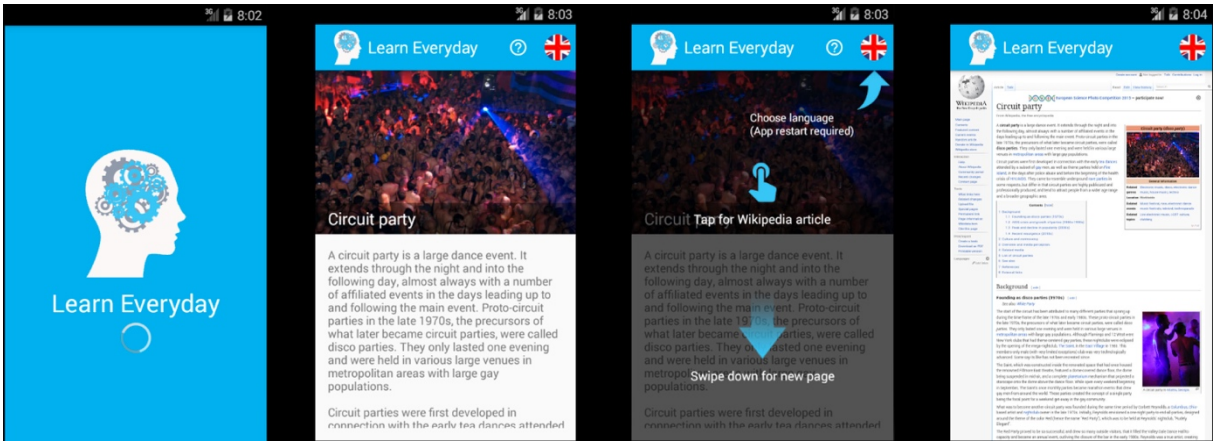
Mappeinnlevering 3 – Apputvikling

Studentnr s198579

Showcase av ferdig produkt:



Ideén

Ideén gikk ut på å lage en applikasjon som henter tilfeldige artikler fra Wikipedia sitt API via. JSON, og presenterer dem på en pen måte, hvor en enkelt kan bla videre til nye artikler, og lære nye nytt hver dag. Derav navnet "Learn Everyday".

Funksjonalitet

Splashscreen

Intro skjerm som vises under oppstart av appen. Har som formål å klargjøre 10 artikler fra Wikipedia sitt API før en blir videreført til hovedaktivitet.
En progressbar indikerer ovenfor bruker at appen laster.

Hovedaktivitet

Viser hovedbilde, tittel, og intro-tekst fra artikkel. Dersom teksten er for lang til å vises, så kan en scrolle nedover. Sveip nedover for å laste en ny artikkel.
Appen har en actionbar med app-ikon, hjelp, og valg av språk.

Jeg har valgt å ikke ha en egen tilbakeknapp, da dette ikke er anbefalt praksis av Google. Anbefalt praksis er å benytte telefonens egen tilbakeknapp for bakover-navigering.

Ref: http://developer.android.com/training/implementing-navigation/temporal.html

Hjelp-skjerm

Et fragment som har til formål å vise brukeren hvordan bruke appen, og forskjellig funksjonalitet appen stiller med. Forsøkt å formidle dette på en brukervennlig og intuitiv måte via. ikoner og tekst. Fragmentet vises ved å trykke på "?" ikonet i actionbar, og kan fjernes igjen ved å trykke på samme knapp, eller hvilket som helst sted på skjermen.

Web-view

Et webview som viser aktuell Wikipedia artikkel i sin helhet.
Kan navigere til denne ved å trykke på bilde eller tittel i hovedaktiviteten.
Støtter zoom-funksjonalitet, via. såkalt "pinch to zoom", "double tap", eller via. zoom in/ut knapper.

Språkbytte

Appen støtter engelsk, tysk, fransk, og norsk, og følger språket valgt på telefonen. Ikon oppe i høyre hjørne av actionbar viser gjeldende språk.
Ved trykk på dette ikonet/flagget, så blir man navigert til språkinnstillinger på telefonen, og kan enkelt bytte språk her. Standardspråk er engelsk.

Appen henter artikler fra Wikipedia versjon av det valgte språket.
Dvs. er engelsk valgt, så henter den engelske wikipedia artikler. Er norsk valgt, så henter den norske wikipedia artikler osv.

