Gruppe 3:

Andreas Lengauer (K1455988)

Lukas Mindlberger (K1255809)

Robert Ördög (K1355928)

Christian Reiter (K1355821)

Dokumentation

Blog-Service: Übung 3

Inhaltsverzeichnis

[1 Aufgabenbeschreibung 3](#_Toc471677266)

[2 Blog-Service API 3](#_Toc471677267)

[2.1 JSON Objekte 3](#_Toc471677268)

[2.2 Schnittstellenbeschreibung 5](#_Toc471677269)

[3 Entwicklung mobiler Client 5](#_Toc471677270)

[3.1 Beschreibung der Oberfläche 6](#_Toc471677271)

[4 Diskussion Rest Konzept 9](#_Toc471677272)

# Aufgabenbeschreibung

Im Rahmen der Übung 3 besteht die Aufgabe darin, einen alternativen „nativen Client“ für den bestehenden Blog-Service zu entwickeln. Dieser mobile Client kann auf einer Mobilplattform nach Wahl basieren (Android, iOS, WindowsPhone). Dazu soll eine gemeinsame API für den Web- und mobilen Client verwendet werden und die bestehende API, wenn nötig, angepasst und erweitert werden.

Dabei soll eine Dokumentation der API erstellt und zur Verfügung gestellt werden.

Zusätzlich soll der REST-Architekturstil anhand des umgesetzten Services diskutiert werden.

Welche zentralen REST-Konzepte wurden umgesetzt, welche nicht?

Wie wurden sie umgesetzt?

Wo wurden ggfs. REST-Konzepte verletzt?

# Blog-Service API

Der vorhandene Webservice musste nicht angepasst werden, da er ohne Probleme mit dem mobilen Client kommunizierte.

Der Blog-Service wurde mit einer REST-Schnittstelle implementiert, wobei der Datenaustausch mit JSON stattfindet.

## JSON Objekte

Folgenden JSON Objekte werden in der Schnittstelle verwendet:

**User.json**

{

"Name": "xx",

"Password": "xx",

"EMail": "xx"

}

**Users.json**

[

{

"Name": "xx",

"Password": "xx",

"EMail": "xx"

}

]

**Group.json**

{

"Name": "xx",

"Participants": [

{

"Name": "xx"

}

]

}

**BlogEntity.json**

{

"Title": "xx",

"Text": "xx",

"Author": "xx",

"Date": "xx",

"GroupID": "xx"

}

**BlogEntities.json**

[

{

"ID": "xx",

"Title": "xx",

"Text": "xx",

"Author": "xx",

"Date": "xx",

"GroupID": "xx",

"Comments": [

{

"Text": "xx",

"Author": "xx",

"Date": "xx"

}

]

}

]

**Comment.json**

{

"BlogEntityID": "xx",

"Text": "xx",

"Author": "xx",

"Date": "xx"

}

## Schnittstellenbeschreibung

Nachfolgend befindet sich eine Beschreibung dieser Schnittstelle:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Pfad | Methode | Statuscode | Parameter | Beschreibung |
| /addUser | POST | 200=OK 500=InternalServerError | Eingabe:User.json | Anlegen eines neuen Benutzers |
| /login | POST | 200=OK 401=Unauthorized | Eingabe: User.json | Login des übergebenen Benutzers |
| /listUsers | POST | 200=OK | Rückgabe: Users.json | Auflistung aller User |
| /addGroup | POST | 200=OK 401=Unauthorized | Eingabe:Group.json | Anlegen einer neuen Gruppe |
| /listGroups | GET | 404=NotFound | Rückgabe: Groups.json | Auflistung aller Gruppen |
| /createBlogentity | POST | 200=OK  401=Unauthorized | Eingabe: BlogEntity.json | Neuen Blogeintrag erstellen |
| /getBlogEntities | POST | 404=NotFound | Rückgabe: BlogEntities.json | Alle Blogeinträge auflisten |
| /addComment | POST | 200=OK  401=Unauthorized | Eingabe: Comment.json | Übergebenes Kommentar zu einem Blogeintrag hinzufügen |

