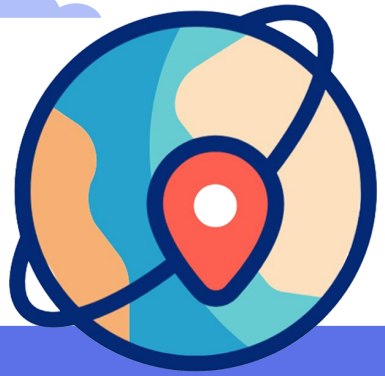
Image created with Microsoft Copilot Designer

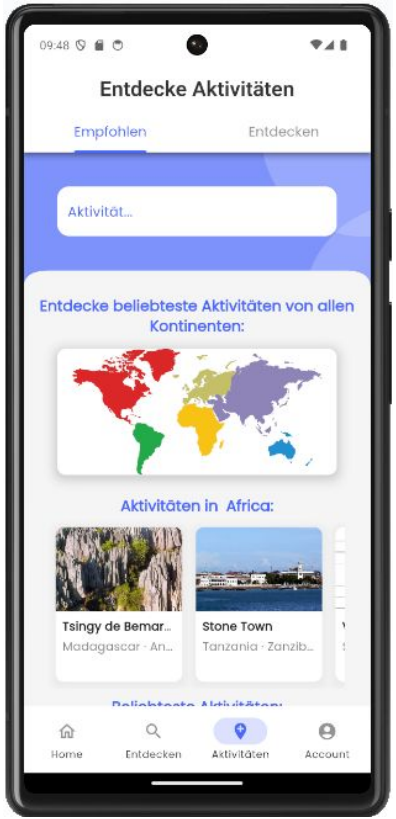


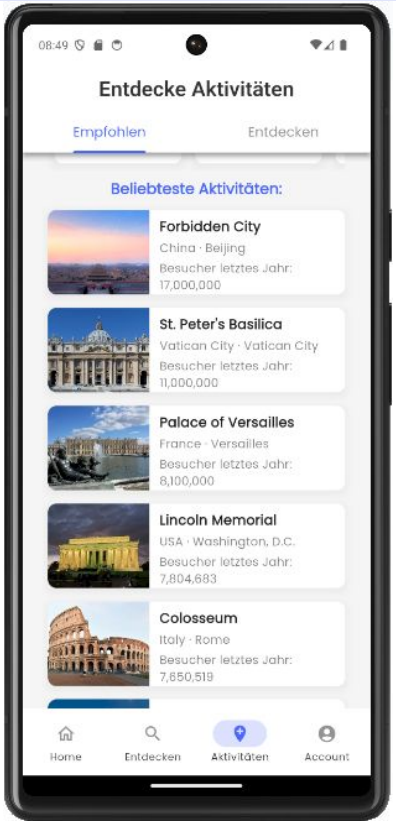
TraveLink

Abschlusspräsentation


Aktivitäten entdecken

Aktivitäten entdecken



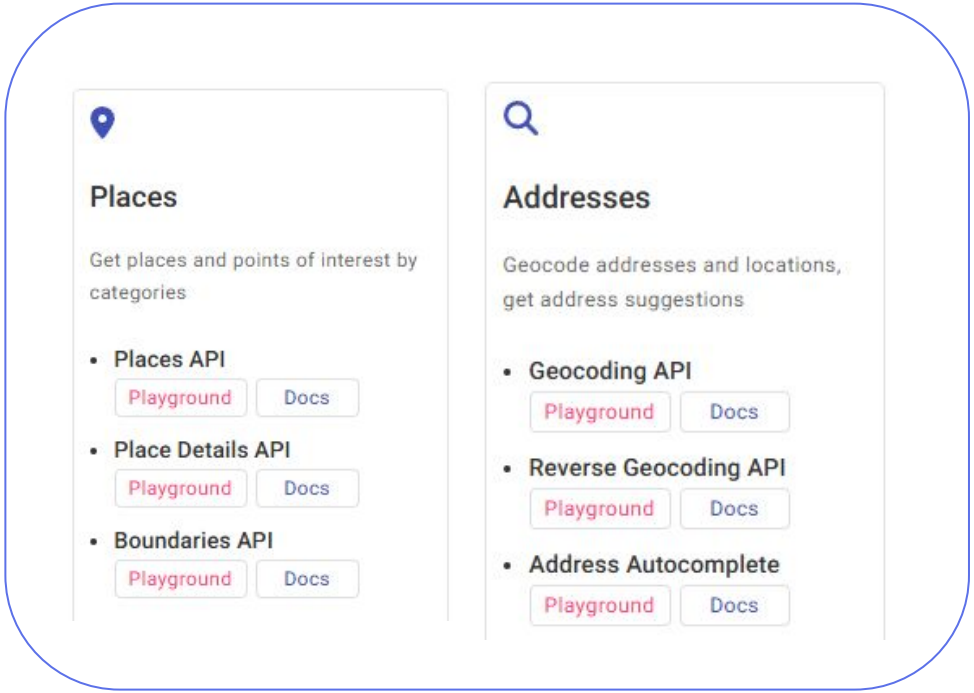


Aktivitäten entdecken:

- Suchleiste: Aktivität oder Stadt, Auswahl an Kategorien
- Unterstützung:
  - Beliebteste Aktivitäten auf den Kontinenten, Zufällige Aktivitäten, beliebteste Aktivitäten
  - Filter: Kategorie, Land, Stadt

Aktivitäten

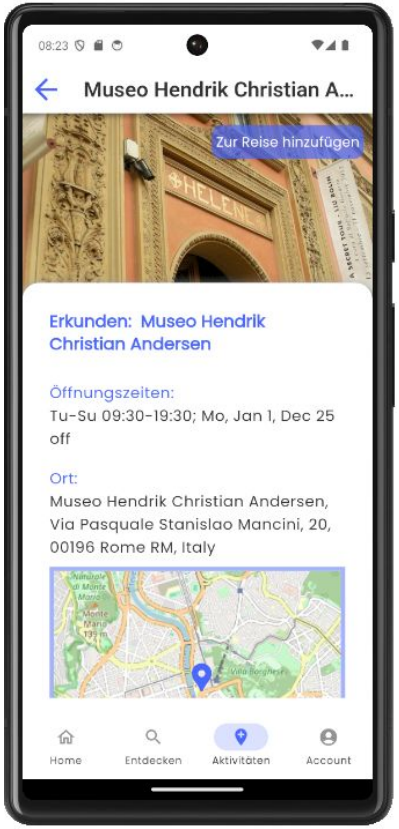
Aktivitäten

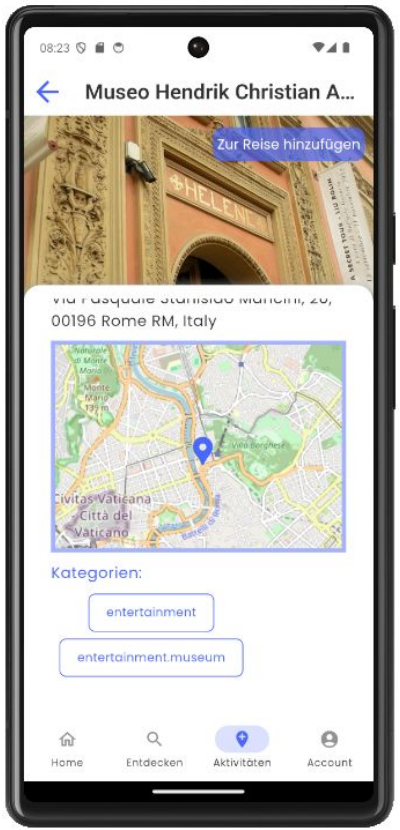


API Geogify:

- Aktivitäten in Stadt mit Kategorien
- Informationen über Aktivitäten
- Autocomplete für die Suchleisten
- Umwandlung von Koordinaten in Adresse

Aktivität - Details

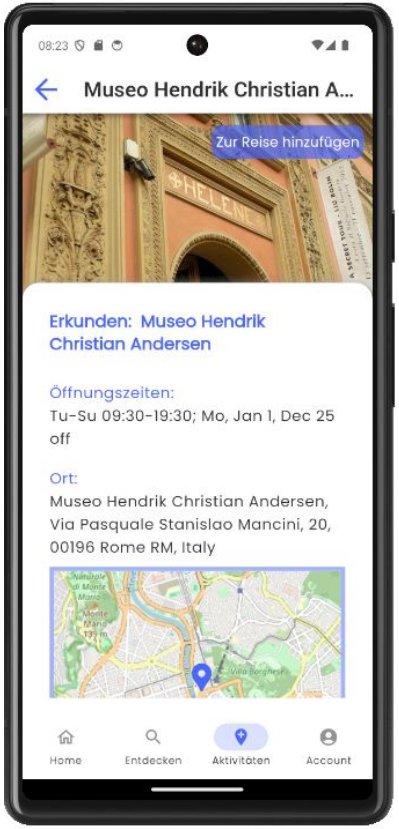


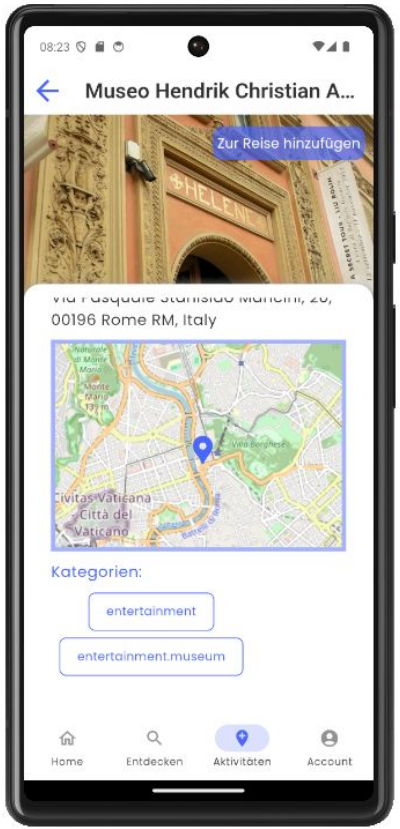


Aktivität:

- Bild
- Name
- Beschreibung/ Öffnungszeiten
- Ort
- Karte
- Kategorie
- Zur Reise hinzufügen

Aktivität - Details






```
class Activity {
  Activity({
    required this.name,
    required this.categories,
    required this.location,
    this.wikidataUrl,
    this.wikidataId,
    this.openingHours,
    this.website,
    this.placeId,
    this.imagePaths = const [],
    this.description = '',
    this.continentType,
    this.isPublic = true,
    this.isUserCreated = false,
    this.creatorId,
    this.image,
    this.imageBytes,
    this.amountVisitors,
    this.firebaseId,
  });
```

```
class PlaceLocation {
  const PlaceLocation({
    required this.lat,
    required this.lon,
    required this.city,
    required this.country,
    required this.formatted,
    required this.countryCode,
  });
```

Probleme + Herausforderungen

- **Aktivitäten:**
  - Welche und woher?
  - Verschiedene Quellen, zusammen anzeigen
- **API Geogify:**
  - An API halten: vorgegebene Struktur, Kategorien
  - Nicht alle Aktivitäten haben die gleiche Menge an Daten (Bsp. keine Beschreibung)
  - Nur eine bestimmte Menge laden
- **Wikipedia/media API:**
  - Nicht alle Aktivitäten habe ein Bild ⇒ Placeholder
- **Anpassungen für Web:**
  - Bilder hochladen, Location (Bsp. Package nur für Handy)
- **Standort:**
  - Zuordnung zu Land/Kontinent mit hilfe von Koordinaten

Beitrag Dawid

Dokumentation

- Paper Prototypes
- Figma Explore Activities
- Explore Activities User Flow + Data Flow
- Cognitive Walkthrough
- Heuristic Evaluation

Implementation

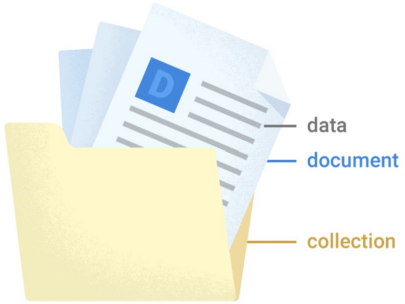
Allgemein

- Theme (Aussehen)
- Geoapify API
  - Aktivitäten
- Wikipedia/Wikidata API
  - Bilder

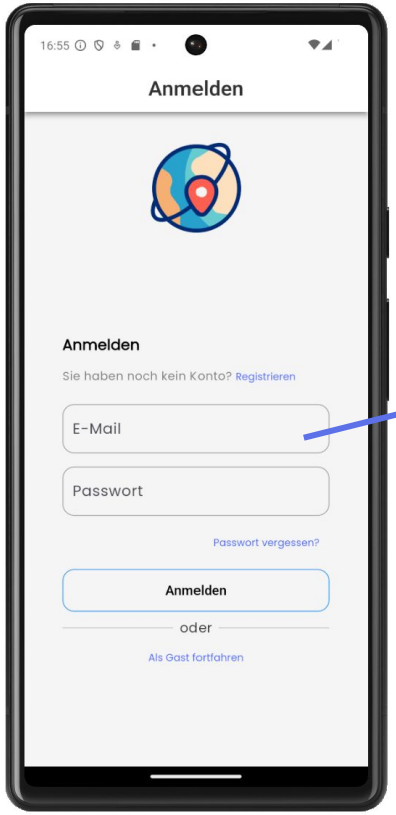
Features / Screens

- Explore Activities: Main Screen (Recommended)
- Explore Activities: Search Screen (Explore All)
- Continents Screen
- 2x Activities Screen (List of Activities)
- Filter Screen
- Activity Details Screen
- Add Activity

