ДОДАТОК А

Таблиці результатів

Таблиця А.1: Кількість та список класів, які необхідно створити для реалізації функції системи

Feature	VIPER	MVP
Infrastructure layer	5 classes: CategoryDao.java, NoteDao.java, DbCategory.java, DbNote.java, NotesDatabase.java	5 classes: CategoryDao.java, NoteDao.java, DbCategory.java, DbNote.java, NotesDatabase.java
Domain layer	5 classes: Category.java, Note.java, CategoryDbMapper.java, NoteDbMapper.java, CategoryUtil.java	5 classes: Category.java, Note.java, CategoryDbMapper.java, NoteDbMapper.java, CategoryUtil.java
Base architecture	21 classes: BaseMapper.java, UserScope.java, ApplicationComponent.java, ActivityComponent.java, ActivityModule.java, ApplicationModule.java, DbModule.java, InteractorModule.java, MapperModule.java, PresenterModule.java, RouterModule.java, SchedulerModule.java, BaseAdapter.java, BaseViewHolder.java, ItemLongClickSelectedListener.java, ItemSelectedListener.java, BaseActivity.java, BaseFragment.java, BaseRouter.java, RxPresenter.java, NotesApplication.java	24 classes: BaseMapper.java, UserScope.java, ApplicationComponent.java, ActivityComponent.java, ActivityModule.java, ApplicationModule.java, DbModule.java, RepositoryModule.java, MapperModule.java, PresenterModule.java, SchedulerModule.java, BaseAdapter.java, BaseViewHolder.java, ItemLongClickSelectedListener.java, ItemSelectedListener.java, BaseActivity.java, BaseFragment.java, BaseRouter.java, RxPresenter.java, NotesApplication.java, CategoryRepository.java, CategoryDbRepository.java NoteRepository.java, NoteDbRepository.java

Продовження таблиці А.1

Main screen	8 classes: MainPresenter.java, MainPresenterImpl.java, MainRouter.java, MainRouterImpl.java, MainView.java, MainActivity.java, InitDbInteractor.java, InitDbInteractorImpl.java	4 classes: MainPresenter.java, MainPresenterImpl.java, MainView.java, MainActivity.java
Create note	9 classes: CreateNotePresenter.java, CreateNotePresenterImpl.java, CreateNoteRouter.java, CreateNoteRouterImpl.java, CreateNoteFragment.java, CreateNoteView.java, CreateNoteActivity.java, AddNoteInteractor.java, AddNoteDbInteractor.java	5 classes: CreateNotePresenter.java, CreateNotePresenterImpl.java, CreateNoteFragment.java, CreateNoteView.java, CreateNoteActivity.java,
View note	7 classes: EditNotePresenter.java, EditNotePresenterImpl.java, EditNoteRouter.java, EditNoteRouterImpl.java, EditNoteFragment.java, EditNoteView.java, EditNoteActivity.java	7 classes: EditNotePresenter.java, EditNotePresenterImpl.java, EditNoteFragment.java, EditNoteView.java, EditNoteActivity.java
Edit note	2 classes: EditNoteInteractor.java, EditNoteDbInteractor.java	0 classes
Delete note	2 classes: DeleteNoteInteractor.java, DeleteNoteDbInteractor.java	0 classes
View list of all notes	10 classes: NotesListPresenter.java, NotesListPresenterImpl.java, NotesListRouter.java, NoteslistRouterImpl.java, NotesListFragment.java, NotesListView.java, NotesAdapter.java, NotesViewHolder.java, NotesListInteractor.java, NotesListDbInteractor.java	6 classes: NotesListPresenter.java, NotesListPresenterImpl.java NotesListFragment.java, NotesListView.java, NotesAdapter.java, NotesViewHolder.java
Create category	9 classes: CreateCategoryPresenter.java, CreateCategoryPresenterImpl.java, CreateCategoryRouter.java, CreateCategoryRouterImpl.java, CreateCategoryView.java, CreateCategoryFragment.java, CreateCategoryActivity.java, AddCategoryInteractor.java, AddCategoryDbInteractor.java	5 classes: CreateCategoryPresenter.java, CreateCategoryPresenterImpl.java, CreateCategoryView.java, CreateCategoryFragment.java, CreateCategoryActivity.java

