Travel Budget Tracker Web-Anwendung

Meilenstein 1a: Anforderungsanalyse

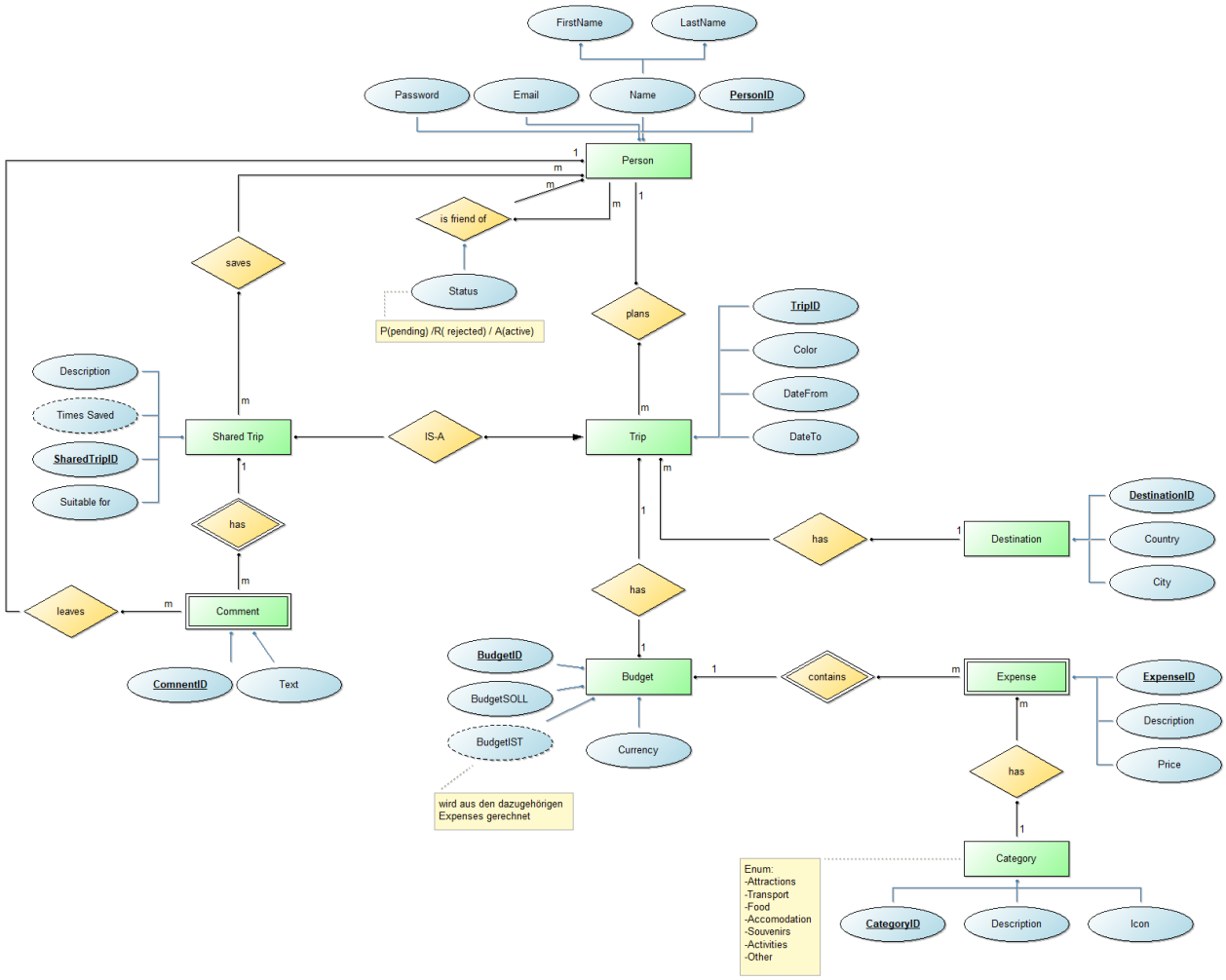
Idee des Projekts:

Die Travel Planner Web-Applikation bietet eine zentralisierte Plattform, um sämtliche Details zu eigenen Reisen zu organisieren, Budgets zu berechnen und Reisepläne zu erstellen und zu entdecken.