# Entwicklung mobiler Client

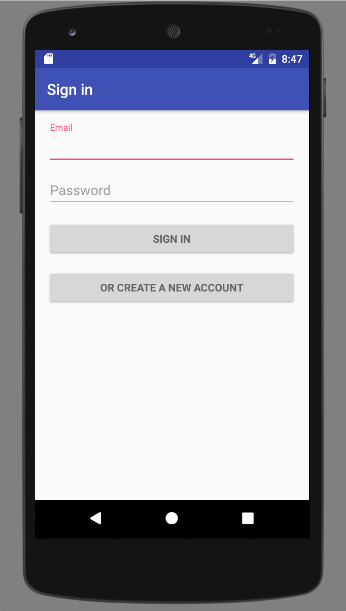
Der mobile Client wurde als Android App entwickelt, da schon Erfahrung im Bereich Android Programmierung vorhanden war. Die App ist für Android 7.1.1 konzipiert und wurde auch in dieser Version getestet.

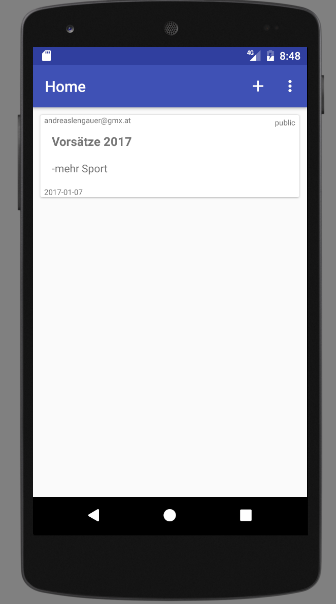
Alle Funktionen des Web-Client wurden in der App ebenfalls implementiert. Diese wurden in verschiedenen Android Activities umgesetzt welche später noch genauer erklärt werden.

## Beschreibung der Oberfläche

**Login**

Die Activity Login zeigt die Eingabefelder für den Usernamen und Passwort. Darunter sind die Buttons zum Einloggen und zum Anlegen eines neuen Benutzers.

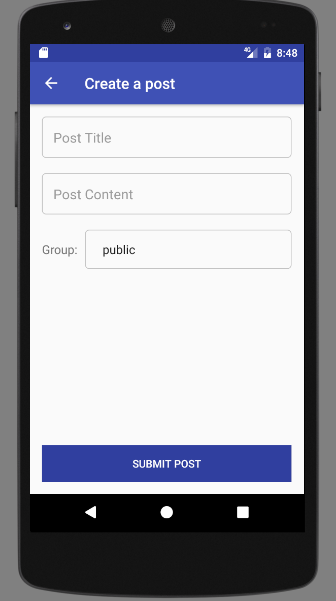


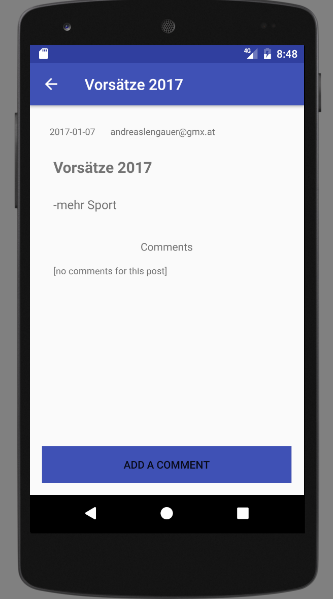
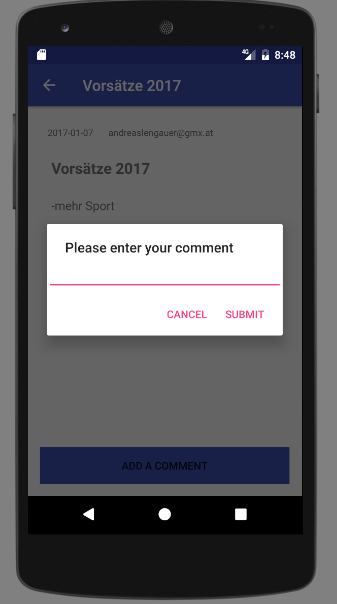
**Home**

Diese Ansicht zeigt alle Blogeinträge welche der eingeloggte User sehen darf. Über das Plus Symbol in der Menüleiste besteht die Möglichkeit einen neuen Blogeintrag zu erstellen. In den weiteren Optionen in dieser Leiste können Gruppen erstellt werden und der Benutzer sich ausloggen.