Кінець таблиці А.1

Edit category	9 classes: EditCategoryPresenter.java, EditCategoryPresenterImpl.java, EditCategoryRouter.java, EditCategoryRouterImpl.java, EditCategoryView.java, EditCategoryFragment.java, EditCategoryActivity.java, EditCategoryInteractor.java, EditCategoryDbInteractor.java	5 classes: EditCategoryPresenter.java, EditCategoryPresenterImpl.java , EditCategoryView.java, EditCategoryFragment.java, EditCategoryActivity.java
Delete category	2 classes: DeleteCategoryInteractor.java, DeleteCategoryDbInteractor.java	0 classes
Set category for a note	1 class: CategoryAdapter.java	1 class: CategoryAdapter.java
View notes by category	10 classes: CategoriesListPresenter.java, CategoriesListPresenterImpl.java, CategoriesListRouter.java, CategorieslistRouterImpl.java, CategoriesListFragment.java, CategoriesListView.java, CategoryAdapter.java, CategoryViewHolder.java, CategoriesListInteractor.java, CategoriesListDbInteractor.java	6 classes: CategoriesListPresenter.java, CategoriesListPresenterImpl.java, CategoriesListFragment.java, CategoriesListView.java, CategoryAdapter.java, CategoryViewHolder.java

Таблиця А.2: Кількість та список класів, які необхідно змінити для реалізації функції системи

Feature	VIPER	MVP
Infrastructure layer	0 classes	0 classes
Domain layer	0 classes	0 classes
Base architecture	0 classes	0 classes
Main screen	4 classes: RouterModule.java, PresenterModule.java, ActivityComponent.java, InteractorModule.java	4 classes: PresenterModule.java, ActivityComponent.java, CategoriesRepository.java, CategoriesDbRepository.java

Продвження таблиці А.2

Create note	7 classes: RouterModule.java, PresenterModule.java, ActivityComponent.java, InteractorModule.java, NotesListRouterImpl.java, NotesListPresenter.java, NotesListPresenterImpl.java	8 classes: PresenterModule.java, ActivityComponent.java, NotesRepository.java, NotesDbRepository.java, NotesListView.java, NotesListFragment.java, NotesListPresenter.java, NotesListPresenterImpl.java
View note	4 classes: RouterModule.java, PresenterModule.java, ActivityComponent.java, NotesListRouterImpl.java	6 classes: PresenterModule.java, ActivityComponent.java, NotesListFragment.java, NotesListView.java, NotesListPresenter.java, NotesListPresenterImpl.java
Edit note	4 classes: EditNotePresenterImpl.java, EditNoteFragment.java, EditNoteView.java, InteractorModule.java	5 classes: EditNotePresenterImpl.java, EditNoteFragment.java, EditNoteView.java, NotesRepository.java, NotesDbRepository.java
Delete note	5 classes: EditNotePresenterImpl.java, EditNotePresenter.java, EditNoteFragment.java, EditNoteView.java, InteractorModule.java	6 classes: EditNotePresenterImpl.java, EditNotePresenter.java, EditNoteFragment.java, EditNoteView.java, NotesRepository.java, NotesDbRepository.java
View list of all notes	5 classes: MainRouterImpl.java, PresenterModule.java, RouterModule.java, ActivityComponent.java InteractorModule.java	5 classes: PresenterModule.java, ActivityComponent.java, NotesRepository.java, NotesDbRepository.java, ActivityComponent.java
Create category	5 classes: RouterModule.java, PresenterModule.java, ActivityComponent.java, InteractorModule.java, CategoriesListRouter.java	8 classes: PresenterModule.java, ActivityComponent.java, CategoriesListFragment.java, CategoriesListView.java CategoriesRepository.java, CategoriesDbRepository.java, CategoriesListFragment.java, CategoriesListView.java
Edit category	5 classes: RouterModule.java, PresenterModule.java, ActivityComponent.java, InteractorModule.java, CategoriesListRouterImpl.java	6 classes: PresenterModule.java, ActivityComponent.java, CategoriesRepository.java, CategoriesDbRepository.java, CategoriesListFragment.java, CategoriesListView.java
Delete category	4 class: InteractorModule.java, EditCategoryFragment.java, EditCategoryPresenter.java, EditCategoryPresenterImpl.java	5 classes: EditCategoryFragment.java, EditCategoryPresenter.java, EditCategoryPresenterImpl.java , CategoriesRepository.java, CategoriesDbRepository.java