UI/UX:

- **Login Page:**
  Die Startseite präsentiert zwei Eingabefelder (E-Mail, Passwort) für den Login. Darunter befindet sich der Login-Button. Zusätzlich steht ein Registrierungsbutton zur Verfügung, falls noch kein Benutzerkonto angelegt wurde.
- **Registrierung Page:**
  Benutzer müssen folgende Daten eingeben: Nickname, First und Last Name, E-Mail und Passwort
- **Landing Page:**
  Es gibt zwei Collections aus dem List der gefundenen Trips: Alle erstellten Trips und Shared Trips. Durch Anklicken des "+"-Buttons kann ein neuer Trip geplant werden. Durch das ‚Edit' Button kann der Trip upgedatet werden. ‚Add Expense' Button gibt Möglichkeit an, die Ausgaben einzugeben. Jeder Trip ist mit seiner Color, Destination (Country + City), Reisedauer und Budgetdetails gekennzeichnet.
- **Create Trip Page:**
  Hier wählt man Color, Destination, Period (Datum From - To), Budget und Currency aus. Außerdem wird hier festgelegt, ob der Trip Shared sein sollte. Walls zutreffend, gibt man zusätzlich Description und Suitable For Felder ein.
- **Edit Trip Page:**
  Hier kann man die eingegebenen Werte beim Erstellen ändern sowie Expenses hinzufügen und ansehen.
- **Administration Page:**
  Hier kann man die Tabelle Destinations, Categories und Users anschauen, wobei bei dem Users Tab kann man eigene Freundschaften steuern.
- **Explore Trips:**
  Es gibt zwei Collections aus dem List der gefundenen Trips: Alle gespeicherten Trips und allen Shared Trips, wobei man ein Trip nicht zwei Mal speichern kann. Durch Anklicken des "Save/Unsave"-Buttons kann ein Trip speichern oder aus dem Gespeicherten gelöscht werden. Mit dem ‚Details'-Button öffnet es sich Shared Trip Detail
- **Shared Trip Detail**
  ist eine Read-Only Maske mit den Details, aufgelisteten Expenses und Kommentaren von verschiedenen Usern.
- **Profile**
  Hier ändert man User-Angaben wie z.B Email, Password und Name. Außerdem kann man hier den Profil löschen oder ausloggen.

Meilenstein 1b: Konzeptioneller Entwurf:



Meilenstein 2: Logischer Entwurf:

Entitäten & Relationen:

Person(PersonID, Password, Email, Firstname, Lastname)
SK: {PersonID}

Is_Friend_Of(SenderId, RecieverId, Status)
SK: {SenderId, RecieverId}
Is_Friend_Of. SenderId ◊ Person
Is_Friend_Of. RecieverId ◊ Person

Destination(DestinationID, Country, City)
SK: {DestinationID}

Category(CategoryID, Description, Icon)
SK: {CategoryId}

Budget(BudgetID, BudgetSOLL, Currency, TripId)
SK: {BudgetID}
Budget.TripId ◊ Trip

Expense(ExpenseID, Description, Price, BudgetID, CategoryId)
SK: {ExpenseID, BudgetID}
Expense.BudgetId ◊ Budget
Expense.CategoryId ◊ Category

Trip(TripId, Color, DateFrom, DateTo, PersonId, DestinationId)
SK: {TripId}
Trip. PersonId ◊ Person
Trip. DestinationId ◊ Destination

Shared_Trip(Description, SuitableFor, TripId, SharedTripId)
SK: {TripId}, {SharedTripId}
Shared_Trip.TripId ◊ Trip

