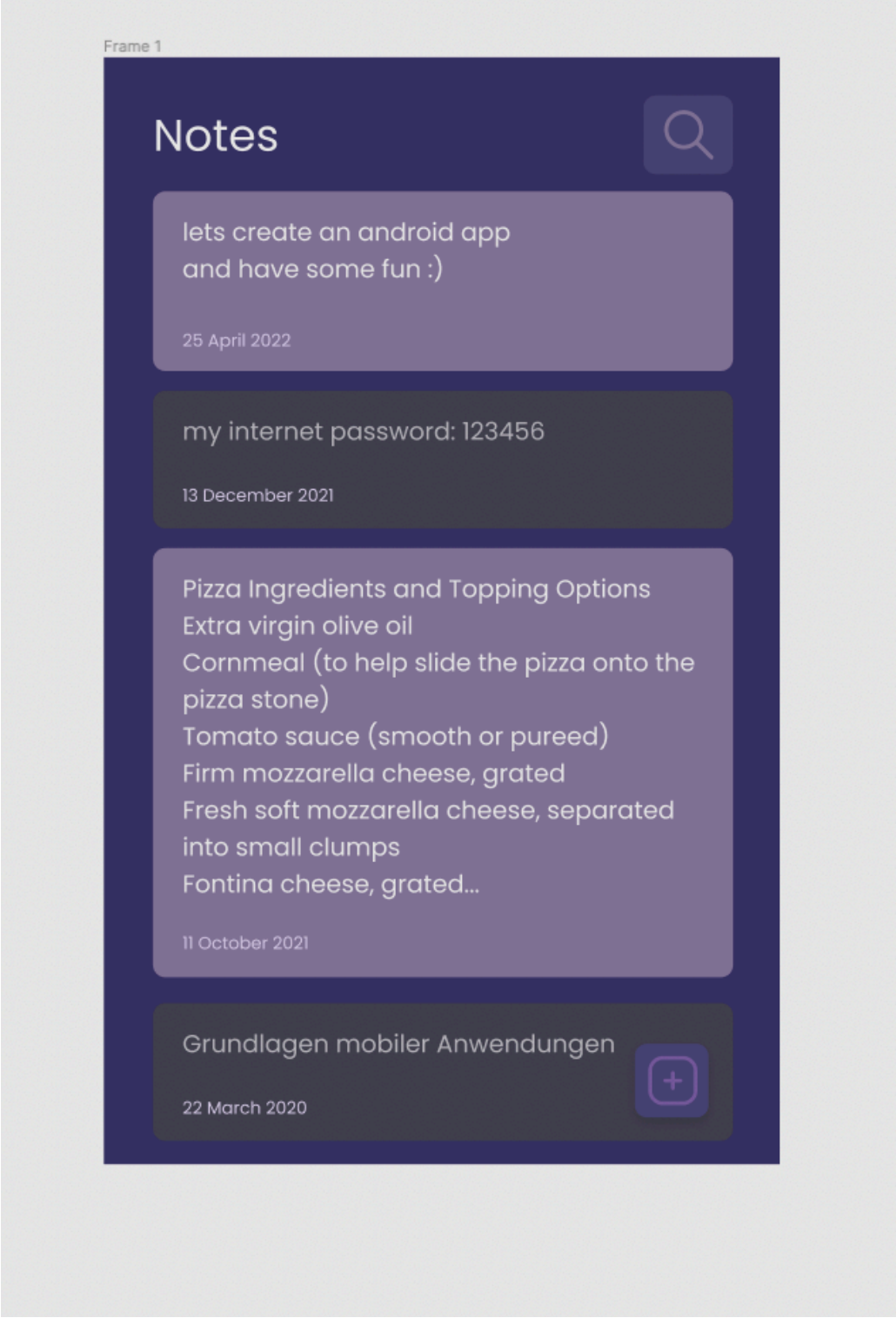
Meilenstein 1

Ich bin an **Verteilte Daten** interessiert und Ich habe mich entschieden ein **Notizen/Notes App** zu entwickeln. Ich werde die Daten Offline speichern und ich werde eine Datenbanksystem (SQLlite vielleicht) benutzen um die Daten zu speichern.

Mein App wird ungefähr so aussehen. Das Design habe mit Figma gemacht. Es kann sein dass ich die Farben noch ändere.



Meilenstein 2

In diesem Teil/Meilenstein 2 werde ich mein App Components bzw. aus welche Components mein App besteht aufteilen/trennen und mein App strukturieren.

