

# Dokumentation Einkaufsbuddy

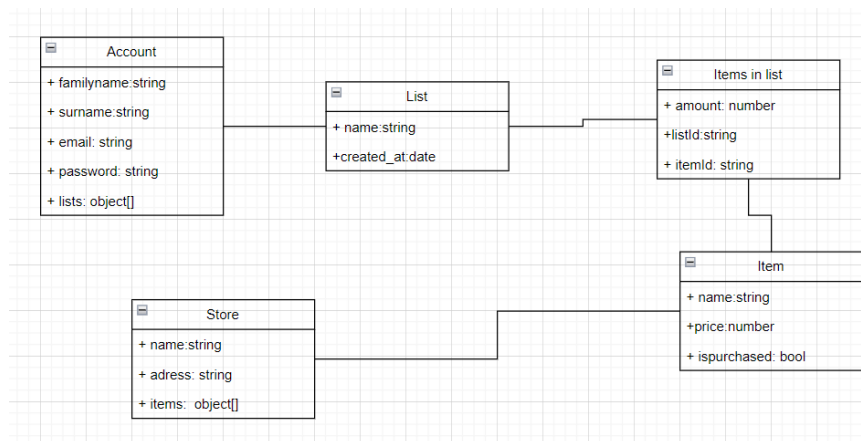
## Eine Online Einkaufslistenverwaltung für den Alltag

### 1. Einleitung

Das Ziel unsere App ist, den täglichen Einkauf zu vereinfachen und weg von dem lästigen Einkaufszettel zu kommen. Dabei kann jeder einzelne Nutzer seine eigenen Einkaufslisten erstellen und nach seinem Interesse gestalten. Auf diese Art und Weise können Einkäufe von Zuhause aus geplant und verhindert werden, das Verlieren der Einkaufsliste für immer.

Nachdem man sich erfolgreich eingeloggt hat, gibt es zwei Optionen. Entweder Artikel suchen oder eine Übersicht über die eigenen Einkaufslisten sehen. Anschliessend können diese eingesehen und verwaltet werden. Wenn die entsprechende Liste ausgewählt wurde, sieht man die enthaltenen Artikel und kann diese verwalten.

### 2. Konzept und Planung



#### Relationen:

Eine Liste gehört zu einem Account, ein Account hat keine, eine oder mehrere Listen, eine Liste hat keine, eine oder mehrere Items, ein Item gehört zu keiner, einer oder mehreren Listen, ein Store hat ein oder mehrere Items, ein Item gehört zu einer oder mehreren Stores.

#### Must have:

- Login
- ein Account kann nur die Listen sehen, die er erstellt hat.
- Datenbank für Items
- Hinzufügen oder entfernen von Listen
- Items zur Liste hinzufügen oder löschen

#### Nice to have:

- Teilen von Listen zwischen den Accounts
- Das Hinzufügen oder Löschen von Läden

### 3. Umsetzung

Wir brauchten Airtable für unser Backend und Postman um Test durchzuführen. Beide haben wir bereits im Unterricht verwendet und deshalb waren wir bereits mit damit vertraut.

Airtable ist benutzerfreundlich, flexibel, anpassungsfähig, erlaubt kollaboratives arbeiten und ist Cloud basiert. Postman ist benutzerfreundlich, effizient, Cloud basiert und Plattformübergreifend. Ausserdem hat Postman automatisierte Tests und API-Monitoring. Das ist eine automatische Überwachung von API-Endpoints zur Erkennung von Problemen. Aufgrund dessen haben wir Airtable und Postman gewählt.

### 4. Ergebnisse

Wir konnten alles machen was geplant war. Jedoch gab es Probleme mit der anzeige von allen Items, diese ist noch nicht komplett optimiert. Da wir nicht genug Zeit dafür hatten, ist dies aktuell eine temporäre Lösung. Deshalb hatten wir jedoch keine Zeit mehr für die «Nice-to-have». Alle Relationen wurden im Code richtig umgesetzt.