Firebase Integration



Firebase Authentication



Firebase Authentication

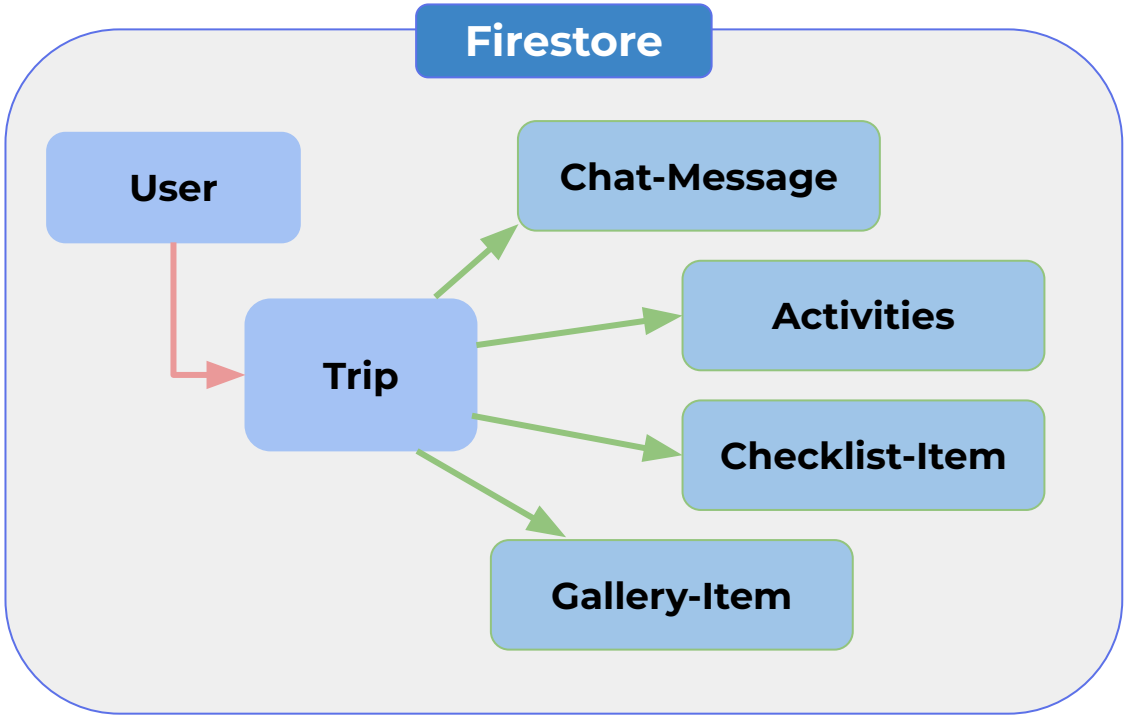
Speichert
- User-id
- Email
- passwort-Hash

Cloud Firestore

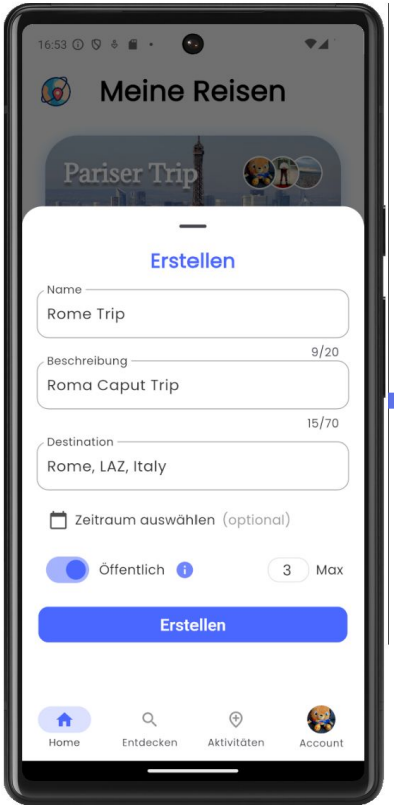
User

Speichert
- user-id
- username
- picture-url
- position
- Description
- Languages
- Interests

Firebase Architektur



Firebase Reise erstellen



Trip

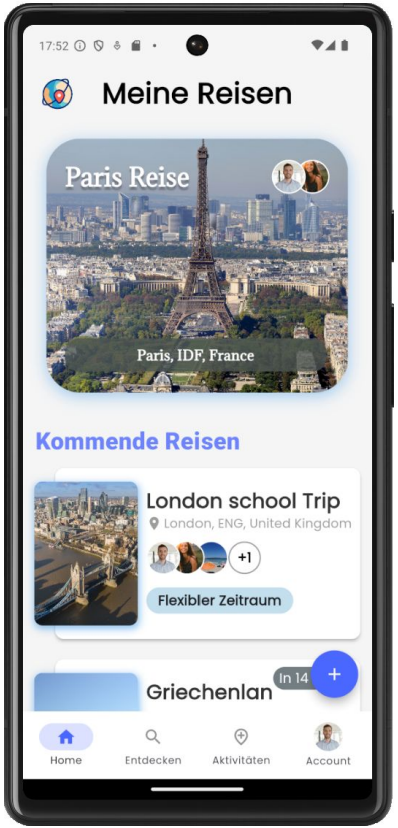
String tripId
String name
String description
Destination destination
String[] images
Boolean isPublic
Int maxParticipants
User[] participants
DateTime startDate
DateTime endDate