**Neuer Blogeintrag**

In dieser Ansicht befinden sich die Eingabefelder für den Titel und den Inhalt des neuen Blogeintrags. Zusätzlich kann über die Auswahlliste die Gruppe bestimmt werden, welche Zugriff auf diesen Eintrag haben darf. Durch den Submit-Button wird der Eintrag erstellt.



**Detailansicht** **Blogeintrag**

Es werden alle Details zu einem Eintrag angezeigt inklusive Kommentare.   
Es besteht natürlich auch die Möglichkeit ein Kommentar hinzuzufügen welches danach unter Comments angezeigt wird.

# Diskussion Rest Konzept

Das RESET-Konzept ist durch folgende Merkmale definiert:

* Ressource
  + Zentrale Abstraktion für Informationen
  + Statische und dynamische Ressourcen
  + Ansprache über einen Ressourcen-Namen
  + Referenz von Konzepten statt einer konkreten Ausprägung eines Konzepts
* Adressierbarkeit
  + Ressourcen sind universell adressierbar und können damit von anderen Komponenten des Systems angesprochen werden
  + Keine „physische“ URL
  + Rückgabewert ist eine Repräsentation der Ressource
* Repräsentation
  + Manipulationen auf die Ressource werden mittels Repräsentation durchgeführt
  + Repräsentiert aktuellen oder gewünschten Zustand der Ressource
  + Bytesequenz + Metadaten zur Beschreibung
  + Kann verschiedene Datenformate (media type) haben
  + Content Negotiation, wenn es mehrere Repräsentationen zu einer Ressource gibt
* Universelle Schnittstelle: HTTP
  + HTTP als Anwendungsprotokoll mit folgenden Befehlen:
    - Holen einer Ressource Repräsentation: GET
    - Erzeugen einer neuen Ressource
      * Neue URL: PUT
      * Existierende URL: POST
    - Modifizieren einer Ressource: PUT
    - Löschen einer Ressource: DELETE
    - Holen von Metadaten: HEAD
    - Prüfen was Ressource unterstützt: OPTIONS
* Zustandslose Kommunikation
  + Jeder Request von Client zur Server enthält alle Informationen die zur Ausführung notwendig sind
  + Kein Kommunikationskontext notwendig
  + Vorteile: Skalierbarkeit, Verständlichkeit, Zuverlässigkeit
  + Nachteil: höhere Netzbelastung
* Caching
  + Daten eines Requests können als „cachable“ oder „non-cachable“ markiert werden
  + Vorteile: geringere Latenz, höhere Effizienz, bessere Skalierbarkeit
  + Nachteile: Reduktion der Zuverlässigkeit, wenn Unterschied zwischen Cache und tatsächlichen Daten signifikant
* Hypertext gesteuert
  + Client navigiert ausschließlich über URLs, welche vom Server bereitgestellt werden
  + Bereitstellung der URIs über Hypermedia
  + Eine lose Bindung wird gewährleistet und die Schnittstelle kann verändert werden
  + Registrierungsdatenbanken o.Ä. werden benötigt

Durch den Vergleich dieser grundsätzlichen Merkmale des REST-Konzepts mit der Blog-Service Schnittstelle, erhält man folgende Ergebnisse:

* Client-Server Prinzip ist erfüllt, da klar zwischen Dienstnutzer und Dienstanbieter unterschieden werden kann
* Die Zustandslosigkeit ist ebenfalls gegeben, da alle benötigten Daten bei jeder Anfrage übertragen werden
* Caching wird nicht verwendet jedoch auch nicht blockiert und daher nicht REST-konform umgesetzt
* Die Adressierbarkeit von Ressourcen ist ebenfalls nicht REST konform, da bestimmten Parameter im Request übergeben werden
* Die Repräsentation zur Veränderung von Ressourcen ist REST konform, weil eine Veränderung einer Ressource nur mit der POST Methode möglich ist
* Die Schnittstelle bietet keine universelle Schnittstelle an und ist auch nicht REST konform in dieser Hinsicht
* HATEOAS wurde in dem Webservice auch nicht implementiert und somit muss der Benutzer der Schnittstelle über die einzelnen Ressourcen Bescheid wissen.