieser Token ermöglicht es dem Benutzer, seine E-Mail-Adresse zu bestätigen. Der Token wird nach der Registrierung erstellt und per Mail versandt. Schickt der Benutzer diesen Token zurück, hat er sich erfolgreich verifiziert.

In unserem Datenmodell verwenden wir teilweise das "gorm.Model" als Basisstruktur für unsere Entitäten. "gorm.Model" ist ein Hilfsmittel, das vom GORM (Go Object Relational Mapping)-Framework, ein ORM-Framework für Go-Programme, bereitgestellt wird. Das "gorm.Model" bietet eine Reihe von Feldern, die bei der Arbeit mit der Datenbank helfen. Es enthält unter anderem ein automatisch generiertes Primärschlüsselfeld namens "ID", das als eindeutige Kennung für jede Instanz einer Entität dient. Darüber hinaus beinhaltet es auch Felder wie "CreatedAt" und "UpdatedAt", die automatisch den Zeitpunkt der Erstellung bzw. des letzten Updates einer Entität verfolgen. Dies erleichterte die Entwicklung, da grundlegende Attribute bereits vorhanden sind.

Wir haben uns für eine PostgreSQL-Datenbank entschieden, da dies eine weit verbreitete Datenbank mit vielen Nutzern ist. Die Datenbank lässt sich unkompliziert auf einem Server installieren und verwalten. Außerdem gilt PostgreSQL als sehr leistungsstarkes Datenbankmanagementsystem mit vielseitigen Funktionen und einer hohen Zuverlässigkeit.

Golang API

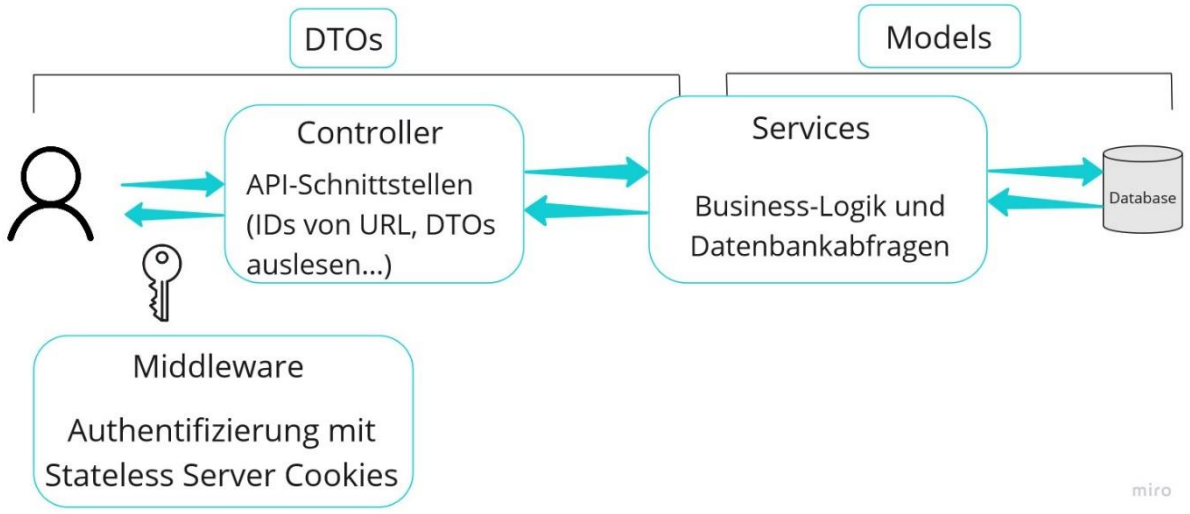

Abbildung 3: Aufbau Golang API

Wir haben uns entschieden, das Frontend und Backend in unserem Projekt strikt zu trennen, um die Verantwortlichkeiten der einzelnen Einheiten klar zu definieren. Durch diese Trennung können wir das

Frontend auf die Darstellung und Interaktion mit den Benutzern konzentrieren, während das Backend für die Datenverarbeitung, Datenbankzugriff und Geschäftslogik verantwortlich ist. Diese klare Aufteilung erleichtert die Wartung, Skalierbarkeit und Weiterentwicklung unserer Anwendung.

Wir haben uns entschieden, das Frontend und Backend in unserem Projekt zu trennen. Durch die Trennung können wir uns bei der Entwicklung des Frontend auf die Darstellung und Interaktion mit den Benutzern konzentrieren, während das Backend für die Datenverarbeitung, Datenbankzugriff und Geschäftslogik verantwortlich ist.

So hatten wir außerdem die Möglichkeit mehrere Programmiersprachen und Frameworks genauer kennenlernen. Noch dazu ist die Golang API wiederverwendbar, wenn eine mobile App für das soziale Netzwerk in Zukunft programmiert wird.