Кінець таблиці А.2

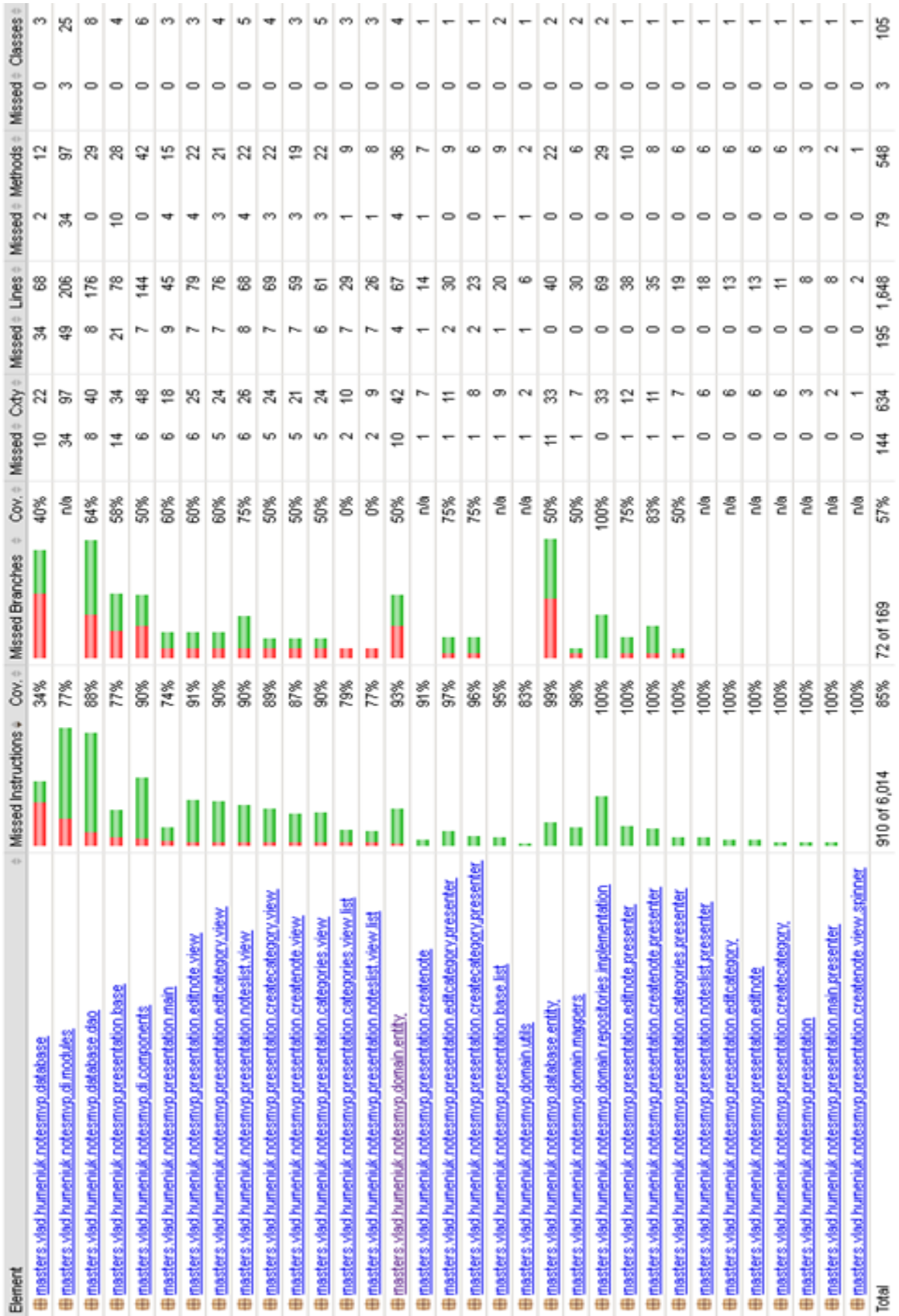
Set category for a note	6 class: CreateNoteFragment.java, CreateNotePresenter.java, CreateNotePresenterImpl.java, EditNotePresenter.java, EditNotePresenterImpl.java, EditNoteFragment.java	6 class: CreateNoteFragment.java, CreateNotePresenter.java, CreateNotePresenterImpl.java, EditNotePresenter.java, EditNotePresenterImpl.java, EditNoteFragment.java
View notes by category	8 classes: MainRouterImpl.java, PresenterModule.java, RouterModule.java, ActivityComponent.java, InteractorModule.java, NotesListFragment.java, NotesListPresenter.java, NotesListPresenterImpl.java	9 classes: MainActivity.java, PresenterModule.java, RouterModule.java, ActivityComponent.java, NotesListFragment.java, NotesListPresenter.java, NotesListPresenterImpl.java, NotesRepository.java, NotesDbRepository.java

