**Spiegazione Architetture Riverpod**

**Presentation Layer –** *insieme delle classi rappresentanti le pagine dell’app e dei controller logici ad esse associate.*

LoginPage: pagina iniziale all’apertura dell’app (dopo eventualmente una schermata di apertura), permette l’accesso o l’iscrizione dell’utente.

LoginController: riceve i dati di accesso e/o quelli di iscrizione e si collega alle funzionalità relative all’accesso (fornite dal server Java); una volta inseriti i dati, l’applicazione si sposta sulla MapSelectionPage.

MapSelectionPage: pagina contenente la lista dei piani di viaggio già creati dall’utente (se ce ne sono, in caso contrario mostra un messaggio che esorta l’utente a crearne di nuovi); è possibile selezionare una delle mappe disponibili toccando la card relativa, spostandosi sulla TripExpositionPage.

In alternativa, in fondo alla pagina è presente un bottone che permette di creare un nuovo piano di viaggio; toccando il bottone ci si sposta alla CitySelectionPage.

MapSelectionPageController: si interfaccia con il server Java per ottenere la lista delle mappe create in precedenza dall’utente.

CitySelectionPage: pagina contenente una listview delle città disponibili per la visita; l’utente ne può selezionare una e dare conferma con un apposito pulsante per passare alla LocationSelectionPage.

CitySelectionPage Controller: si interfaccia con il server Java per ottenere la lista delle città disponibili nel database.

LocationSelectionPage: viene proposta una lista di località disponibili per la visita nella città selezionata; l’utente può sceglierne uno o più e dare conferma con un pulsante per passare alla RequirementsPage.

LocationSelectionPage Controller: si interfaccia con il server Java per ottenere la lista delle località disponibili nel database, filtrando per la città selezionata precedentemente; inoltre si occupa del recupero e del salvataggio delle immagini relative alle località in una cache locale (questo per velocizzare la visualizzazione delle immagini).

RequirementsPage: pagina dove l’utente può inserire una serie di informazioni relative alle proprie necessità per il viaggio: dove è alloggiato (o per lo meno, da dove vuole partire per questa visita), la velocità di marcia (lenta/media/rapida), il numero di giorni che intende trascorrere a visitare la città, e gli orari di partenza e sosta che gradirebbe fare giorno per giorno; una volta terminata la compilazione, l’utente può confermare e passare alla TripExpositionPage.

RequirementsPageController: si occupa del corretto salvataggio dei dati inseriti dall’utente in un’apposita struttura dati, denominata Mappa (Map), in modo che questa possa essere utilizzata per calcolare l’itinerario di viaggio.

TripExpositionPage: (TripPage, alternativamente) Propone all’utente l’itinerario del viaggio, mostrando con chiarezza orari di arrivo previsti per ogni tappa; è possibile scorrere verticalmente nella pagina per visualizzare tutte le località ordinate e, se sono stati impostati più giorni di viaggio, scorrere orizzontalmente per visualizzare l’itinerario giorno per giorno.

Per le pause pranzo, è possibile selezionare il luogo dove si desidera mangiare tra una lista dei ristoranti più vicini nell’orario definito nei requisiti; questi sono visibili in un widget a comparsa che si apre alla pressione del riquadro della tappa relativa alla pausa pranzo.

Da questa pagina è possibile tornare alla RequirementsPage, qualora fosse necessario modificare i requisiti del viaggio, oppure alla MapSelectionPage, qualora si desiderasse creare o selezionare un itinerario differente.

TripExpositionPageController: se si sta creando una nuova mappa, invia al server il dato di tipo Mappa in modo che il server possa generare, salvare e restituire il piano di viaggio, che viene codificato in un’altra struttura dati, Itinerario;

se si sta aprendo una mappa preesistente, invia una richiesta al server per recuperare i dati dell’itinerario.

Altra funzionalità di questo componente è quella di gestire la ricerca dei ricerca dei ristoranti per gli appositi widget di selezione ristorante; anche in questo caso la ricerca è effettuata inviando una richiesta al server;

**Application Layer –** *buisness logic dell’applicazione- generalizzazione dei casi d’uso.*

InsertUserData: l’applicazione deve permettere all’utente di registrarsi, in modo di tener traccia degli itinerari creati da ogni utente.

MapSelection/Display: l’applicazione deve utilizzare le località e le preferenze inserite dall’utente per creare una tabella di marcia scandita temporalmente, modificabile e consultabile in un secondo momento.

LocationSelection: l’utente deve poter selezionare la città che desidera visitare e i luoghi che è interessato a visitare in tale città.

InsertTripRequirements: l’utente deve essere in grado di specificare le proprie preferenze al fine di vivere un’esperienza su misura.

**Domain Layer –** *insieme delle strutture dati previste per il funzionamento dell’applicazione.*

User: rappresenta i dati di accesso dell’utente all’applicazione

* userName (stringa)
* password (stringa)

Mappa : descrive i dati inseriti dall’utente riguardo al viaggio, prima dell’elaborazione del server

* nomeUtente (string)
* idMappa (string)
* latitudineAlloggio (double)
* longitudineAlloggio (double)
* numeroGiorni (int)
* velocitàMedia (double)
* luoghi (List di oggetti di tipo Luogo)
* giornate (List di oggetti di tipo Giornata)

Itinerario: descrive il piano di viaggio restituito dall’API lato server

* nomeItineratio (string)
* nomeUtente (string)
* giorniViaggio (Dizionario, con chiave la stringa giornoViaggio e valore un oggetto di tipo LuogoEsteso)

Luogo: descrive una località d’interesse in una città tra quelle visitabili dall’utente

* nome (stringa)
* latitudine (double)
* longitudine (double)
* città (string)
* indirizzo (string)
* tipo (string)
* tempoVisita (string)
* immagine (string dell’URL)

LuogoEsteso: si tratta di un oggetto Luogo a cui è assegnato un orario di arrivo

* luogo (Luogo)
* oraArrivo (TimeOfDay – formato di tempo ora:minuti)

Giornata: descrive i requisiti forniti dall’utente per una specifica giornata di viaggio

* oraInizio (TimeOfDay)
* devoPranzare (bool)
* oraPranzo (TimeOfDayx)
* pausa (int)
* tempoPranzo (int)
* tempoVisita (int)

**Data Layer –**  *Repository/tabelle di database alle quali l’app fa riferimento per il proprio funzionamento*

UserData: tabella del database contenente i dati di login di tutti gli utenti, ha la sola funzione di permettere/negare l’accesso all’app.

MapData: tabella del database contenente tutti i dati riguardanti gli itinerari salvati dagli utenti tramite l’utilizzo dell’app.

CityLocationData: tabella del database contenente i dati delle località disponibili per la visita (ogni località ha come attributo la propria città, non c’è una tabella delle città).