TripSave (PersonId, SharedTripID)
SK: {PersonId, SharedTripID}
TripSave. PersonId ◊ Person
TripSave. SharedTripID ◊ Shared_Trip

TripComment (CommentId, PersonId, SharedTripID, Text)
SK: {CommentId}
TripComment. PersonId ◊ Person
TripComment. SharedTripID ◊ Shared_Trip

Meilenstein 4: Implementierung:

Normalform Überprüfung:

Normalform (1NF): Ich habe sichergestellt, dass alle Eigenschaften in den Tabellen grundlegend und einfach sind. Das bedeutet, sie enthalten nur einzelne Werte und keine wiederholten Gruppen oder Listen.

Normalform (2NF): Ich habe sicher gestellt, dass alle Eigenschaften in den Tabellen vollständig von der gesamten Primärschlüssel abhängen und nicht von Teilen des Schlüssels. Das bedeutet, dass alle Informationen über einen Datensatz durch den gesamten Primärschlüssel identifiziert werden können.

Normalform (3NF): Ich habe sicher gestellt, dass es keine versteckten Abhängigkeiten zwischen den Nicht-Schlüssel-Eigenschaften gibt. Das bedeutet, dass keine Information über ein Nicht-Schlüsselattribut von einem anderen Nicht-Schlüsselattribut abhängt.

Programmierumgebung:

-Visual Studio Code (ich habe mich mit SSH verbunden und alles dadurch programmiert)
-Oracle SQL Developer

Java:

Bei der Generierung der Daten habe ich im allgemeinem das **java.util.Random** benutzt um die Random Zahlen zu generieren. In der Destination und Budget Tabellen Generierung habe ich csv Dateien benutzt. (*worldcities.csv* für Generierung der Destination Tabelle *currency.csv* fürs Mappen der Destination auf die Currency, sodass jedes generierte Budget für den Trip eine der Destination ansprechende Currency hat.)

Für manche Tabellen habe ich selbst einen Array von Werten geschrieben und diese permutiert, wie z.B. in der Tabelle Person, TripComment und Expense.

Kompiliert bzw. ausgeführt habe ich mit den **javac -classpath ojdbc11.jar:. Trip.java** und **java -classpath ojdbc11.jar:. Trip** Befehlen.

Bevor ich tatsächlich mehrere Einträge in die DB eingefügt hätte, habe ich immer mit dem

```
System.out.println(insertSql);
```

überprüft, ob die Schreibweise passt, und ein Eintrag in Oracle überkopiert und ausgeführt damit ich sicher bin, dass es auch funktioniert. Nachdem kommentiere ich die

```
int rowsAffected = stmt.executeUpdate(insertSql);
```

Zeile aus, und führe den Code noch mal aus.

PHP:

Mein PHP Projekt hat folgende Struktur:

- **DatabaseHelper.php :** Datei für die unmittelbare Verbindung zur Datenbank, wo alle Funktionen aufgelistet sind.
- **Index.php**: Landing Page wo man sich in das System einloggt.
- **styles.css**

- **header.php :** beinhaltet die obere Leiste mit den Tabs, die immer gleich bleiben, um Code ein bisschen sauberer zu halten
- **Hilfsdateien *_process.php, submit_*.php, delete_*.php, add_*.php (z.B. Login_process.php)**: Dateien, die POST Methoden bearbeiten und an die ansprechende Funktion in DatabaseHelper weiterleiten.
- **Weitere View *.php Dateien:** die das HTML-Container beinhalten und die Oberfläche abbilden

Für die Anzeige meiner Werte habe ich viele Views benutzt, sodass ich leicht auf die Daten in der lesbaren Form zugreifen und anzeigen kann, in der Weise ich es benötige.

Außerdem befinden in einigen View auch berechenbare Properties wie Times_Shared in **SharedTripView.sql** und BudgetIst in **BudgetView.sql**