Таблиця А.3: Кількість рядків коду, які необхідно створити або змінити для реалізації функції системи

Feature	VIPER	MVP
Infrastructure layer	216 LOC	216 LOC
Domain layer	270 LOC	270 LOC
Base architecture	631 LOC	659 LOC
Main screen	242 LOC	185 LOC
Create note	356 LOC	336 LOC
View note	278 LOC	266 LOC
Edit note	102 LOC	74 LOC
Delete note	110 LOC	80 LOC
View list of all notes	293 LOC	267 LOC
Create category	336 LOC	300 LOC
Edit category	390 LOC	344 LOC
Delete category	51 LOC	35 LOC
Set category for a note	123 LOC	107 LOC
View notes by category	452 LOC	405 LOC

ДОДАТОК Б

Результати покриття тестами



Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
masters.vlad.humeniuk.notesmvp.database		34%		40%	10	22	34	68	2	12	0	3
masters.vlad.humeniuk.notesmvp.di.modules		77%		n/a	34	97	49	206	34	97	3	25
masters.vlad.humeniuk.notesmvp.database.dao		86%		64%	8	40	8	176	0	29	0	8
masters.vlad.humeniuk.notesmvp.presentation.base		77%		58%	14	34	21	78	10	28	0	4
masters.vlad.humeniuk.notesmvp.di.components		90%		50%	6	48	7	144	0	42	0	6
masters.vlad.humeniuk.notesmvp.presentation.main		74%		60%	6	18	9	45	4	15	0	3
masters.vlad.humeniuk.notesmvp.presentation.editnote.view		91%		60%	6	25	7	79	4	22	0	3
masters.vlad.humeniuk.notesmvp.presentation.editcategory.view		90%		60%	5	24	7	76	3	21	0	4
masters.vlad.humeniuk.notesmvp.presentation.noteslist.view		90%		75%	6	26	8	68	4	22	0	5
masters.vlad.humeniuk.notesmvp.presentation.createcategory.view		89%		50%	5	24	7	69	3	22	0	4
masters.vlad.humeniuk.notesmvp.presentation.createnote.view		87%		50%	5	21	7	59	3	19	0	3
masters.vlad.humeniuk.notesmvp.presentation.categories.view		90%		50%	5	24	6	61	3	22	0	5
masters.vlad.humeniuk.notesmvp.presentation.categories.view.list		79%		0%	2	10	7	29	1	9	0	3
masters.vlad.humeniuk.notesmvp.presentation.noteslist.view.list		77%		0%	2	9	7	26	1	8	0	3
masters.vlad.humeniuk.notesmvp.domain.entity		93%		50%	10	42	4	67	4	36	0	4