Die Entscheidung, Golang als Programmiersprache für unser Projekt zu verwenden, basiert auf mehreren Gründen. Golang ist bekannt für seine hohe Leistung, Effizienz und einfache Skalierbarkeit. Es bietet eine starke Unterstützung für gleichzeitige Abläufe und ermöglicht die effektive Nutzung von Multi-Core-Systemen. Golang hat auch eine klare und einfache Syntax, die die Lesbarkeit des Codes verbessert und die Entwicklung erleichtert.

Für die Entwicklung des Backend-Teils haben wir uns für das Gin-Framework entschieden. Gin ist ein leichtgewichtiges und schnelles Framework für die Entwicklung von Webanwendungen in Golang. Es bietet eine einfache und Entwicklung von Routenverwaltung, Anfragenverarbeitung und Middleware-Implementierung.

Für den Datenbankzugriff haben wir uns für GORM entschieden. GORM ist ein leistungsstarkes ORM-Framework für Golang, das uns dabei hilft, die Datenbankoperationen zu abstrahieren und direkt in Golang zu schreiben. Es bietet eine einfache Syntax für die Erstellung von Abfragen, Unterstützung für Beziehungen zwischen Entitäten und die Integration mit verschiedenen Datenbanken.

Die Authentifizierung erfolgt mittels Stateless Server Cookies. Das hat einerseits den Vorteil, dass wir keine Authentifizierungsinformationen in der Datenbank speichern wollen. Außerdem haben wir die Möglichkeit Cookies vom Backend aus zu setzen und im Frontend keine Authentifizierungsmechanismen mehr implementieren müssen.

Bereitgestellt wird die das Backend über eine Linux-VM von IONOS, auf der die Datenbank läuft und die Golang-Anwendung in einem Docker-Container. Die Linux-VM von IONOS bietet eine sehr günstige Gelegenheit, einen Server frei nach unseren Bedürfnissen zu konfigurieren.

Frontend

Bei der Entwicklung des Frontends haben wir uns insgesamt an bestehenden sozialen Netzwerken orientiert. Wir haben uns für die Verwendung des Svelte-Frameworks entschieden, da wir während unserer Vorlesungen damit gearbeitet haben und so die Möglichkeit nutzen konnten, unsere Kenntnisse zu vertiefen.

Für die Bereitstellung des Frontends haben wir uns für Vercel entschieden. Der Deploy-Prozess auf Vercel war schnell und unkompliziert, mit nur wenigen Klicks konnten wir unsere Anwendung bereitstellen. Ein weiterer Vorteil von Vercel ist, dass es eine kostenlose Option gibt, die uns die Möglichkeit gab, das Frontend ohne zusätzliche Kosten zu hosten.

„Lessons Learned“ & Fazit

Im Vergleich zu vorherigen Semestern konnten wir feststellen, dass wir bei der Durchführung von Programmierprojekten bereits deutlich strukturierter vorgehen können. Wir haben mittlerweile mehr

Erfahrung im Umgang mit IDEs, GitHub und anderen Tools gesammelt, was uns dabei geholfen hat, effizienter zu arbeiten.

Besonders spannend war die Möglichkeit, in diesem Projekt neue Programmiersprachen kennenzulernen, mit denen wir bisher noch nicht in Berührung gekommen waren. Dies hat unseren Horizont erweitert und uns gezeigt, wie vielfältig die Welt der Softwareentwicklung ist.

Ein wichtiger Aspekt, den wir während des Projekts erkannt haben, ist die Bedeutung einer kontinuierlichen Kommunikation. Es war entscheidend, dass das Backend-Team und das Frontend-Team ständig im Austausch standen, um die erforderlichen Schnittstellen und deren Funktionalitäten gemeinsam zu definieren. Durch diese enge Zusammenarbeit konnten Missverständnisse vermieden und die Effizienz des Projekts gesteigert werden.

Für zukünftige Projekte nehmen wir uns vor, von Anfang an noch stärker im Austausch zu stehen. Es wäre möglicherweise auch hilfreich, wenn jedes Teammitglied sowohl am Frontend als auch am Backend mitgearbeitet hätte. Dies hätte nicht nur dazu beigetragen, Missverständnisse zu minimieren, sondern auch zu einer umfassenderen Erfahrung geführt, da alle Teammitglieder verschiedene Technologien kennenlernen und ihr Wissen erweitern könnten.

Insgesamt sind wir mit den Fortschritten und dem Lerneffekt, den wir während dieses Projekts erzielt haben, sehr zufrieden. Wir freuen uns darauf, dieses Wissen in zukünftigen Projekten anzuwenden und unsere Fähigkeiten weiterzuentwickeln.

Arbeitsaufteilung

Wir haben die Aufgaben wie folgt aufgeteilt:

- Frontend: Lina Groth (9220827), Klara Kulinna (4379580), Holger Theis (4585051)
- Backend:
    - Datenbank, Controller-Logik, Authentifizierung: Marc Buddemeier (8466584)
    - Service-Logik: Jennifer Tielke (6181218)