Capstone project

Contents

GitHub Username: Suver1

Tilbud

("Tilbud" translates to "Offers".)

Description

Discover the best offers from hundreds of norwegian shops.
When your favourite online shop has an offer, it's not necessarily a good offer. Powered by an engine logging millions of prices daily, this app gives you a simple overview of the offers worth paying attention to.

Discover - Pin - Buy. On sales like Black Friday and weekly campaigns it's hard to get an overview with the thousands of deals. Not anymore! Just browse through the most popular offers, pin deals that interest you to your own list, then navigate to prisguide.no. There you can take a closer look at the offer before you buy, and compare products and prices.

Intended User

The app is intended for all norwegian consumers.

Features

- Discover thousands of popular offers.
- Save/Pin the offers that interests you.
- Share offers with your friends
- Go directly to Prisguide.no to get more info about the products before you buy your favourite offers.
- Get the newest offers in a widget

User Interface Mocks

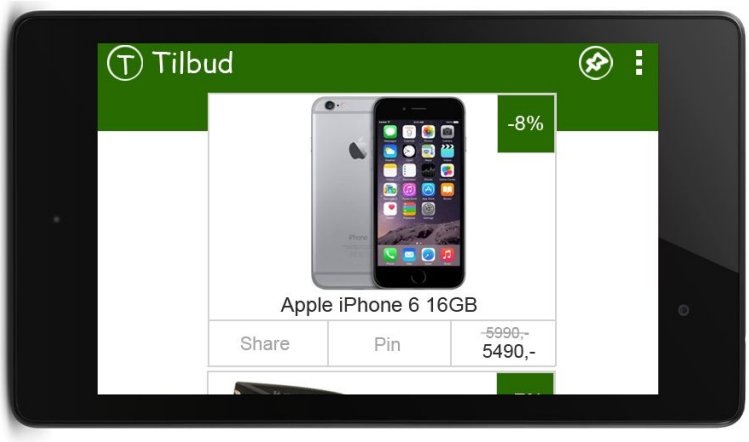
Screen 1

The main screen - displaying the “best” (most popular and most discounted) offers.
Clicking anywhere on the item (picture, discount, price or title) will open the offer in www.prisguide.no, through the default web browser.
“Share” and “Pin” will have a descriptive icon beside the text.
The menu consists of Logo, app name, a pin button (takes user to screen 2) and a overflow button - containing a single option; About - which takes user to an activity describing what the app does (not mocked).

Screen 2

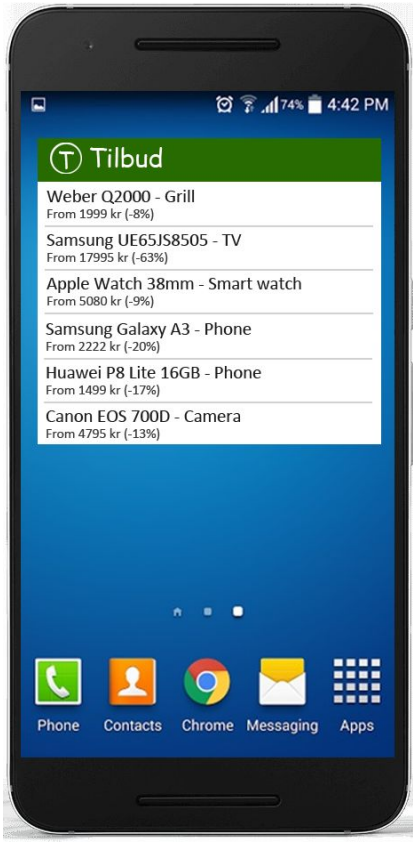
Pin screen - A list of user's saved/pinned offers. Items will behave and look exactly the same as the items on screen 1, except clicking the "Unpin"-button which will remove the item from user's list.

Screen 3



Tablet design

Screen 4



Widget with a list of the latest offers

Key Considerations

Data persistence

All data in the app will be received through an API delivered by Prisguide.no where I work. As offers most often have a time limit and prices change frequently, I will not store any data from the API, except product IDs selected by the user to save.
The app will use an IntentService to pull data from the API. The data will be loaded with a CursorLoader.
I will use a ContentProvider to provide the product IDs saved in the database.
The API data will be gotten with a GET request using the okhttp library. The product IDs will be saved in a SQLiteDatabase.

Images from the API will be cached. All other data from the API will not be cached as product info updates often, and the API has its internal cache system (Varnish) with fast response times. This way the user will spend a bit more data (and battery life) for the price of always getting the latest offers.

Data will be fetched every time the app loads, when the user scrolls to unloaded data (“infinite scroll”), and every time the user navigates to another activity.