Mein App wird aus folgende layers bestehen:

- **View/Frontend/GUI Layer**
  Mein App wird ein GUI haben. Die UI wird ungefähr ein Design haben wie im Meilenstein 1 zu sehen ist.

- **Service/Logic Layer**
  Hier befindet sich die Hintergrund Prozesse. Wie zum Beispiel Daten zu hole oder Daten aktualisieren usw. Oder was soll passieren wenn ein Button geclickt wird.

- **Model/Data Layer**
  Wie und wo sollen die Daten gespeichert werden. Hier geht es um Daten und die Speicherung der Daten. In meinem Fall werden die Daten Lokal(data persistence) gespeichert werden. Ich werde SQLite Datenbank für mein App verwenden.

GUI

Mein App wird ein *Main Activity(1)* haben. Auf dem Main Activity wird die Notizen in form eine liste(Recycler View) zu sehen sein. Oben rechts steht ein Search Button um die Notizen durchzusuchen. Unten rechts steht Plus(+) Button um neue Notizen zu schreiben. Das Button wird eine *neue Activity(2)* öffnen da wo der User einen neuen Notiz schreiben kann. Das App wird auch *ein Activity(3)* haben um die Notizen zu bearbeiten oder die Notizen zu löschen. GUI und Logic werden mit einander Interaktion haben um Tasks zu erledigen.

Logic

Mein App wird ein Logic Layer haben um mit Daten zu arbeiten und die UI funktional zu machen. Die UI wird mit Logic verbunden um Daten/Die Notizen von Local Storage/DB auf dem Activity zu präsentieren. Die Logic layer wird sich darum kümmern um neue Notizen zu speichern, bearbeitete Notizen zu speichern oder Notizen aus dem Datenbank/Storage zu löschen. Es wird sich darum kümmern, wenn der User auf dem Plus(+) clickt dass die Create Note Activity geöffnet wird.

Data

Ich werde Room Database(SQLite) benutzen um die Notizen/Daten Lokal zu speichern. Ich werde eine Tabelle Notizen haben. Die Tabelle wird folgende Spalten haben: ID, Titel, Note, Date.

Interfaces

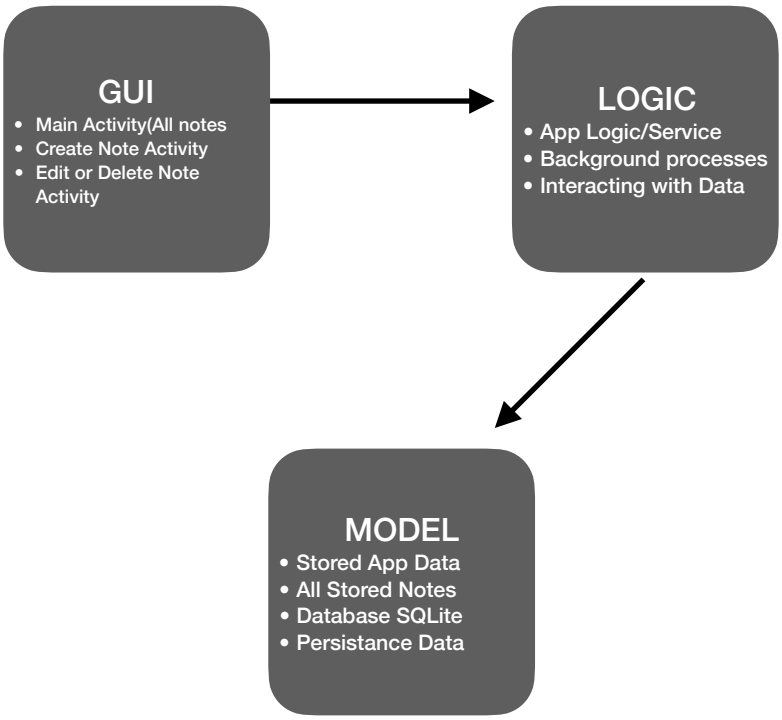
- Note Interface (Java Interface, Service Logic)
  Methoden:
  - Void addNote(String title, String Note)
  - Void editNote(int noteId)
  - List<Note> viewAllNotes()
  - Boolean removeNote(int noteId)

Das Komponent/Klasse Note(Logic) wird mit Model(Datenbank) Interaktion haben.

