

# Simple News App

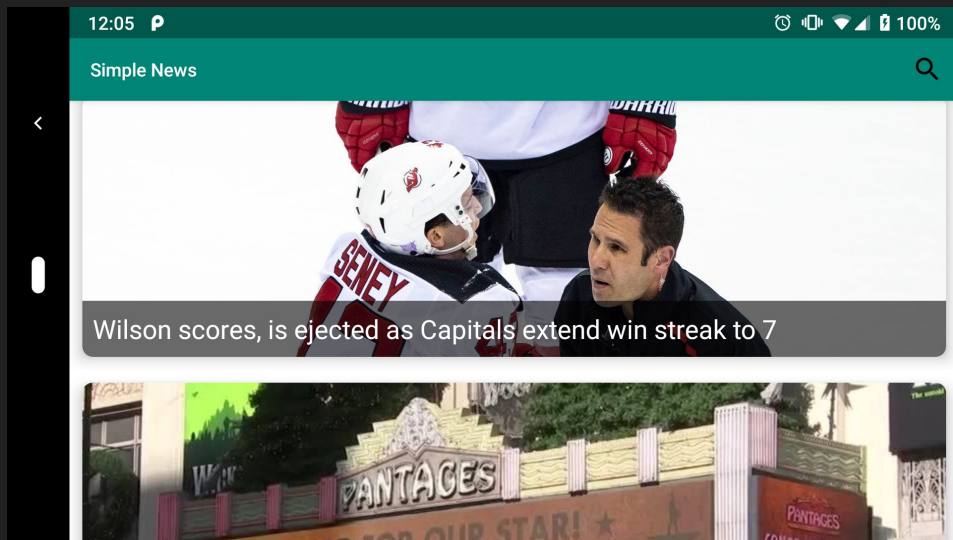
PROG39402 - Final Project

# What is Simple News

Simple News is an application that displays a list of recent news articles for the user to view at their leisure.

Screens

# Article Feed




# Article Details

12:05 P

🕒 📶 🔋 100%

← Article



## Wilson scores, is ejected as Capitals extend win streak to 7

By: AP

Tom Wilson scored again before being ejected, Nicklas Backstrom had a hat trick and the Washington Capitals beat the New Jersey Devils 6-3 to extend the defending Stanley Cup champions' winning streak to seven games


Published: 2018-12-01

View Article

12:05 P

🕒 📶 🔋 100%

← Article



## Wilson scores, is ejected as Capitals extend win streak to 7

By: AP

Tom Wilson scored again before being ejected, Nicklas Backstrom had a hat trick and the Washington Capitals beat the New Jersey Devils 6-3 to extend the defending Stanley Cup champions' winning streak to seven games

