技术文档

OverView

https://github.com/OrangeJuiceWithIce/ByteDance-Client-Side-Training-Camp-Proj

主体为两个Activity：新闻首页(index=2)、历史记录页(index=4)

采取MVVM架构,组件化设计

使用ComposeUI

功能:数据和卡片展示、加载更多、下拉刷新、数据库sqlite历史记录

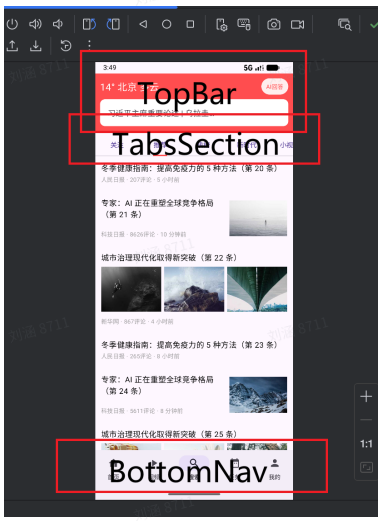
View

基本布局(两个activity,intent切换)

MainActivty

/ui/main/mainscreen

- /ui/main/components/TopBar
- /ui/BottomNavBar
- [content]:
  - ui/main/components/TabsSection
  - ui/news/NewsList
    - ui/news/NewsItem

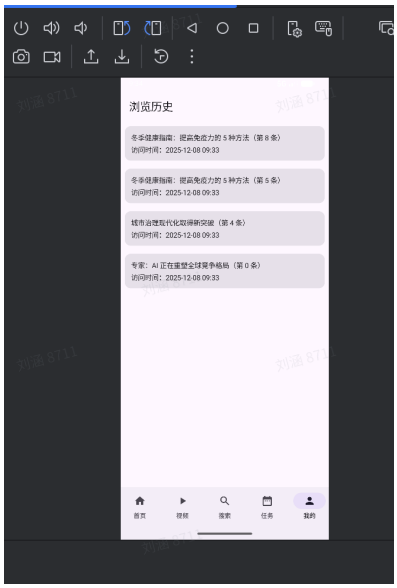


HistoryActivity

/ui/history/historyscreen

- /ui/history/components/HistoryTopBar
- ui/BottomNavBar
- [content]:
  - ui/history/components/HistoryList

- ui/history/components/HistoryItem



组件实现

- **MainScreen/HistoryScreen**——Compose.Scaffold
- **TopBar/HistoryTopBar**——Row/Column/TopAppBar...
- **BottomNavBar**——Compose.NavigationBar
- **TabsSection**——Compose.ScrollableTabRow
- **NewsList/HistoryList**——Compose.LazyColumn

组件化设计

- 支持组件复用
- 将状态管理的逻辑抽取出来

ui/main/components/TabsSection

```
fun TabsSection(
    tabs: List<String> = listOf("关注", "推荐", "热榜", "新时代", "小视频", "视频"),
    selectedTab: String,//被选中的tab
    onTabSelected: (String) -> Unit
)
```

ui/news/NewsList

```
fun NewsList(
    news: List<News>,
    onLoadMore: () -> Unit
)
```

ui/BottomNavBar

```
fun BottomNavBar(
    items: List<BottomNavItem>,
    selectedIndex: Int,//默认选中的按键
    onItemSelected: (Int) -> Unit
)
```

Model

数据模型

新闻

model/news/News

```
data class News(
    val id: String,
    val title: String,
    val source: String,//消息来源
    val commentCount: Int,//评论数
    val time: String,//发布时间
    val images: List<String> = emptyList(),//图片列表
    val type: NewsType = NewsType.PureText
)

enum class NewsType {
    PureText,//纯文字类型
    SingleImage,//单图片类型
    MultiImage,//多图片类型
}
```

历史记录entry

model/history/HistoryEntry

```
data class HistoryEntry(
    val id: Long?=null,
    val newsId: String,
    val title: String,
    val readTime: Long
)
```

MainUiState

model/main/MainUiState

```
data class MainUiState(
    val selectedTab: String = "推荐",
    val newsList: List<News> = emptyList(),
    val isRefreshing: Boolean = false
)
```

BottomNavItem

用于自定义底部导航栏的按钮

model/BottomNavItem

```
data class BottomNavItem(
    val index: Int,
    val label: String,
    val icon: ImageVector
)
```

