OMinisterul Educaţiei, Culturii şi Cercetării al Republicii Moldova
Universitatea Tehnica a Moldovei
Facultatea Calculatoare, Informatică şi Microelectronică
Departamentul Ingineria Software şi Automatică

Raport

Lucrarea de laborator Nr. 5,6
la Programarea aplicatiilor mobile
1. Tema: Continuare laborator 4

A efectuat
St. gr. TI-206
Pleşu Cătălin

A verificat
asist. univ.
Alexandru Stamatin

Chişinău 2022

Scopul: De finisat aplicaţia

Sarcina:

1. **Încărcarea datelor se vor face din REST-API utilizind protocolul http.**
2. **Necesar este sa folosim paginare infinita la preluarea feed-ului de date.**
3. **Implementare o noua pagina UI - detalii**
4. **Adaptare proiect existent la Clean Architecture - separare pe cele 3 layer-e: Domain, Data, Presentation**

Varianta3.

Rest-API e aici:

https://news-app-api.k8s.devebs.net/

Design preluati de aici:

https://www.figma.com/file/vR3YnmX5hNTElDSnMqi5aD/PAM---Laborator-4?node-id=0%3A1

În figura 1 poate fi observata implementare citirii de pe api, aceasta este realizat cu ajutorul protocolului http, doarece serverul respectiv a fost implementat protocolul respectiv.

```dart
import 'dart:convert';
import 'package:data/entities/api/article_api_dto.dart';
import 'package:data/entities/api/articles_response_api_dto.dart';
import 'package:http/http.dart' as http;

abstract class NewsApiDataSource {
  Future<List<ArticleApiDto>> getArticles(int page, int per_page);
}

class NewsApiDataSourceImpl implements NewsApiDataSource {
  @override
  Future<List<ArticleApiDto>> getArticles(int page, int per_page) async {
    Uri uri = Uri.parse(
        'https://news-app-api.k8s.devebs.net/articles?per_page=$per_page&page=$page');
    var response = await http.get(
      uri,
    );

    if (response.statusCode == 200) {
      ArticlesResponseApiDto articlesResponse =
          ArticlesResponseApiDto.fromJson(jsonDecode(response.body));
      return articlesResponse.articles;
    }
    return [];
  }
}
```

Figura 1. Realizarea citirii de pe API

În figura 2 poate fi observat faptul că lista de ştiri se află întro listă ceea ce va permite să avem încărcăm în aplicaţie un număr infinit de elemente (deşi nu este posibil fiind că nu dispunem de timp suficient pentru a observa acest fenomen).

```
48      return Material(
49        color: Colors.white,
50        child: Obx(
51          () => ListView.builder(
52              controller: scrollController,
53              itemCount: controller.articleList.value.length + 1,
54              itemBuilder: (BuildContext context, int index) {
55                if (index == 0) {
56                  return Column(
57                    children: [
58                      Featured(),
59                      Container(
60                          margin: EdgeInsets.all(16),
61                          child: Row(
62                            children: [
63                              Text("News",
64                                  style: TextStyles.textStyleSourceSansPro20(
65                                      color: CustomColors.pickledBluewood,
66                                      fontWeight: FontWeight.w600)),  // Text
67                              Spacer(),
68                              Text("See all",
69                                  style: TextStyles.textStyleSourceSansPro16(
70                                      fontWeight: FontWeight.w600)),  // Text
71                            ],
72                          )),  // Row, Container
73                    ],
74                  );  // Column
75                } else {
76                  // return NewsItem();
77                  return NewsItem(article: controller.articleList.value[index - 1]);
78                }
79              }),  // ListView.builder
80        ),  // Obx
```

Figura 2. Realizarea paginarii infinite utilizand listView.Builder

În figura 3 este demonstrată realizarea paginii cu detalii care conţine două imagini, text şi nişte vectori SVG.