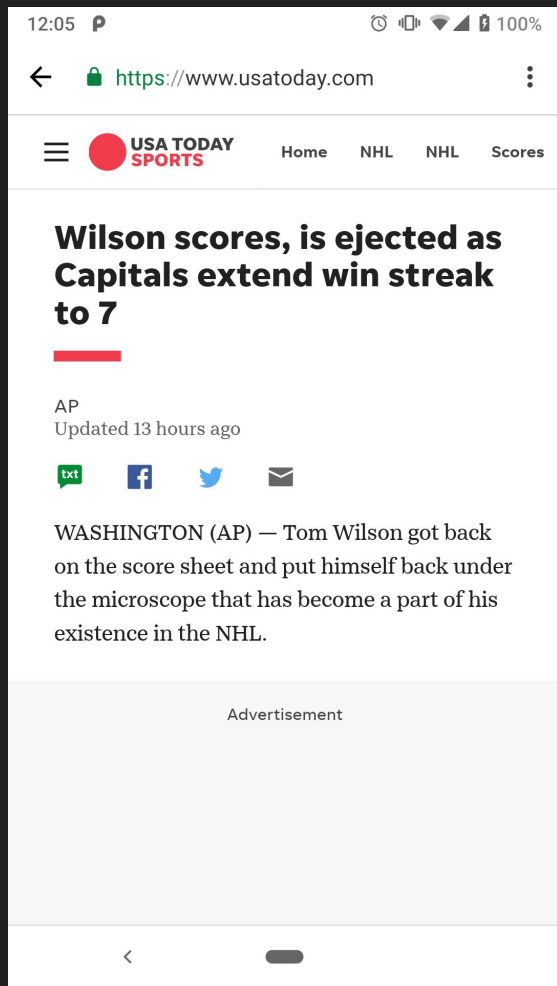
Published: 2018-12-01

View Article

# Chrome CustomTabs

Relatively new way to display webpages in android applications

- Highly performant
- Customizable
  - Up Button
  - Toolbar Color
  - Overflow Menu



# Technologies/Libraries Used



# Kotlin



Android



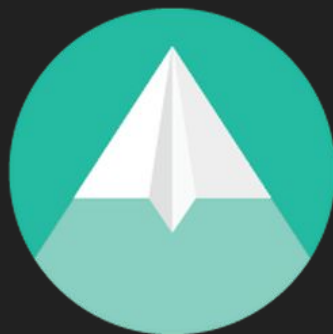
Retrofit



OkHttp



Dagger 2



glide





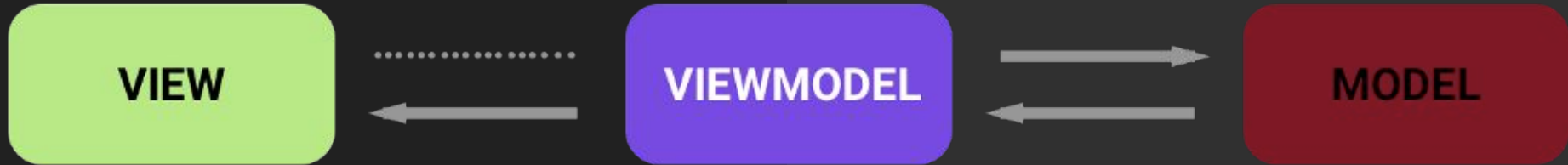
# Model View ViewModel

The entire application follows the  
Model View ViewModel (MVVM)  
Design pattern

Increased Maintainability

Testability

Extensibility



# Remote Data

All data used in the application is retrieved from a third party api

Retrofit2, the HTTP client for Android and Java was used to simplify network requests

API: [Currents API](#)

```
interface NewsRetrofit {  
    companion object {  
        const val TAG = "NewsRetrofit"  
  
        const val BASE_URL = "https://api.currentsapi.services/v1/"  
  
        const val QUERY_START_DATE = "start_date"  
        const val QUERY_END_DATE = "end_date"  
        const val QUERY_LANGUAGE = "language"  
    }  
  
    @GET(value: "latest-news/")  
    fun latestNews(): Deferred<Response<News>>  
  
    @GET(value: "search/")  
    fun getWithQuery(  
        @Query(QUERY_START_DATE) start: String = DateUtil.weekAgo.toRFC3339(),  
        @Query(QUERY_END_DATE) end: String = DateUtil.today.toRFC3339(),  
        @Query(QUERY_LANGUAGE) language: String = "en"  
    ): Deferred<Response<News>>  
}
```

```
private suspend fun getAllFromRemote(): List<RoomArticle> {  
    val request = remote.getWithQuery()  
    val response = request.await()  
    if (!response.isSuccessful || !response.body()?.status.equals("ok")) {  
        return emptyList()  
    }  
    return response.body()?.news?.map { article ->  
        RoomArticle(article.url, article.author, article.description, article.image)  
    } ?: emptyList()  
}
```