Geoapify

WIKIDATA

User

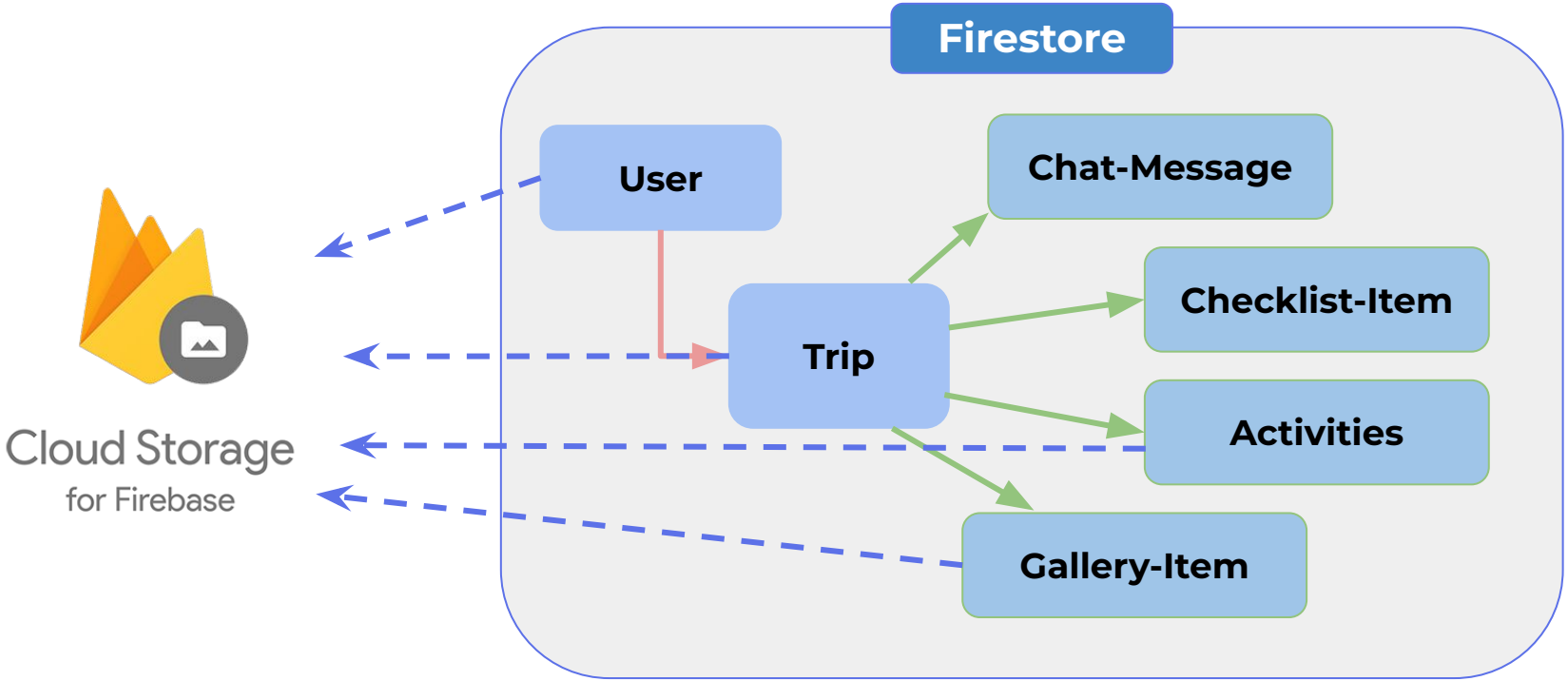
Reisen Screens



Trip

String tripId
String name
String description
Destination destination
String[] images
Boolean isPublic
Int maxParticipants
User[] participants
DateTime startDate
DateTime endDate

Firebase Storage



Gallerie: Bild hinzufügen

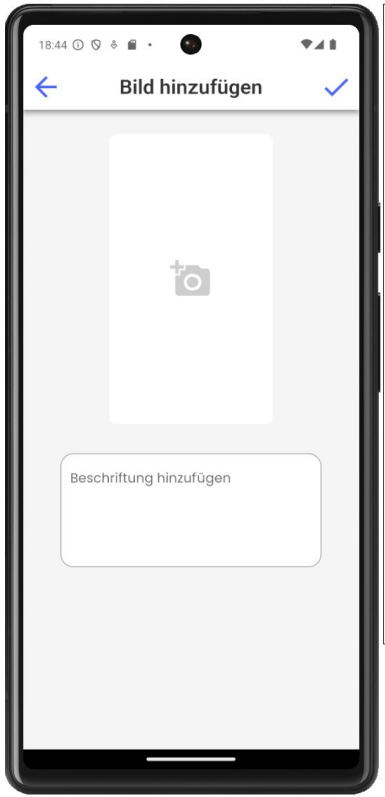


Bild in geteilte Gallerie posten

```dart
class SharedGalleryRepository {
  const SharedGalleryRepository(this._firestore, this._storage);
  final FirebaseFirestore _firestore;
  final FirebaseStorage _storage;

  static String tripGalleryPath(String tripId) => 'trips/$tripId/gallery';

  // create
  Future<void> postPicture({
    required String description,
    required String uid,
    required Uint8List picture,
    required String tripId,
  }) async {
    final timestamp = Timestamp.now();
    final pictureId = '${tripId}_${uid}_${timestamp.millisecondsSinceEpoch}';
    final firestorageRef = _storage.ref().child('gallery/$pictureId');
    await firestorageRef.putData(
      picture,
      SettableMetadata(contentType: 'image/jpeg'),
    );
    final pictureUrl = await firestorageRef.getDownloadURL();

    return _firestore.collection(
      tripGalleryPath(tripId)).doc(pictureId).set({
      'description': description,
      'uid': uid,
      'picture': pictureUrl,
      'timestamp': timestamp,
    });
  }
  ...
}
```

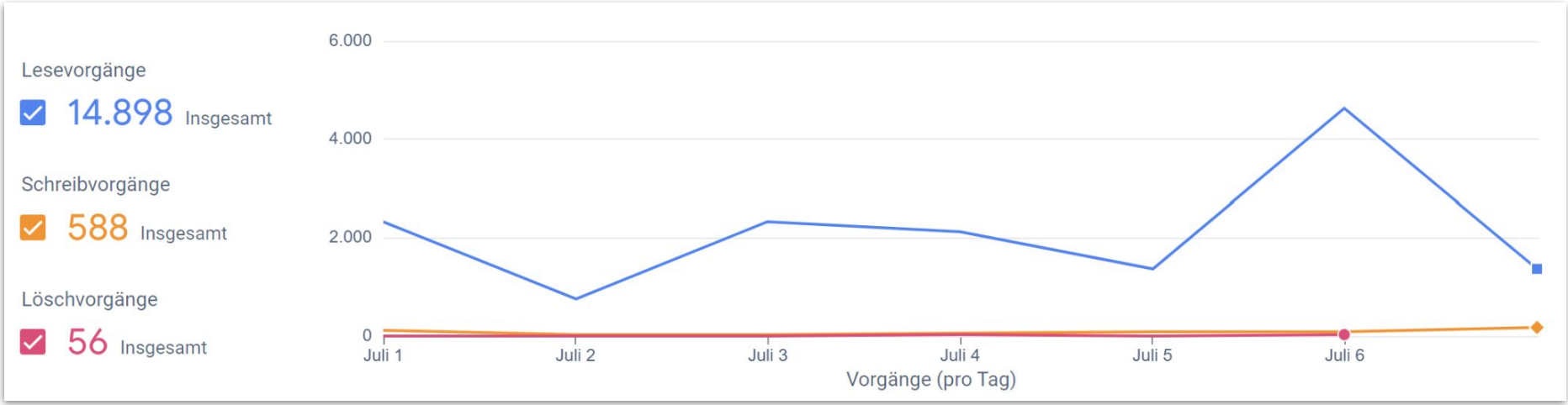
Gallerie: Bilder laden

Bilder laden

```
...
Future<List<PicturePost>> fetchPicturePosts({required String tripId}) async
{   final posts = await queryTripsPicturePosts(tripId: tripId).get();
    return posts.docs.map((doc) => doc.data()).toList();
  }

  //QUERIES
  Query<PicturePost> queryTripsPicturePosts({required String tripId}) =>
      _firestore
          .collection(tripGalleryPath(tripId))
          .orderBy('timestamp', descending: true)
          .withConverter(
            fromFirestore: (snapshot, _) =>
                PicturePost.fromMap(snapshot.data()!, snapshot.id),
            toFirestore: (picturePost, _) => picturePost.toMap(),
          );
...
```

Nutzung

Letzten 7 Tage



Insgesamt ~170k Lesevorgänge

2GB gesendete Bandbreite Firebase Storage

Beitrag Ante

Dokumentation

- Konkurrenzanalyse
- Interview Guide und Survey Guide
- User Scenarios
- MVP Definition
- Vision Statement
- User Evaluation Tasks
- Paper Prototypes
- Figma Main Pages und Trip Section
- User und Data-Flow
- Implementation
- Thinking Aloud Tests
- Accessibility Tests

Implementation

Allgemein

- Routing
- Firebase Integration
  - Firebase Auth
  - Firestore
  - Firebase Storage
- Geoapify API und wikidata
  - Auto-complete
  - Aktivitäten
- Pipeline

Features / Screens

- Anmelden, Registrieren / 2
- Meine Reisen / 1
- Erstelle Reisen / 3
- Entdecke Reisen / 3
- Reisen Management / 1
- Reise Übersicht / 1
- Reise Teilnehmer / 2
- Reise Chat / 1
- Reise Gallerie / 3
- Reise Aktivitäten / 1