masters.vlad.humeniuk.notesmvp.presentation.createnote		91%		n/a	1	7	1	14	1	7	0	1
masters.vlad.humeniuk.notesmvp.presentation.editcategory.presenter		97%		75%	1	11	2	30	0	9	0	1
masters.vlad.humeniuk.notesmvp.presentation.createcategory.presenter		96%		75%	1	8	2	23	0	6	0	1
masters.vlad.humeniuk.notesmvp.presentation.base.list		95%		n/a	1	9	1	20	1	9	0	2
masters.vlad.humeniuk.notesmvp.domain.utils		83%		n/a	1	2	1	6	1	2	0	1
masters.vlad.humeniuk.notesmvp.database.entity		99%		50%	11	33	0	40	0	22	0	2
masters.vlad.humeniuk.notesmvp.domain.mappers		98%		50%	1	7	0	30	0	6	0	2
masters.vlad.humeniuk.notesmvp.domain.repositories.implementation		100%		100%	0	33	0	69	0	29	0	2
masters.vlad.humeniuk.notesmvp.presentation.editnote.presenter		100%		75%	1	12	0	38	0	10	0	1
masters.vlad.humeniuk.notesmvp.presentation.createnote.presenter		100%		83%	1	11	0	35	0	8	0	1
masters.vlad.humeniuk.notesmvp.presentation.categories.presenter		100%		50%	1	7	0	19	0	6	0	1
masters.vlad.humeniuk.notesmvp.presentation.noteslist.presenter		100%		n/a	0	6	0	18	0	6	0	1
masters.vlad.humeniuk.notesmvp.presentation.editcategory		100%		n/a	0	6	0	13	0	6	0	1
masters.vlad.humeniuk.notesmvp.presentation.editnote		100%		n/a	0	6	0	13	0	6	0	1
masters.vlad.humeniuk.notesmvp.presentation.createcategory		100%		n/a	0	6	0	11	0	6	0	1
masters.vlad.humeniuk.notesmvp.presentation		100%		n/a	0	3	0	8	0	3	0	1
masters.vlad.humeniuk.notesmvp.presentation.main.presenter		100%		n/a	0	2	0	8	0	2	0	1
masters.vlad.humeniuk.notesmvp.presentation.createnote.view.skinner		100%		n/a	0	1	0	2	0	1	0	1
Total	910 of 6,014	85%	72 of 169	57%	144	634	195	1,646	79	548	3	105

Рисунок Б.1: покриття тестами для MVP

Рисунок Б.2: покриття тестами для VIPER

ДОДАТОК В

Слайди презентації

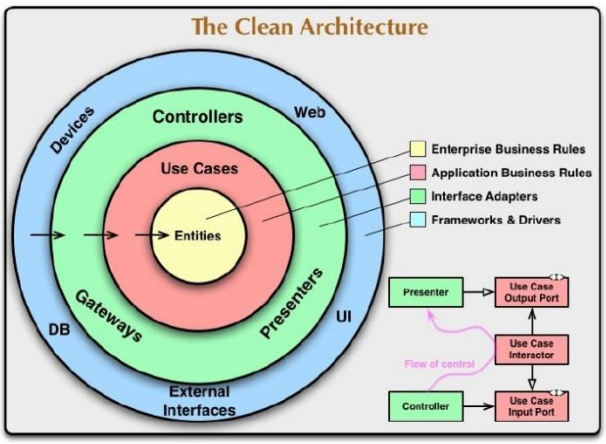
Атестаційна робота магістра

Метод порівняння архітектур для Андроід на прикладі Viper та Mvp

Керівник:
проф. Дудар З.В.

