GitHub Username: Pascal Dierich (https://github.com/PascalDierich)

Watchdog

Description

Watchdog lets you observe your most important social profiles. You can create an 'Observable' profile and add their social accounts to never miss a post again.
Currently there's only YouTube supported, but the architecture let's you implement new ones easily.

Intended User

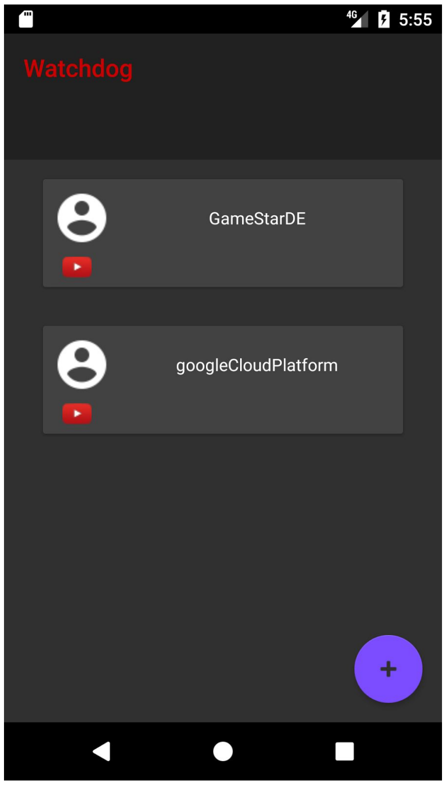
This App is for everyone who wants to pay extra attention to one of his social contacts. Who never want to miss a post but also don't have the time to check for it every 5 minutes.

Features

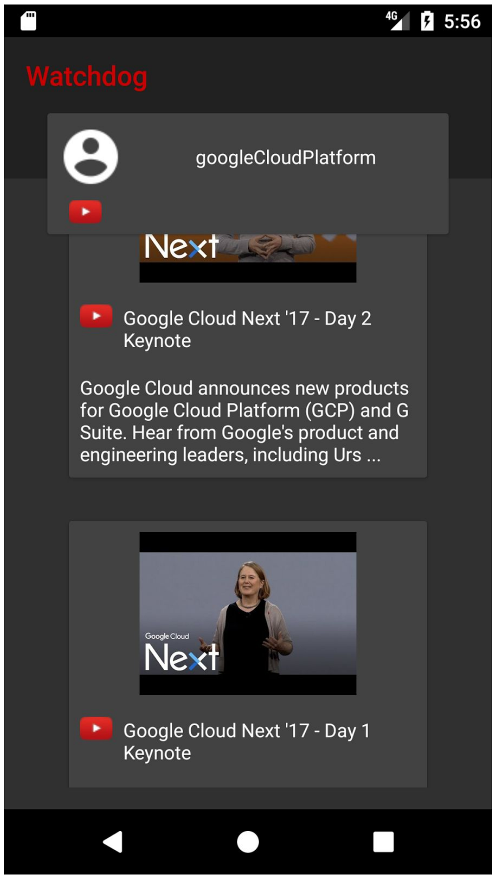
- Create a profile for an observable and let the app check for his social networks (currently only YouTube is supported)
- Get to the Posts directly per notification and check them out on their social App or website

