Update list UI & View Model

1. Display name of the author
    1. Get using authorRepository.getAuthor(author_id)
2. Display image using imageUrlString of the Author
    1. Get using authorRepository.getAuthor(author_id)
3. Display human readable version of “updatedAt”
    1. Add the param to Article data model

Tap to open article

1. When article is clicked, we want to display article. This is basically wiring. In the list view, onClick will take the article object and pass it to ArticleScreen(articleObject).
2. Impl:
    1. When article is clicked, the article should be navigated to with the args.
    2. On the list view, we want the click handler to setup all this.
        1. Clickable = { navController.navigateToArticle(article, author)} [DONE]
    3. [DONE] We want an article screen composable. It should take args like article and possibly author.
        1. @Composable
        2. Fun ArticleScreen(article: Article, author: Author)
    4. [DONE] NavHostController should exist as a remember at the top level.
    5. [DONE] NavHost should register the article screen at the top level.

Article View

1. Use slots and scaffold for this when needed.
2. [DONE] Display article image in the header
    1. Use Koil
    2. article.imageURL
    3. We already had this image. Would be good to reuse than download again.
3. [DONE] Display title
    1. Text(title)
4. [DONE] Display body of the article
    1. [DONE] Text(body)
5. [DONE] Display image using imageUrlString of the Author
    1. Use Koil
    2. Look up author using author id and get image url to download it.

Key Cases:

6. If article image fails to download, hide it.
7. If koil fails to load image, hide it
8. If author data is empty, hide it

AuthorRepository

1. At app start, get all authors. Store in memory.
2. Create Repository
    1. Various ways of wanting authors?
        1. On article list, using author id
        2. On article page, using author id.
        3. authorRepository.getAuthor(author_id) (get from in memory)
3. Corner cases:

    1. While response fails. Should return a meangiful value when author details wanted. Don’t block app just for this reason.
    2. When incorrect or null author id is given. Should log meaningful information. And return a meaningful value so that the UI is not blocked.
4. Impl:
    1. AuthorRepository file
    2. Suspend fetchAuthors(): List<Author>. Stores in property. Should be fetched at app start. Don’t need flow.
    3. Suspend getAuthor(author_id): Author. Might not need flow as there may not be live data.

League UI

1. When league tab is clicked, get all leagues. Suspend List<League> getAllLeagues() in which LeagueApi.kt file.
2. When a league is looked up by league id, can get league details from there.
3. Show list of leagues
4. When a league is clicked, show all the articles in that league
    1. Show header for league page.
5. When an article is clicked, show the article.

Key Cases:
1. Leagues are unable to be fetched.
2. Image of league can’t be rendered
3. Articles can’t be fetched.

Potential broken patterns

4. Can we do a coroutine instead of a callback in ArticlesViewModel?
5. For MainActivity.BottomNavigation, can we do coroutine instead of callback for onScreenSelected?
6. Write one test per type:
    1. JUnit or Android test
    2. Compose test
    3. Coroutine test
7. Article has two model files

Potential leaks or inefficiencies

1. Use persistence
    1. For authors. Only have author name and id in memory.
    2. Store authors pics.
    3. For leagues
2. Auto convert json definition to model.
3. Convert xml to compose

Extensions

1. When author is clicked, fetch from persistence, all author details, and show.
2. Can make getAuthors “flow” type and show them on a separate screen