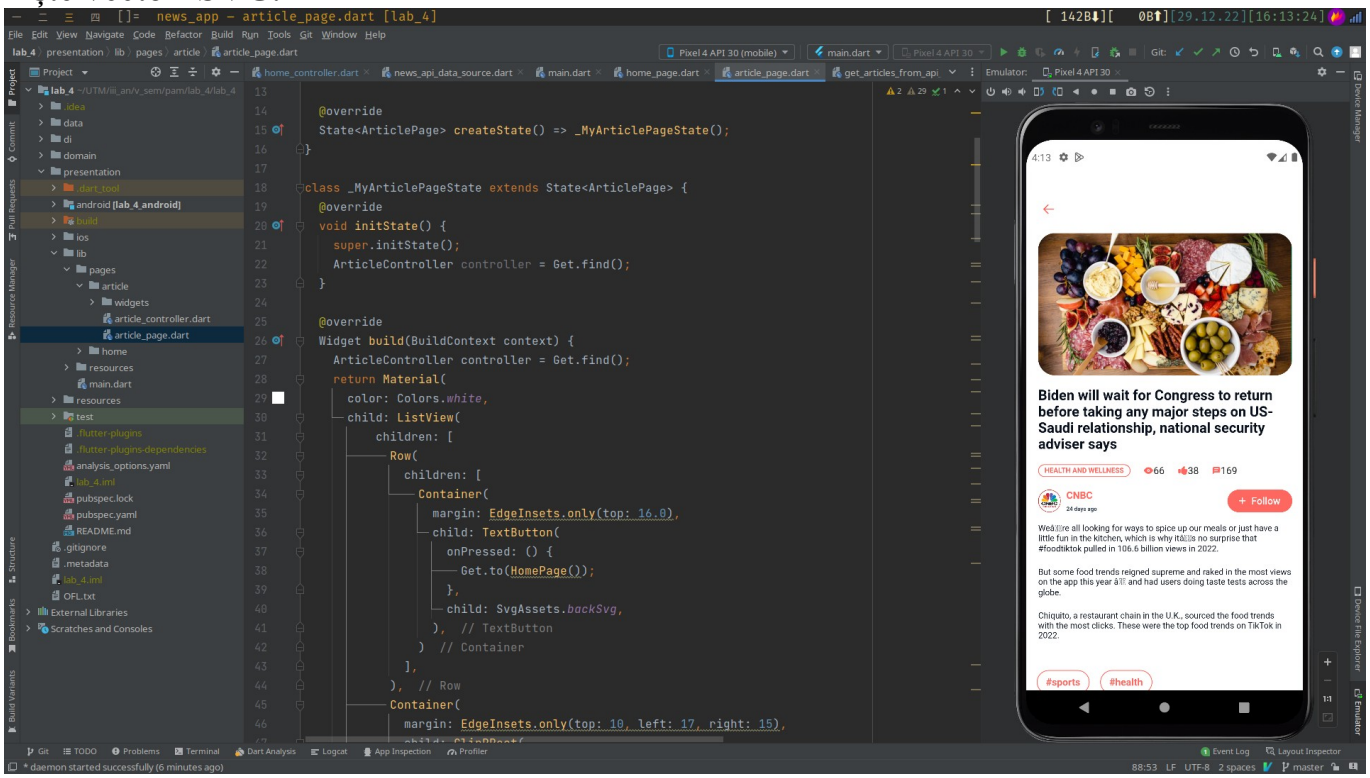

Figura 3. Implementarea paginii de detalii.

În figura 4 putem observa că în aplicația noastră sunt 4 proiecte, dintre care data, domain și presentation sunt conform arhitecturii curate, iar di (data injection) este ceva suplimentar pentru a simplifica utilizarea straturilor.

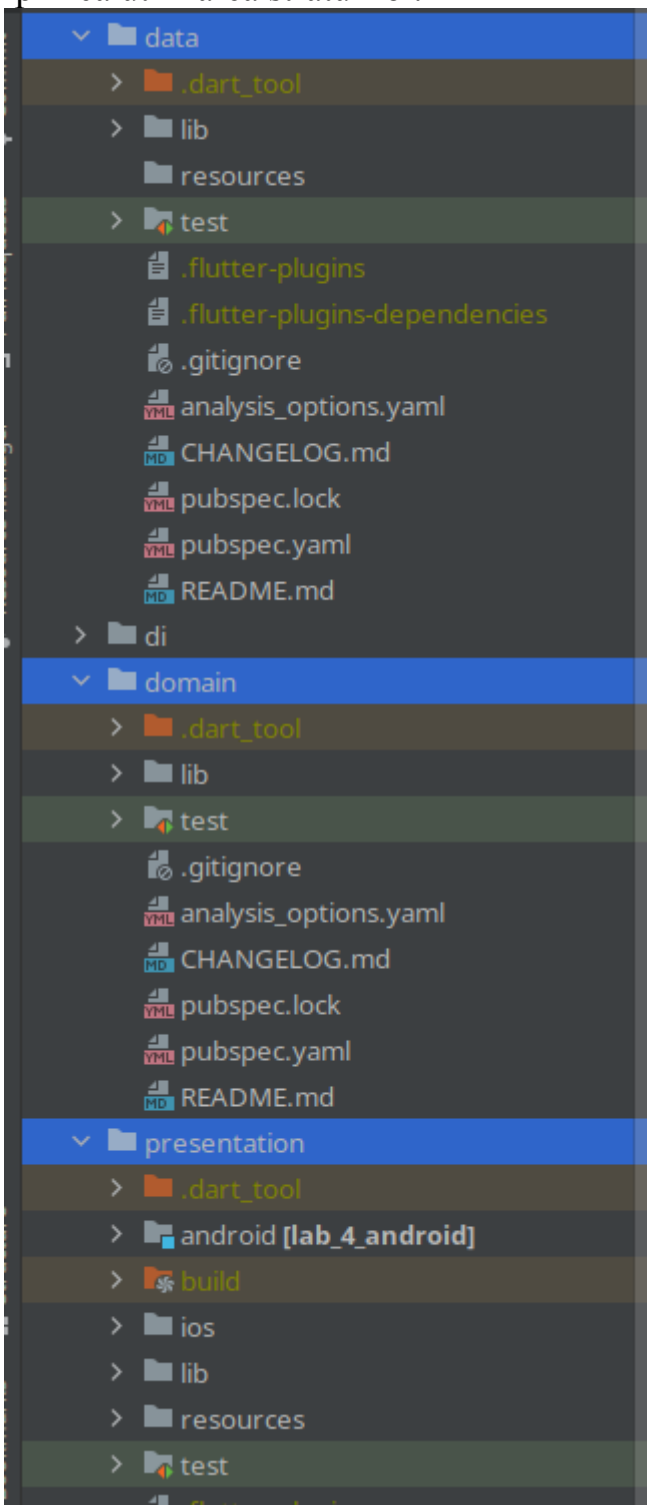

Figura 4. Adaptarea proiectului la Clean Architecture

Repozitorie github - https://github.com/CatalinPlesu/PAM_labs
(continuat în mapa cu laboratorul 4)

Concluzie:

În cadrul acestei lucrări de laborator am completat aplicația creată în cadrul lucrării precedente de laborator, am utilizat clean architecture și am citit datele de pe api, de asemente aplicația dată poate stoca datele într-o bază de date locală.