57 Issues

Checklist

Checklist

Image created with Microsoft Copilot Designer

Teil Features Checklist

- Items abhaken
- Items löschen
- Items bearbeiten
- Eigene Items
- Item Vorschläge
- Erledigungsdatum setzen
- Persönliche- und Gruppen Checkliste
- Trip Mitgliedern Aufgaben zuweisen
- Fortschrittsanzeige in der Gruppen Checkliste
- Item Vorschau

Inspiration - GitLab

Trip Mitglieder Aufgaben zuweisen

Assigning User in GitLab

Erledigungsdatum setzen

Due dates in GitLab

Fortschrittsanzeige in der Gruppen Checkliste

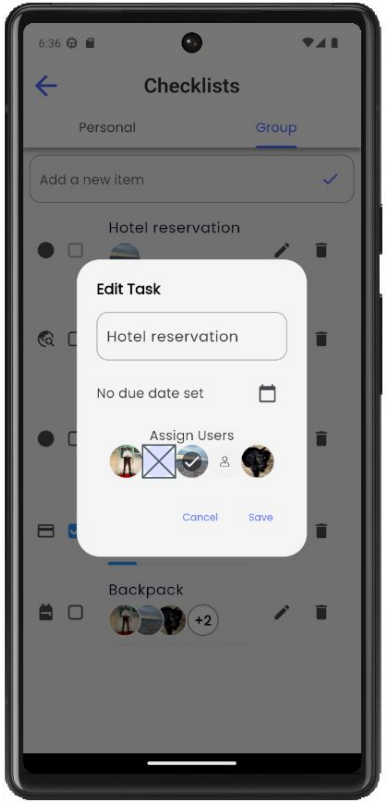
Labels in Git (In progress, nearly done, done)

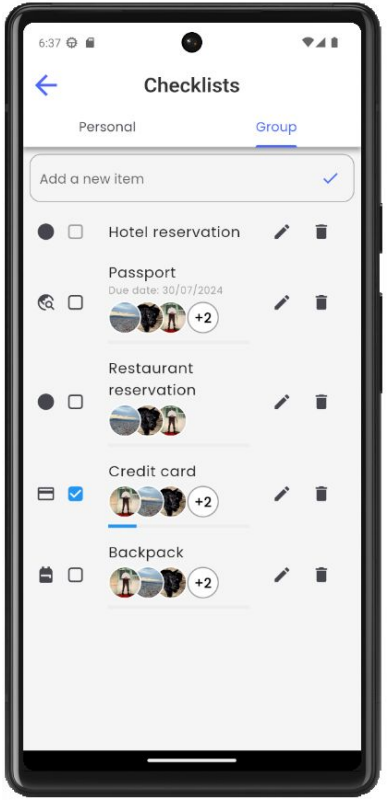
Probleme + Herausforderungen

- Item Vorschläge in einer ListView mit Focus Node
    - AutoComplete

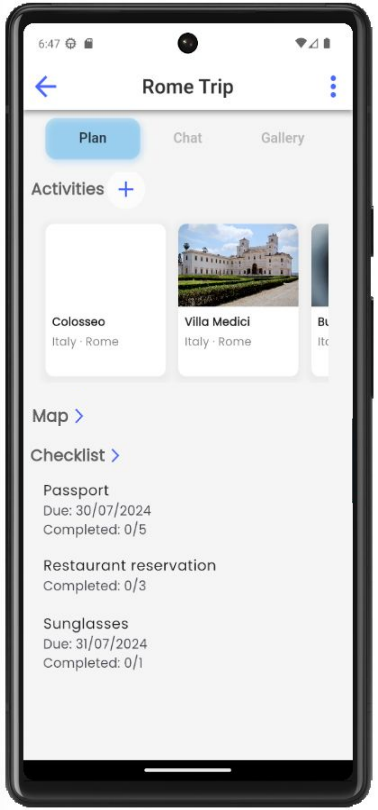
- Nicht umgesetzte Idee: Vorschläge nach bereits eingegebenen Items ändern

Assign Users





Checklist Item Preview



Beitrag Tom

Dokumentation

- NZSE User Research
- Preparing Interview guideline
- Restructure Interview Guide after test Interview
- Interview durchführen
- Results of Interview
- Persona profile
- Paper Prototype
- User flow diagram for Checklist
- Figma screens for Checklist
- Checklist documentation
- Example usage
- Heuristic analysis
- Thinking aloud test
- Selfreflection

Implementation

Allgemein

- Item suggestions
- Icons

Features / Screens

- Designing the Checklist Page
- Plan Trip screen Checklist preview
- Personal Checklist
- Group Checklist

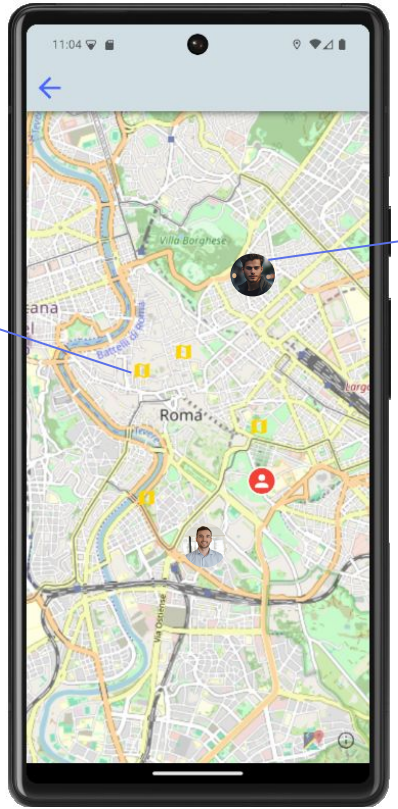
Trip Map

Code to generate the Map

```dart
@override
Widget build(BuildContext context) {
  final sharedState = ref.watch(sharedStateProvider);

  return Scaffold(
    backgroundColor: const Color.fromARGB(255, 204, 219, 226),
    appBar: AppBar(
      backgroundColor: const Color.fromARGB(255, 204, 219, 226),
    ), // AppBar
    body: Row(
      children: [
        Expanded(
          flex: 3,
          child: activitiesFetched
              ? FlutterMap(
                  options: MapOptions(
                    initialCenter: LatLng(
                      widget.destination.lat ?? 49.8728,
                      widget.destination.lon ?? 8.6512,
                    ), // LatLng
                    minZoom: 3,
                    maxZoom: 18,
                    initialZoom: 8,
                  ), // MapOptions
                  // ignore: deprecated_member_use
                  nonRotatedChildren: [
                    RichAttributionWidget(
                      attributions: [
                        TextSourceAttribution(
                          'OpenStreetMap contributors',
                          onTap: () => launchUrl(
                            Uri.parse(
                              'https://openstreetmap.org/copyright',
                            ),
                          ),
                        ), // TextSourceAttribution
                      ],
                    ), // RichAttributionWidget
                  ],
                  children: [
                    TileLayer(
                      urlTemplate:
                          'https://tile.openstreetmap.org/{z}/{x}/{y}.png',
                      userAgentPackageName: 'com.example.app',
                    ), // TileLayer
                    CurrentLocationLayer(
                      style: LocationMarkerStyle(
                        marker: const DefaultLocationMarker(
                          color: Colors.red,
                          child: Icon(
                            Icons.person,
                            color: Colors.white,
                          ), // Icon
                        ), // DefaultLocationMarker
                        markerSize: const Size.square(35),
                        accuracyCircleColor: Colors.green.withOpacity(0.1),
                        headingSectorColor: Colors.green.withOpacity(0.8),
                      ), // LocationMarkerStyle
                      moveAnimationDuration: Duration.zero,
                      positionStream: const LocationMarkerDataStreamFactory()
                          .fromGeolocatorPositionStream(
                        stream: Geolocator.getPositionStream(
                          locationSettings: const LocationSettings(
                            accuracy: LocationAccuracy.high,
                            timeLimit: Duration(minutes: 1),
                          ), // LocationSettings
                        ),
                      ),
                    ), // CurrentLocationLayer
                    PolylineLayer(
                      polylines: [
                        Polyline(
                          points: sharedState.way,
                          color: Theme.of(context).primaryColor,
                          borderColor: Colors.deepPurple,
                          borderStrokeWidth: 5,
                        ), // Polyline
                      ],
                    ), // PolylineLayer
                    MarkerLayer(
                      markers: userMarkers + activitiesMarker,
                    ), // MarkerLayer
                  ],
                ) // FlutterMap
              : const Center(child: CircularProgressIndicator()),
        ), // Expanded
      ],
    ), // Row
  ); // Scaffold
}
```