Tilstandsbevaring

Appen støtter tilstandsbevaring. Eksempler på dette:

- Husker hvilken side man var på dersom appen går i bakgrunnen, og hentes fram igjen.

- Splashscreen og klargjøring av artikler gjøres kun dersom det ikke allerede er gjort. Dvs. kun når appen startes opp fra scratch.

Landskap-modus

Appen støtter landskapsmodus ved en alternativ layout, som er bedre tilpasset horisontal visning.

Tredjepartsbiblioteker / rammeverk

Jeg har benyttet med av 3 stk. tredjepartsbiblioteker for animasjoner, JSON parsing, og bildevisning. Dette gir meg mulighet til å skrive kortere, enklere, og mer robust kode for kompleks funksjonalitet.

GSON
https://github.com/google/gson

Google sitt eget bibliotek for parsing av JSON, og har følgende mål:

- Provide simple toJson() and fromJson() methods to convert Java objects to JSON and vice-versa

- Allow pre-existing unmodifiable objects to be converted to and from JSON extensive support of Java Generics

- Allow custom representations for objects

- Support arbitrarily complex objects (with deep inheritance hierarchies and extensive use of generic types)

Picasso
http://square.github.io/picasso/

Et populært bildebibliotek for Android, som forenkler arbeid med bildebehandling og bildevisning, og har følgende mål:

- Handling ImageView recycling and download cancelation in an adapter.
- Complex image transformations with minimal memory use.
- Automatic memory and disk caching.

Dette biblioteket gjør det mulig for meg å enkelt håndtere evnt. exceptions ved forsøk på å vise bilde fra URL i et ImageView, samt kunne benytte med av diverse callback-funksjoner for å bla. determere hvorvidt bildet er ferdig nedlastet mm.

EasyAndroidAnimations
https://android-arsenal.com/details/1/940

Animasjonsbibliotek med et flust av animasjonsmuligheter.
Gjør det enkelt å animere views, alt fra enkle animasjoner til mer komplekse sammensatte animasjoner.

Benyttes i hovedsak her for å animere sveip til ny side/artikkel.

Ressurser

https://www.google.com/design/icons/

Alle ikoner er hentet fra Google sin egen nettside for android ikoner, og modifisert med egne farger. Det er ikoner for alle android sine størrelses-klasser (Dvs. mdpi, hdpi, xhdpi, xxhdpi, og xxxhdpi), slik at skalering mellom skjermstørrelser blir korrekt.

Logo er hentet fra nettet, og blitt tungt modifisert og tilpasset appen, for å unngå evnt. copyright problemer.

Gjennomgang av kode

Page.java

POJO klasse for representasjon av en wikipedia artikkel.

Splashscreen.java

```
private void isSplashShown()
```

Sjekker hvorvidt splashscreen allerede er vist, og i såfall går videre til MainActivity.

```
private boolean checkNetworkState()
```

Sjekker hvorvidt man er koblet til internett eller ei.
Hvis ikke, så vises det en alert-dialog med beskjed om at en må være tilkoblet internett for å kunne bruke denne appen.

```
private class FetchDataDuringSplash
```

Privat klasse som extender AsyncTask, og har som formål å hente og klargjøre 5 wikipedia artikler i en separat tråd under visning av splashscreen.

```
private void visAlert()
```

Metode som viser en alert-dialog med beskjed om at man ikke er koblet til internett. Ved trykk på "Ok", så avsluttes deretter appen.

Contentfactory.java

Statisk klasse med formål å hente artikler fra wikipedia, parse JSON, og opprette artikler i en ArrayList med Page-objekter.
Inneholder også en static string som inneholder språkkoden for valgt språk på telefonen.

Har gjort klassen statisk her, da den ikke skal representere noe objekt, og har kun som formål å hente data fra Wikipedia sitt API.
Ved å gjøre klassen statisk, så kan jeg også enkelt få tilgang på dens attributter direkte fra andre klasser.

```
public static void addPage()
```

Kjører API kall mot wikipedia for å finne en ny side med bilde.
Fortsetter fram til den finner en artikkel med bilde via. en while løkke.

Siden det ikke er mulig å hente både tittel, innhold, og bilde i et og samme API kall, så gjøres det 2 kall:

- Et for å hente tittel, page ID, og innhold
- Et for å hente bilde URL til hovedbilde basert på page ID.