数据管理

Repo

两个Repo分别管理News和History

news/data/NewsRepository

```
interface NewsRepository {
    suspend fun getNewsList(
        tab: String,//对应TabsSection的选项
        page: Int,//请求的page offset
        pageSize: Int//请求的page size
    ): List<News>
}
```

代码块

```
class HistoryRepository(private val dao: HistoryDao) {
    fun recordHistory(news: News) {
        val entity = HistoryEntry(
            newsId = news.id,
            title = news.title,
            readTime = System.currentTimeMillis()
        )
        dao.insert(entity)
    }
```

```
10      fun getHistory(): List<HistoryEntry> {
11          return dao.getAll()
12      }
13  }
```

DataSource

news/data/FakeNewsRepository.kt

- 用于测试
- 手动编写的数据位于 `news/data/FakeNewsSource.kt` 中
- 图片提供地址，在newsItem里使用 `Coil.AsyncImage` 进行异步获取

数据库sqlite（负责存储历史记录）

代码块

```
1   CREATE TABLE $TABLE_HISTORY (
2       $COLUMN_ID INTEGER PRIMARY KEY AUTOINCREMENT,
3       $COLUMN_NEWS_ID TEXT UNIQUE,
4       $COLUMN_TITLE TEXT NOT NULL,
5       $COLUMN_READ_TIME LONG
6   );
```

ViewModel

main/MainViewModel

```
1   class MainViewModel(
2       private val repo: NewsRepository = FakeNewsRepository()
3   ) : ViewModel() {
4
5       private val _uiState = MutableStateFlow(MainUiState())
6       val uiState: StateFlow<MainUiState> = _uiState
7
8       private var page = 0
9       private val pageSize = 10
10      private var isLoading = false
11      private var isEndReached = false
12
13      fun onTabSelected(tab: String)
14      fun loadFirstPage(tab: String) {}//切换tab时加载第一页
15      fun loadNextPage() {}//靠近底部时加载下一页
16      fun refresh() {}//下拉刷新
```

```
17    }
```

StateFlow传递信息

1. uiState:只读，通知view层重新渲染
2. _uiState: 可变，用于内部维护更新

功能实现

加载更多

news/ui/NewsList

```
1   val shouldLoadMore = remember {
2       derivedStateOf {
3           val total = listState.layoutInfo.totalItemsCount
4
5           if (total == 0) return@derivedStateOf false
6
7           val lastVisible =
    listState.layoutInfo.visibleItemsInfo.lastOrNull()?.index ?: 0
8
9           lastVisible >= total - 3
10      }
11  }
12
13  LaunchedEffect(shouldLoadMore.value) {
14      if (shouldLoadMore.value) {
15          onLoadMore()
16      }
17  }
```

- 使用 `derivedStateOf` ，监测 `LazyColumnState.layoutInfo.totalItemsCount` 和 `LazyColumnState.layoutInfo.visibleItemsInfo` 的变化
- 当两者大小接近时， `shouldLoadMore=True`
- `LaunchedEffect` 监测 `shouldLoadMore` 变化，值为 `True` 时预加载获取下一页的 `NewsList`

图片预加载

在前面提到的获取下一页NewsList的同时，对图片进行预加载

main/MainViewModel

```
private fun preloadImage(context: Context, url: String) {
    if (url.isBlank()) return

    val request = ImageRequest.Builder(context)
        .data(url)
        .diskCachePolicy(CachePolicy.ENABLED)
        .memoryCachePolicy(CachePolicy.ENABLED)
        .build()

    context.imageLoader.enqueue(request)
}
```

下拉刷新

使用Compose.PullToRefreshBox

main/ui/MainScreen

```
PullToRefreshBox(
    isRefreshing = uiState.isRefreshing,//刷新状态
    onRefresh = { viewModel.refresh() },//刷新逻辑，在一开始将uiState.isRefreshing设为true，末尾设为false
    modifier = Modifier
        .padding(padding)
        .fillMaxSize()
) {
    //content
}
```

数据库存储

- 在MainActivity点击item就会将item插入到数据库（采用冲突替换策略，保持最近访问时间）
- 在HistoryActivity通过HistoryRepo获取数据里的所有记录