Map mit Activitäten und Usern

Aktivität

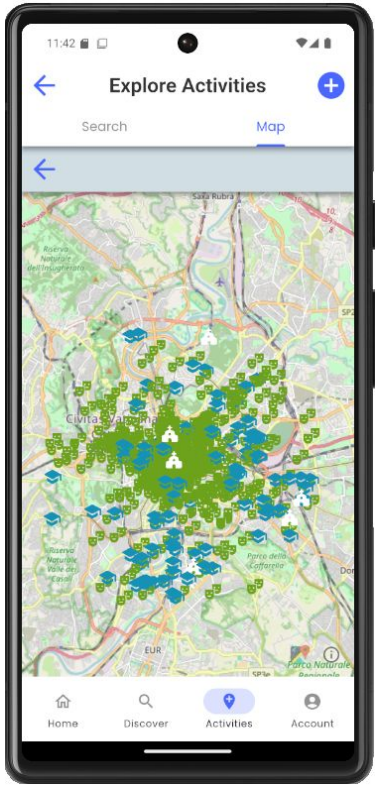


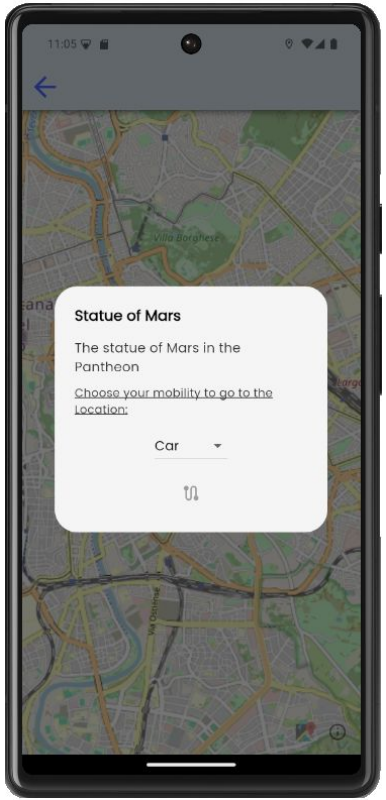
User-Position

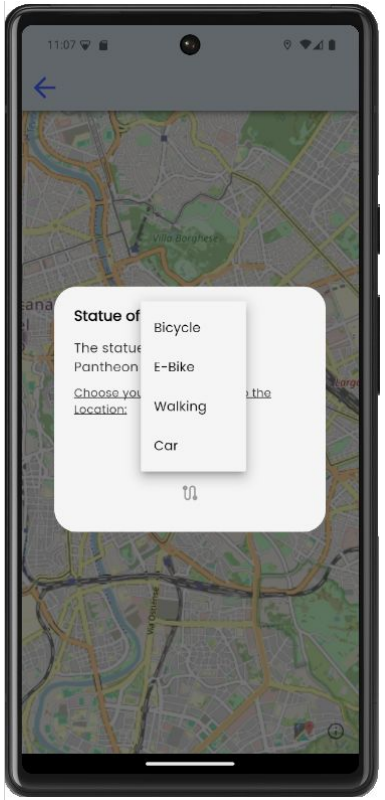
Code for a Marker

```
Marker createEntertainmentActivity(
    LatLng position, String name, String description, WidgetRef ref) {
  final LatLng locationPosition = position;
  String usedMobility = 'Car';
  String mobilityForAPI = 'cycling-regular';
  final ValueNotifier<Color> colorNotifier =
      ValueNotifier<Color>(const Color.fromARGB(255, 109, 162, 30));
  final ValueNotifier<Color> starColorNotifier =
      ValueNotifier<Color>(Colors.grey);
  final ValueNotifier<bool> isLoadingNotifier = ValueNotifier<bool>(false);

  return Marker(
    point: position,
    width: 40,
    height: 40,
    child: ValueListenableBuilder<Color>(
      valueListenable: colorNotifier,
      builder: (context, color, child) {
        return IconButton(
          icon: Icon(Icons.theater_comedy, color: color),
          onPressed: () {
            // ignore: inference_failure_on_function_invocation
            showDialog(
              context: context,
              builder: (context) => AlertDialog(
                title: Text(name), // Set the title to the name
                content: StatefulBuilder(
                  builder: (context, setState) {
                    return Column(
                      mainAxisSize: MainAxisSize.min,
                      children: <Widget>[
                        Text(description), // Display the description
                        const SizedBox(height: 10), // Add some spacing
                        // Text above the dropdown menu with smaller size and underlined
                        const Text(
                          'Choose your mobility to go to the Location:',
                          style: TextStyle(
                            fontSize: 14, // Smaller font size
                            decoration:
                                TextDecoration.underline, // Underlined text
                          ), // TextStyle
                        ), // Text
const SizedBox(height: 10), // Add some spacing
// Dropdown menu
DropdownButton<String>(
  value: usedMobility,
  items: <String>['Bicycle', 'E-Bike', 'Walking', 'Car']
      .map((String value) {
    return DropdownMenuItem<String>(
      value: value,
      child: Text(value),
    ); // DropdownMenuItem
  }).toList(),
  onChanged: (value) {
    setState(() {
      if (value != null) {
        usedMobility =
            value; // Update the selected value
        mobilityForAPI = translateMobilityToApiReadable(
            usedMobility);
      }
    });
  },
), // DropdownButton
const SizedBox(height: 10),
ValueListenableBuilder<bool>(
  valueListenable: isLoadingNotifier,
  builder: (context, isLoading, child) {
    return Column(
      mainAxisSize: MainAxisSize.min,
      children: <Widget>[
        ValueListenableBuilder<Color>(
          valueListenable: starColorNotifier,
          builder: (context, starColor, child) {
            return IconButton(
              icon: Icon(
                Icons.route_rounded,
                color: starColor,
              ), // Icon
              onPressed: () async {
                isLoadingNotifier.value = true;
                final LatLng currentUserLocation =
                    await getCurrentLocation();
                List<LatLng> route;
                                route = await calculateRoute(
                                  mobilityForAPI,
                                  convertLatLngToCoordinates(
                                      currentUserLocation),
                                  convertLatLngToCoordinates(
                                      locationPosition),
                                );

                                // Update the shared state
                                // with the new route
                                ref.read(sharedStateProvider).way =
                                    route;

                                isLoadingNotifier.value = false;
                              },
                            ); // IconButton
                          },
                        ), // ValueListenableBuilder
                        if (isLoading)
                          const CircularProgressIndicator(),
                      ], // <Widget>[]
                    ); // Column
                  },
                ), // ValueListenableBuilder
              ], // <Widget>[]
            ); // Column
          },
        ), // StatefulBuilder
      ), // AlertDialog
    );
  },
); // IconButton
      },
    ), // ValueListenableBuilder
  ); // Marker
}
```