Benytter en BufferedReader sammen med en HttpConnection for å utføre API kallet. Jeg benytter deretter en StringBuilder for å bygge strengen av JSON. Benytter en StringBuilder her da det er mindre ressurskrevende enn å konkatenere strenger.

Traverserer meg så nedover i JSON objekt hierarkiet for å hente de aktuelle objektene jeg er ute etter. Da navnet til det ene objektet (page id) vil variere mellom hvert enkelt kall, så benytter jeg et Map som jeg deretter traverserer igjennom for å få tak i de aktuelle objektene.

Gjør deretter det samme for å hente bilde URL med et separat API kall basert på Page ID funnet i det første API kallet.

MainActivity.java

Applikasjonens hovedklasse om du vil. Mest aktuelt å snakke om her er bruken av SwipeRefreshLayout, som er en relativt ny layout fra Android sitt support library. Den brukes for en standard implementering av det typiske " Pull to Refresh" mønsteret på Android og andre smarttelefon OS. Google skriver:

"The SwipeRefreshLayout should be used whenever the user can refresh the contents of a view via a vertical swipe gesture".

Layouten passer sådan ypperlig for denne applikasjonens formål, og gir brukeren enkelt mulighet til å oppdatere innhold ved å sveipe ned med fingeren.

```
private void updateContent()
```

Oppdaterer tittel, bilde og innhold ved å hente data fra ArrayList<Page> i Contentfactory.java for gjeldende side. Setter deretter progressbar for bilde synlig, og fjerner den igjen når bilde er ferdig lastet.

Bilde lastes ved hjelp av Picasso metoder. Dersom det skulle kastes en exception av typen IOException i forbindelse med lasting av bildet, så vises et placeholder bilde i stedet.

```
private void showHelpFragment()
```

Viser eller skjuler hjelp-fragmentet, alt etter hvorvidt det allerede er synlig eller ei.

```
public boolean onTouchEvent(MotionEvent event)
```

Skjuler hjelp-fragmentet dersom det allerede synes, og brukeren trykker på skjermen.

```
private class FetchNewData
```

AsyncTask klasse som henter en ny artikkel hver gang brukeren sveiper til en ny side. På denne måten vil det alltid være en ny side tilgjengelig før brukeren i det hele tatt navigerer til den.

```
private void setUpListeners()
```

Setter opp lyttere for trykk på bilde og SwipeRefreshLayout.
I lytteren for bilde, så skjules hjelp-fragmentet dersom det er synlig, hvis ikke så går man videre til WebView aktiviteten, og sender med page ID for artikkelen WebView skal laste.

I lytteren for SwipeRefreshLayout, så gis det beskjed til view'et at data lastes ved hjelp av *swipeView.setRefreshing(true),* og deretter igjen når det er data er ferdig lastet ved *swipeView.setRefreshing(true).* Dette er nødvendig for at SwipeRefreshLayout skal fungere som forventet. Til slutt så animeres overgangen til ny side ved hjelp av en *FlipVerticalAnimation***.**

WebActivity.java

Klasse som har som formål å vise et WebView med en wikipedia artikkel. Inneholder et WebView, og en pageID som sier hvilken side view'et skal laste. PageID hentes ut fra Intent'et som sendes fra MainActivity.

Utfordringer

De største utfordringene lå i å sette seg inn i Wikipedia sitt API, samt GSON og jobben med å få parset JSON resultatene fra Wikipedia, og det ble brukt 2 uker til dette alene.

Andre utfordringer har ligget i å sette seg inn i Splashscreen funskjonalitet, og hvordan implementere dette, samt SwipeRefreshLayout. Det ble forsøkt flere muligheter her, før jeg falt på valget av SwipeRefreshLayout som en robust løsning for navigering.

Bugs

Av kjente bugs som jeg ikke har rukket å fått fikse finner vi følgende:

- Dersom en blar nedover i teksten, og går over til en ny side, så forsvinner ofte teksten fra TextViewet på den nye siden. Den kommer tilbake så snart man forsøker å scrolle, eller gjøre noen gesture.

- Dersom en blar lenge nok til nye sider, så dukker sider iblant opp 2 ganger etter hverandre.

- Dersom en blar lenge nok til nye sider, så slutter de fleste bilder å vises i ImageView. Dvs, hverken placeholder bilde eller artikkelbilde vises lenger.