- Model Interface
  - Connection mit Datenbank
  - SQL Anfragen

Testing

Unit Testing: Ich werde Unit tests machen um meine Java Klassen/Komponenten zu testen bzw. mein App Logic zu testen um zu gucken dass die Methoden richtig funktionieren.
Instrumentation Testing: Mein App Activities/Life cycle werde ich auch mit Hilfe von instrumentation testen, um zu gucken dass die Android Activities richtig funktionieren. Ich werde Emulator und ein echtes Android Smartphone benutzen für diese tests.
UI Testing: Hier wird das Komplette App über UI getestet. Hier wird das App wie in real life getestet zum Beispiel: Notizen wird erstellt, Notizen wird bearbeitet und gelöscht. Die Search Funktion wird auch getestet und Notizen werden gesucht. Alle mögliche Funktionen was ein User benutzen kann.



Meilenstein 3

Notes App

Use Cases:

- **Use case 1:** alle Lokal im Datenbank gespeicherte Notizen zeigen
- **Use case 2:** nach Notizen suchen und die Resultate zeigen
- **Use case 3:** Notizen schreiben
- **Use case 4:** Notizen Lokal(Persistence) speichern
- **Use case 5:** gespeicherte Notizen bearbeiten
- **Use case 6:** Notizen loeschen

Components

View/GUI Components

- **Main View:** Zeigt alle gespeicherte Notizen im form von einer Liste auf dem Display(RecyclerView wird verwendet)
  hat ein Search Bar ganz oben, die Resultate werden auch auf Main View angezeigt.
- **Add View:** um neues Notiz zu schreiben. Form: Title, Message, Save(button)
- **Edit View:** um ein Notiz zu bearbeiten

Espresso Tests:

- Uc1: testen ob alle Notizen auf dem Display angezeigt werden.(Positivtest)
  viewAllNotesTest()
- Uc2: testen ob such Funktion funktioniert, nach Notiz suchen und Resultate überprüfen (Positivtest)
  searchNoteTest()
- Uc3: testen ob Notiz schreiben Activity/Seite funktioniert, neues Notiz schreibe(Positivtest)
  writeNoteTest()
- Uc4: Notiz speichern und überprüfen ob es funktioniert(Positivtest)
  saveNoteTest()
- Uc5: Notiz bearbeiten und wieder speichern und überprüfen (Positivtest)
  editNoteTest()
- Uc6: testen ob Notiz gelöscht wird (Positivtest)
  deleteNoteTest()

Logic/Service Components

- Notes Interface + Notes Implementation
  Methods:
    - viewAllNotes(): List<Note>
      Im Datenbank gespeicherte Notizen von Model/Data holen
    - searchNotes(String text): List<Note>
      Nach Note/Text im Notes Liste suchen und holen/zeigen
    - saveNote(String title, String message): boolean
      Note im Datenbank speichern / an Model weitergeben
    - editNote(int noteId): void
      im Datenbank gespeicherte Note bearbeiten/aendern an Model weitergeben
    - deleteNote(int noteId): boolean
      Note von Datenbank loeschen

JUnit Tests:

- Methode viewAllNotes() testen ob die Liste der Notizen von Datenbank richtig geliefert wird. Positivtest
- Methode searchNotes(text) testen ob die suche funktioniert und die Notizen geliefert werden. Positivtest
- Methode saveNote(title, message) tested ob Notiz gespeichert wird. Positivtest
- Methode editNote(noteId) testen ob Notiz bearbeitet wird und gespeichert wird mit neue Änderung
- Methode deleteNote(noteId) testen ob Notiz erfolgreich gelöscht wird, Positivtest

Es wird getestet, um zu gucken dass alle Methoden korrekt funktionieren.

Model/Data

- Note Model, ich werde Persistance Room Library SQLite verwenden
  Mit dem Datenbank kommunizieren
  - neues Objekt/Item im Datenbank einfuegen(sql insert statement)
    insertData(Note note)
  - Objekte/Items von Datenbank holen(sql select statement)
    getData()
  - Objekt/Item im Datenbank updaten/aendern(sql update statement)
  - Objekt/Item von Datenbank loeschen(sql delete statement)

JUnit Tests:

- Insertion im Datenbank und überprüfen. Positivtest
  insertDataTest()
- Daten von Datenbank holen. Positivtest
  fetchDataTest()
- Daten im Datenbank updaten. Positivtest
  updateDataTest()

- Daten im Datenbank loeschen. Positivtest
  deleteDataTest()

Es wird getestet, um zu gucken dass alle Methoden korrekt funktionieren.

Sequence Diagram