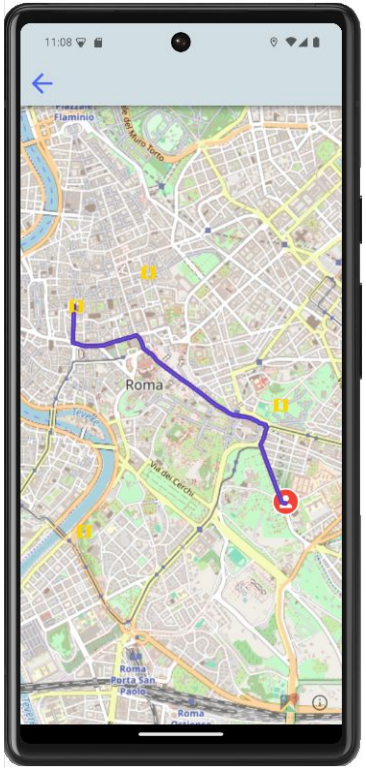
Display of Activities in Rome

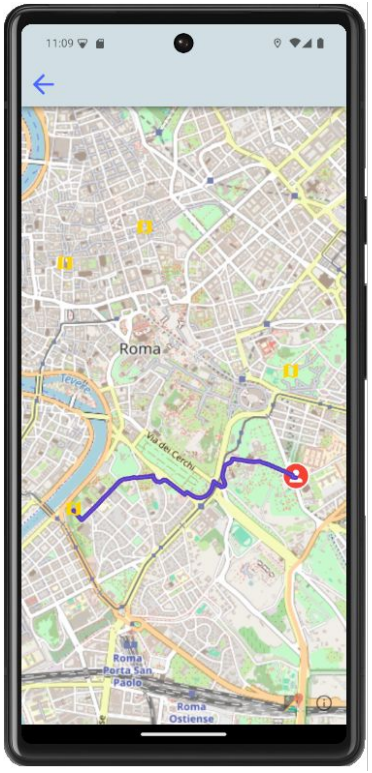






Route Calculation

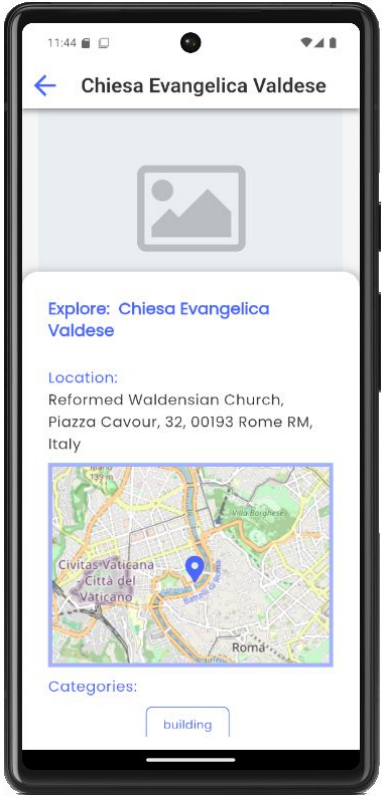




Code to calculate the Route

```dart
Future<List<LatLng>> calculateRoute(
  String mobility,
  Coordinates startPunkt,
  Coordinates endPunkt,
) async {
  final List<LatLng> way = [];
  final PolylinePoints polylinePoints = PolylinePoints();

  final url =
      Uri.parse('https://api.openrouteservice.org/v2/directions/$mobility');

  final headers = {
    'Accept':
        'application/json, application/geo+json, application/gpx+xml, img/png; charset=utf-8',
    'Authorization': apiKey,
    'Content-Type': 'application/json; charset=utf-8',
  };

  final request = {
    'coordinates': [
      [startPunkt.longitude, startPunkt.latitude],
      [endPunkt.longitude, endPunkt.latitude],
    ],
    'maneuvers': 'true',
  };

  final response =
      await http.post(url, headers: headers, body: jsonEncode(request));

  // ignore: strict_raw_type
  final Map jsonData = jsonDecode(response.body) as Map;

  // Check if the Coordinates are Invalid,
  // Should they be invalid the Map that is returned containes 2 and not 3 elements
  if (jsonData.containsKey('error')) {
    return way;
  }

  final features = jsonData['routes'][0];
  final String geometry = features['geometry'] as String;
  final List<PointLatLng> result = polylinePoints.decodePolyline(geometry);
```

Detailed view of Activity



Beitrag Yannick

Dokumentation

- Preparing Interview guideline
- Restructure Interview Guide after test Interview
- Conduct Interview
- Results of Interview
- Persona acquisition and evaluation
- Paper Prototype analysis
- User flow diagram for Map
- Figma screens for Map
- Map documentation
- Heuristic analysis
- Thinking aloud test
- Tasks for User Evaluation

Implementation

Allgemein

- Icons for the various activities
- Icon Colour for the various activities

Features / Screens

- Designing the Map Page
- Display Activities from a trip on the Map
- Display all Activities from a Location on a Map
- Route Calculation from Marker
- Marker Creation for Individual Activities

Account

Account Management

Account

Accountverwaltung
Login Informationen

Einstellungen
App-Konfiguration

Hilfe
Kontakt &
Hilfestellungen

Über uns
Entwicklerinformationen &
App-Version

Account


Accountverwaltung
Login Informationen
Einstellungen
App-Konfiguration
Hilfe
Kontakt &
Hilfestellungen
Über uns
Entwicklerinformationen &
App-Version

Account


Accountverwaltung
Login Informationen
Einstellungen
App-Konfiguration
Hilfe
Kontakt &
Hilfestellungen
Über uns
Entwicklerinformationen &
App-Version