Funktionen:

```
async function login() {
  try {
    await loginUser(email.value, password.value);
    await router.push('/');
  } catch (exception) {
    console.error('login error', exception);
    errors.value = exception.errors;
  }
}
```

Funktion die das Login handhabt.

```
async function itemToList(listId, itemName, amountOfItem) {
  try {
    items.value = await fetchItems();
    for await (const element of items.value.records) {
      if (element.fields.Name == itemName) {
        item.value = await fetchItem(element.id);
      }
    }
    console.log("Adding item to list:", listId, item.value.id, amountOfItem);
    await addItemToList(listId[0], item.value.id, amountOfItem);
  } catch (error) {
    console.error(error);
  }
}
```

Funktion die Items zur Liste hinzufügt.

```

async function load() {
  loading.value = true;
  try {
    const listData = await fetchList(id.value);
    list.value = listData.fields;
    console.log('List geladen', list.value.fields);

    items.value = list.value.ItemsInList;

    for await (const element of items.value) {
      const recordData = await fetchList(element);
      record.value = recordData.fields;
      console.log('Record geladen', record.value.Items);

      items.value[items.value.indexOf(element)] = record.value.Items;
    }

    for await (const element of items.value) {
      const itemData = await fetchItem(element);
      record.value = itemData.fields;
      console.log('Record geladen', record.value.Name);

      items.value[items.value.indexOf(element)] = record.value;
    }
  } catch (error) {
    console.error(error);
  } finally {
    loading.value = false;
  }
}

```

Alles relevante rein Laden.

```

async function itemDelete(itemToDel) {
  try {
    await deleteItem(itemToDel[0]);
    load();
  } catch (error) {
    console.error(error);
  }
}

```

Item löschen.

**UID:**

Home



## Login

[Home](#)

# Login

Email

Password

Login

## Listen Übersicht

[Add](#)

# Shopping Lists

[Home](#)