# Local Data

All data was saved with the Room Persistence Library

```
@Entity
data class RoomArticle(
    @PrimaryKey(autoGenerate = false)
    var url: String,
    var author: String = "Anonymous",
    var description: String,
    var image: String,
    var language: String,
    var published: Calendar?,
    var title: String
) : Comparable<Article> {
    override fun compareTo(other: Article): Int {
        return this.url.compareTo(other.url, ignoreCase: false)
    }
}
```

```
@Dao
abstract class ArticleDAO : BaseDao<RoomArticle>() {

    @Query( value: "SELECT * FROM RoomArticle ORDER BY published DESC")
    abstract fun allArticles(): LiveData<List<RoomArticle>>

    @Query( value: "SELECT * FROM RoomArticle WHERE url=:url")
    abstract fun articleById(url: String): LiveData<RoomArticle?>

    @Query( value: "SELECT * FROM RoomArticle WHERE title LIKE :title ORDER BY published DESC")
    abstract fun searchArticlesByTitle(title: String): LiveData<List<RoomArticle>>

    @Query( value: "DELETE FROM RoomArticle WHERE published < :cutOff")
    abstract fun trimByDate(cutOff: Calendar = Calendar.getInstance().apply { add(Calendar.DAY_OF_YEAR, amount: -7) })

}
```

# Dependency Injection

Dependency Injection is used to generate the required dependencies throughout the application

From Activities and Fragments to Repository and DAOs everything is injected seamlessly

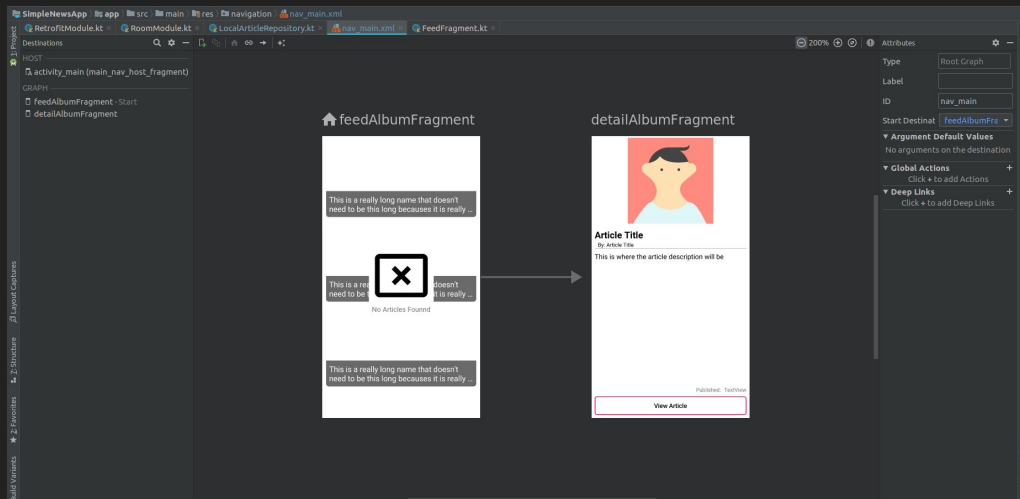
```
@Module
class RoomModule {

    @Provides
    @Singleton
    fun provideAppDatabase(context: Context): AppDatabase {
        return Room.databaseBuilder(context, AppDatabase::class.java, AppDatabase.DATABASE_NAME).build()
    }

    @Provides
    @Singleton
    fun provideArticleDao(appDatabase: AppDatabase): ArticleDAO {
        return appDatabase.articleDao()
    }
}
```

```
class LocalArticleRepository @Inject constructor(
    private val articleDAO: ArticleDAO
) {
```

# Android Architecture Component Navigation



Navigation between different fragments is done using the Navigation component included Android Jetpack

# Comprehensive List of Libraries Used

- [Kotlin](#)
- [Kotlin Coroutines](#)
- [Android KTX](#)
- Android Material Library
- [Retrofit2](#)
- [Retrofit2 Coroutine Call Adapter](#)
- [GSON](#)
- [Navigation](#)
- [Dagger2](#)
- [Room](#)

- [Glide](#)
- [Glide Transformations](#)

Questions

# Quick Demo



End