Account

Accountverwaltung
Login Informationen

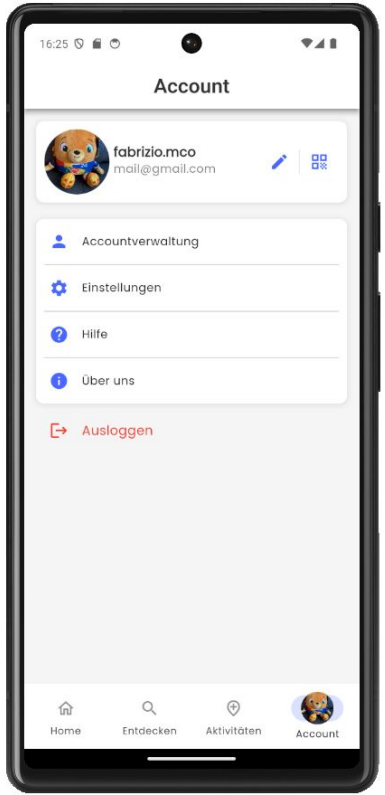
Einstellungen
App-Konfiguration

Hilfe
Kontakt &
Hilfestellungen

Über uns
Entwicklerinformationen &
App-Version

Account

- Accountverwaltung & Einstellungen



- Zugriff auf öffentliches Profil
- QR-Profil-Code

Profile

Profilverwaltung & Vernetzung von Nutzern

Profile

Aufrufen

Profile von Nutzern sollten gut einsehbar sein

Bearbeiten

Das eigene Profil sollte man anpassen können

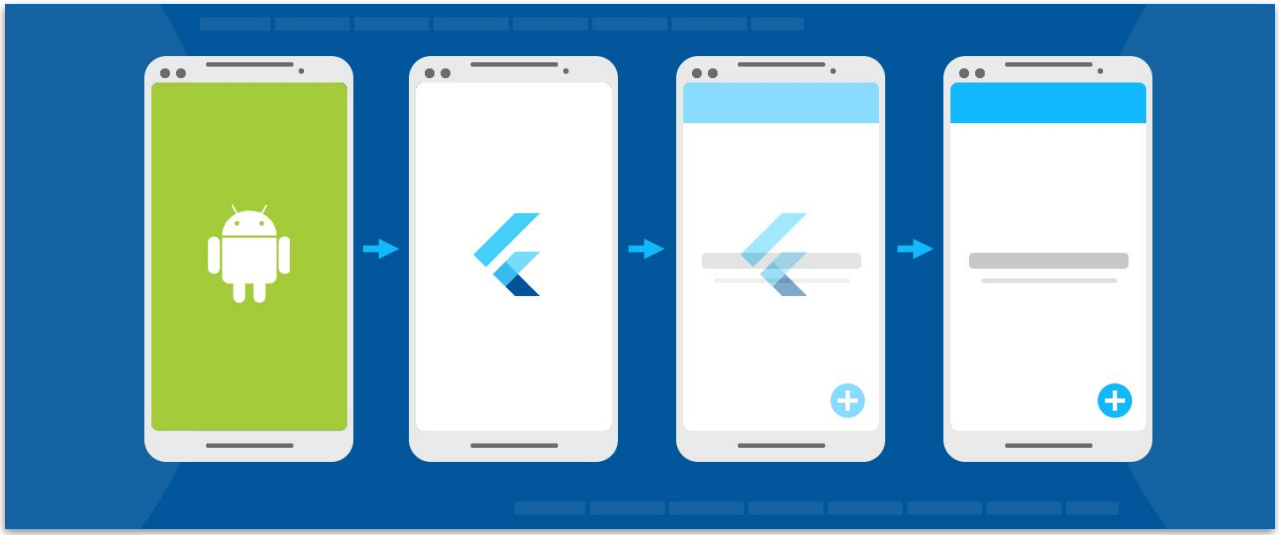
Vernetzen

Schnelles Vernetzen mit QR-Codes

Branding

Logos & Splash Screen

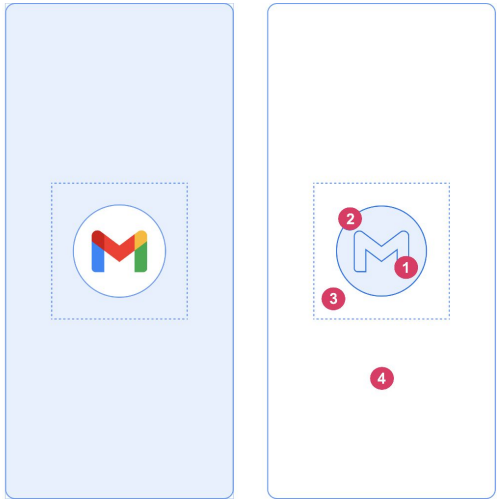
Splash Screens



Splash Screens

Anpassungen

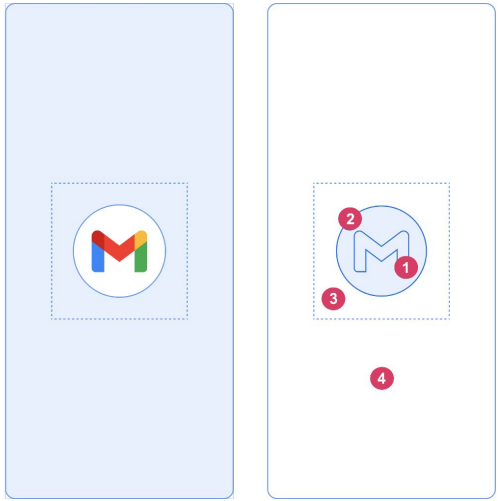
1. App-Symbol (=Icon)
2. Symbolhintergrund
3. Adaptiver Hintergrund
4. Fensterhintergrund
5. Branding

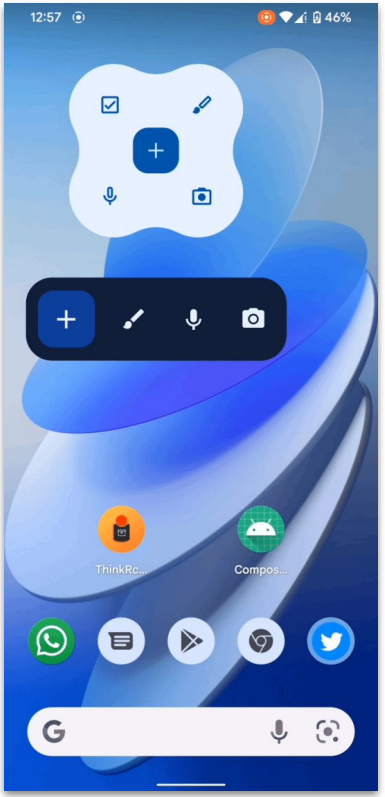


Splash Screens

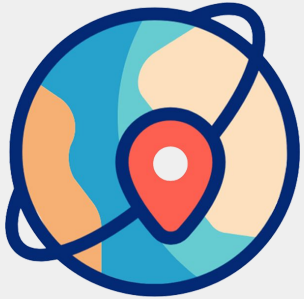
Anpassungen

1. App-Symbol (=Icon)
2. Symbolhintergrund
3. Adaptiver Hintergrund
4. Fensterhintergrund
5. Branding





Branding



App-Logo

h_da

h_da Logo

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

h_da Banner

Beitrag Fabrizio

Dokumentation

- Umfragen Auswertung
- Account Management
- Profile Management
- QR-Code Integration
- User Flow
- Data Flow

Implementierung

Allgemein

- App-Branding
- User Interaktion
- User Profil Integration in verschiedene Widgets

Features / Screens

- Account Screen
- Edit Profile Screen
- Account Information Screen
- Settings Screen
- Help Screen
- About Screen

TraveLink

h_da hochschule darmstadt

member of EUT+ EUROPEAN UNIVERSITY OF TECHNOLOGY