Flynn

EditDelete

test3

EditDelete

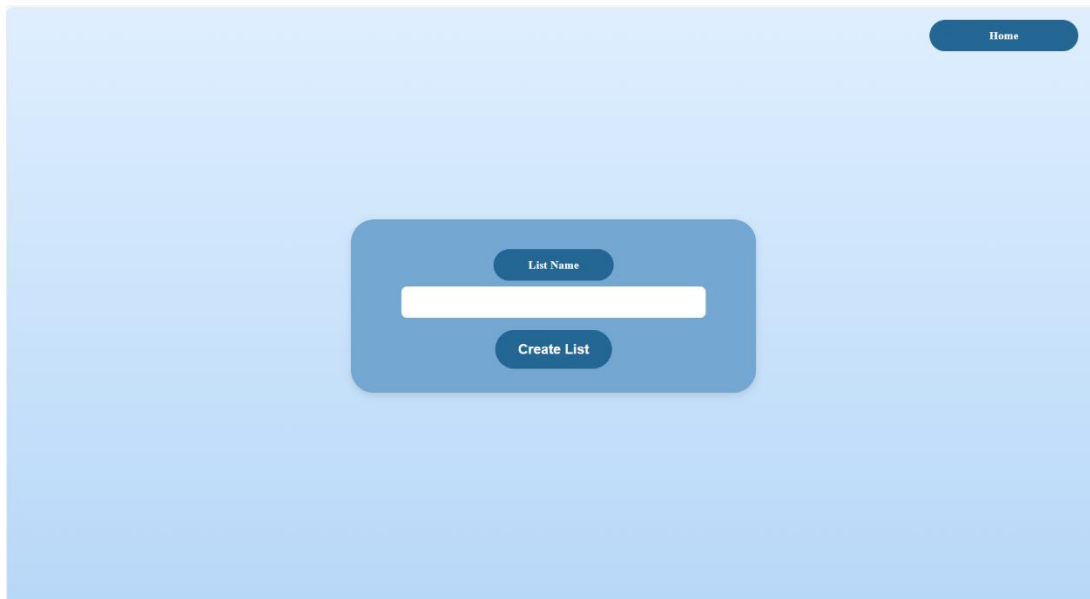
hahsha

EditDelete

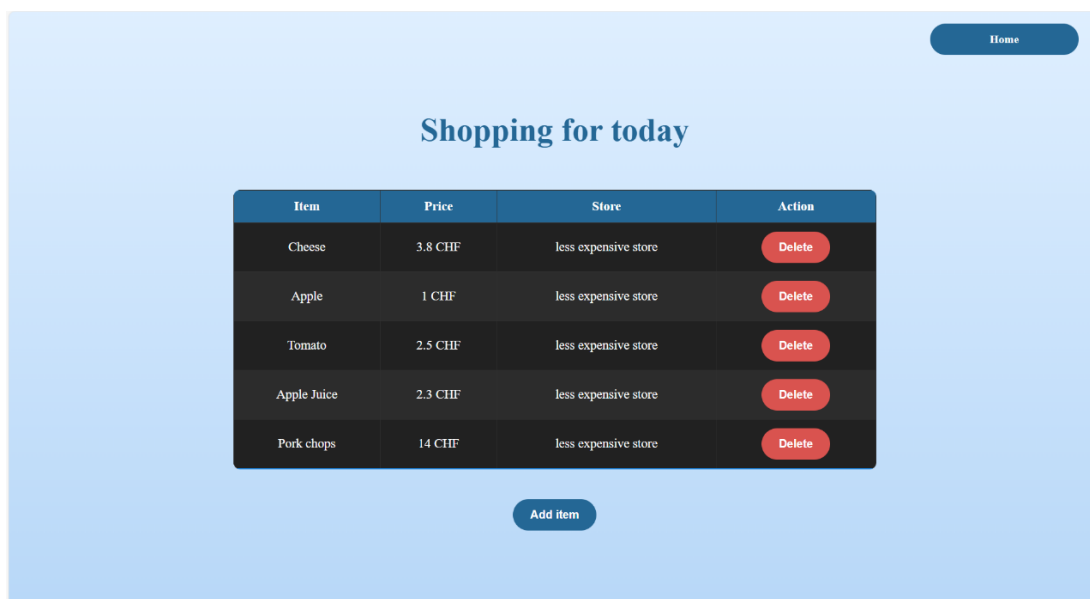
AD

EditDelete

## Liste Hinzufügen



## Liste mit enthaltenen Items



Item	Price	Store	Action
Cheese	3.8 CHF	less expensive store	Delete
Apple	1 CHF	less expensive store	Delete
Tomato	2.5 CHF	less expensive store	Delete
Apple Juice	2.3 CHF	less expensive store	Delete
Pork chops	14 CHF	less expensive store	Delete

## 5. Herausforderungen und Lösungen

Während des Projekts war die Verwendung einer Zwischentabelle das einzige wirklich grosse technische Problem. Zunächst konnten wir dieses Problem durch die Verwendung von 'for await...of'-Loops lösen. Später, als wir herausfanden, dass es Lookup-Felder von Airtable gibt, haben wir diese verwendet. Jedoch haben wir diese erst am Abend der Abgabe entdeckt und deshalb konnten wir aus Zeit technischen Gründe diese nur noch an einem Ort verwenden.

Team bezogen gab es keine Probleme. Ioannis und ich haben bereits mehr als ein Projekt zusammen gemacht und deshalb lief das Teampplay effizient und einwandfrei. Wir kennen gegenseitig unsere Stärken und Schwächen und konnten dadurch die Arbeit gut aufteilen.

## 6. Technische Details

Anleitung:

Zuerst führen Sie im Terminal den Code «npm install» aus. Anschliessend führen Sie «npm run dev» aus und öffnen den Link der erscheint. Nun sind Sie auf dem Startscreen angekommen.

Zuerst müssen Sie sich anmelden. Wenn sich jemand vorher angemeldet hat, werden die Token manchmal nicht richtig gelöscht, so dass Sie die Listen zwar ansehen können, aber keine Listen sehen. Wenn das passiert, müssen Sie sich ausloggen und dann wieder einloggen. Dann können Sie Ihre Listen sehen. Um eine Liste hinzuzufügen, drücken Sie auf den Button "Add" und folgen den Anweisungen.

Um alle verfügbaren Artikel zu sehen, drücken Sie einfach auf Suche nach Lebensmitteln. Keine Anmeldung erforderlich

Damit Sie testen können, haben wir Ihnen einen Account angelegt, dieser lautet wie folgt:

E-Mail: test1@gmail.com

Passwort: 1234

Wir haben ausschliesslich Airtable für das Front- und Backend verwendet. Jedoch zum Testen haben wir zusätzlich Postman verwendet.

## 7. Anhang

Gitlab:

<https://gitlab.com/ipt4.11/einkaufsbuddy>

Verwendete Ressourcen:

[https://airtable.com/invite/l?inviteId=inv3CdeOcXz8KDHwz&inviteToken=1a6174e952f933bf35b19ab163e65a21b718495693a674d5b0340f9c00be9fc7&utm\\_medium=email&utm\\_source=product\\_team&utm\\_content=transactional-alerts](https://airtable.com/invite/l?inviteId=inv3CdeOcXz8KDHwz&inviteToken=1a6174e952f933bf35b19ab163e65a21b718495693a674d5b0340f9c00be9fc7&utm_medium=email&utm_source=product_team&utm_content=transactional-alerts)

Allgemeine Website:

<https://www.airtable.com>

<https://www.postman.com>