UX corner cases

- When the user clicks on an offer, the offer will be opened in the default browser (or browser selection menu).
- In the offers screen, if the user scrolls towards the bottom of the loaded data, new data will be fetched and added the list.
- When data from the API or database loads, a loading icon will be displayed.
- All errors related to fetching data or connecting to the internet will be displayed with a Toast, conveying what action the user should take to solve the issue.
- If the user navigates to the pinned list and the list is empty, an empty item with a relevant message will be displayed.
- In the pinned list, if the user unpins an offer, the offer-item will be removed from the list. If there are no more items in the list, the empty item will be displayed.
- In the pinned list, if an offer has expired, the offer badge and discount price will be removed from the item. The user will have to unpin the product manually.
- When the user navigates to the pinned list and then back to the offers list, the offer-list position will NOT be remembered. The user will start at the top, displaying the newest and most popular offers.

Libraries

I will use okhttp to handle GET requests (fetching data from the API) and Retrofit to simplify using the REST API.

I will use Picasso to load and cache images.

Required Tasks

User experience recap

The App will:

- Upon launch, present the user with a list of popular offers
- Allow the user to tap on an offer to open the offer in Prisguide.no in the users default browser
- Allow the user to pin/save offers to a custom list
    - Products will be saved in the database
    - Prices and discount badge must be updated for expired offers
- When user navigates to the saved offers via the action bar, the saved offers is displayed
- Allow user to read about the app through an overflow menu

The widget will:

- Display the latest offers
- Allow the user to click on an offer and be taken directly to the web
- Allow the user to open the app by clicking the logo

Task 1: Project Setup

- Setup a clean Android Studio project
    - Support minimum Android 4.0
- Set up the github repository
    - Create a README and git ignore file
    - Commit the new Android studio project to github
- Configure build.gradle to include the following libraries
    - Picasso
    - Okhttp
    - Retrofit
    - Material design, Appcombat v7 etc.
- Prepare API
    - Add required permissions to manifest
    - Check that the API works as intended with a unique API-key for the project
    - Make sure that the API-key won't be committed to github
- Setup design resources
    - Create primary and accent colors
    - Add logo and icons to drawables
    - Add texts to strings.xml
    - For UI testing:
        - Add strings with dummy texts, like prices, discount, product titles
        - Add dummy pictures to drawables

Task 2: Implement UI for each activity and fragment

- Build UI for OffersActivity's ActionBar
    - Use Material design (App bar)
    - Elements:
        - Logo
        - Title
        - Pin icon
        - Overflow menu
- Build UI offer item
    - GridLayout
    - Item elements:
        - Image
        - Discount badge

        - Product title
        - Share button
        - Pin button
        - Prices (before, now)
- Build UI for SavedOffersFragment / SavedOffersActivity and the OffersFragment / OffersActivity
    - Fragment - Starting point for ListView
    - Up Action for SavedOffersActivity's ActionBar
- Build UI for AboutOffersActivity
    - Up Action
    - Simple TextView

Task 3: Display some data

Create the following classes and display some dummy data

- OffersActivity (MainActivity, extends AppCompatActivity)
- OffersFragment
- SavedOffersActivity (extends AppCompatActivity)
- SavedOffersFragment
- AboutActivity (extends AppCompatActivity)

Task 4: Interaction

Set up the foundation for all clickable elements

- Saving offer
- Navigation to the offer in the default web browser
- All menu navigations
    - Saved offers
    - Overflow menu
    - Up actions

Task 5: Handle API calls

- Create classes that uses okhttp and retrofit to fetch and handle API calls and responses.
    - Adapter
    - Interface
    - Getter Setter Class
- Do error checking

Task 6: Request and display API data

- Set up an IntentService to call the API asynchronously
- Initialize the service from a Loader in OffersActivity

Task 7: Add database support

- Contract
- DbHelper (extends SQLiteOpenHelper)
- Provider (extends ContentProvider)

Task 8: Saving offers

- Call the ContentProvider to save selected Products (IDs)
- Mark item as saved

Task 9: Displaying saved offers

- Set up an IntentService to get offers from the API with the product IDs in the database via ContentProvider
- Use a LoaderManager and CursorLoader to fetch the product IDS
- Populate the result into the listView

Task 10: Handle deleting saved offers

- Call the ContentProvider to delete product IDs from the database
- Remove item from list

Task 11: Handle screen rotation

- Make sure screen rotation works in all activities/fragments

Task 12: Support tablet

- Build tablet UI
- Make sure appropriate image sizes are loaded

Task 13: Create widget

- Build widget UI (ListView widget)
    - Widget ActionBar
    - List item (CollectionList)
- Create a WidgetDataProvider and WidgetProvider (extends AppWidgetProvider) class
- Create a WidgetService (extends RemoteViewsService) to fetch API data and set up API syncing at a fixed interval

Task 14: Support for accessibility

- Make sure content descriptions are in place and working properly
- Optimize for navigating with a D-pad

Task 15: Google Analytics

- Implement tracking for views and clicks to web

Task 16: Admob

- Display a fullscreen ad (test) when user navigates to SavedOffersActivity

Task 17: Animations

- Display a simple transition when navigating between activities

Task 18: Build and deployment

- Make sure the app builds and deploys using the installRelease Gradle task.
- Set up a signing configuration