User Interface Mocks

MainActivity



PostActivity


5:56

Watchdog

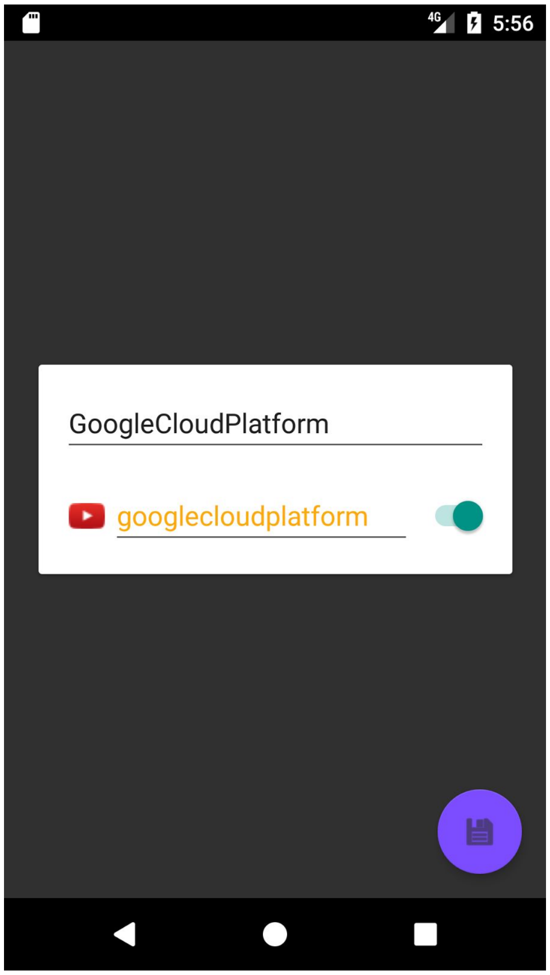
googleCloudPlatform

Google Cloud Next '17 - Day 2 Keynote

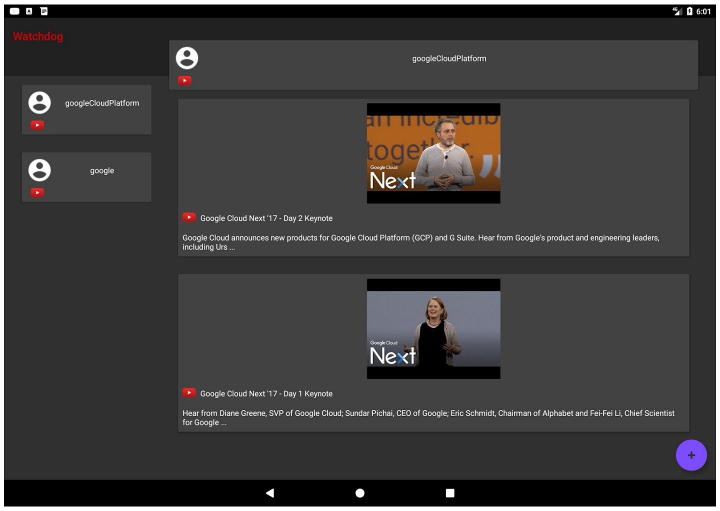
Google Cloud announces new products for Google Cloud Platform (GCP) and G Suite. Hear from Google's product and engineering leaders, including Urs ...

Google Cloud Next '17 - Day 1 Keynote

SetObservableActivity



Tablet



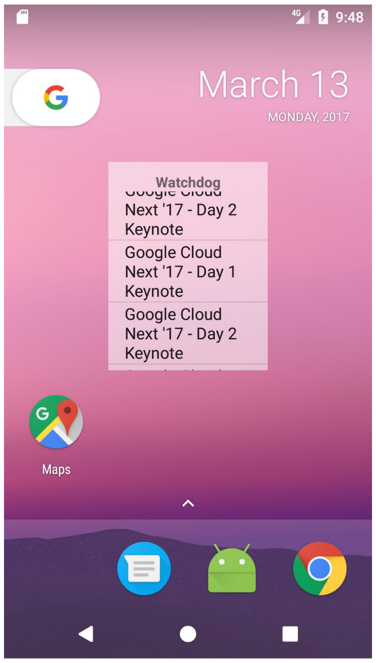
Google Cloud Next '17 - Day 2 Keynote

Google Cloud announces new products for Google Cloud Platform (GCP) and G Suite. Hear from Google's product and engineering leaders, including Urs ...

Google Cloud Next '17 - Day 1 Keynote

Hear from Diane Greene, SVP of Google Cloud; Sundar Pichai, CEO of Google; Eric Schmidt, Chairman of Alphabet and Fei-Fei Li, Chief Scientist for Google ...

Widget



Google Cloud Next '17 - Day 2 Keynote

Google Cloud Next '17 - Day 1 Keynote

Google Cloud Next '17 - Day 2 Keynote

Key Considerations

How will your app handle data persistence?

App will store downloaded information about posts in local SQLite database.
The saved profiles (Observables) will get stored in a SQLite table too.

Describe any corner cases in the UX.

Give user feedback about entered YouTube name.
Coordinate Tablet layout with two pane mode.

Describe any libraries you'll be using and share your reasoning for including them.

- This app will use Retrofit in the Data Module for accessing APIs.
- Butterknife for more readable UI Code.
- Android Support Annotations for easier development
- Picasso for Image handling

Describe how you will implement Google Play Services.

Firebase Crash Report: to improve UX
Firebase Cloud Messaging: to have an open connection to the user

Next Steps: Required Tasks

Task 1: Project Setup

There will be 3 tasks for setup:

1) Setup the Data Android Module (Data Layer)
- Define Repositories (Boundaries) for communication between Data and Domain Layer

2) Setup the Domain Java Library (Domain Layer)

- Define Interactors (Use Cases) for communication between Domain and Presentation Layer

3) Setup the Presentation Android Module (Presentation Layer)
- Setup abstract Base UI Presenter as Interface of all UI Presenter Interfaces
- Setup abstract Base Presenter as Superclass of all Presenter

Task 2: Implement Network Data Package (Data Layer)

- Setup Retrofit to access APIs
- Couple Retrofit Interfaces with Repository Classes

Task 3: Implement Local Data Package (Data Layer)

- Setup local database
- Define database schema
- Create DBHelper
- Provide ContentProvider

- Couple ContentResolver with Repository Classes
    - Network methods will run in workerthread
    - The Interface will provide CursorLoader to access locally stored data.
        - The Cursor-object will get converted to a POJO

Task 4: Create Interactors (Domain Layer)

- Create Interactors (as POJOs)
    - Create a SyncAdapter to check Servers in regular intervals
    - Run Interactors in workerthread when performing network-tasks
- Couple Interactors with Repositories to access Data Layer

Task 5: Define UI for Each Activity and Fragment

- Create Layouts for every Activity / Fragment
- Define UI interactions
- Create a widget for homescreen:
    - Create a widget layout in res/layout and widgetprovider.xml specification
    - Create a RemoteViewsService and Factory as dataProvider
    - Declare Service and WidgetProvider in AndroidManifest.xml

Task 6: Create Presenter-View Interfaces

- Define Callback Interfaces for each Presenter
- Let them implement Base UI Presenter

Task 7: Create Abstract-View Presenters

- Create for each UI Presenter an abstract Presenter as Superclass
- Let each abstract Presenter extends Base Presenter to let them run on Background Thread

Task 8: Define Layouts for each social Network

- Create a layout for each possible Post to include them in the ‘News Feed’