Виконав:
ст.гр. ІПЗм-17-1
Гуменюк В.В.

Предметна обсласть. Основні принципи архітектури

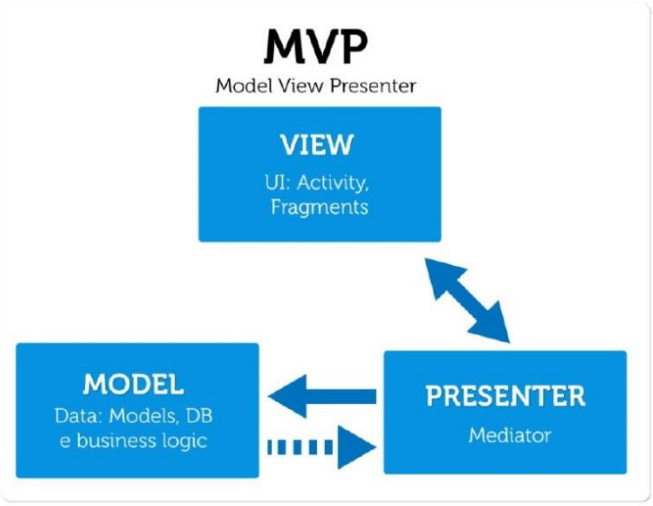


Основні принципи архітектури:

- Single Responsibility Principle
- Open/Closed Principle
- Liskov Substitution Principle
- Interface Segregation Principle
- Dependency Inversion

2

Предметна область. **MVP**



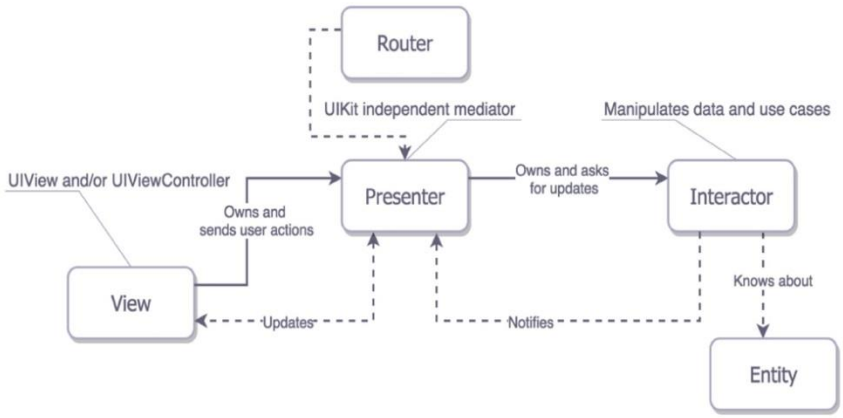
M odel

V iew

P resenter

3

Предметна область. **VIPER**



V iew

I nteractor

P resenter

E ntity

R outer

4

Постановка задачі

Щоб зрозуміти, як правильно підібрати архітектуру для проекту з конкретними вимогами та обмеженнями, треба відповісти на такі питання:

- Яку архітектуру слід використовувати для розробки програми з урахуванням можливих обмежень (тобто часу, вартості, бажаної продуктивності)?
- Яку архітектуру слід використовувати для розробки додатків конкретного розміру (тобто невеликого застосування прототипу, середнього розміру, великого застосування з великим планом розширення)?
- Яка архітектура краще підходить для модульного застосування?
- Яка архітектура краще підходить для бібліотек?

5

Дослідження. Загальні положення

На основі огляду літератури був спланований експеримент розрахований на оцінку архітектур програмного забезпечення Android. Програмні архітектури, MVP і VIPER, оцінюються шляхом вимірювання набору властивостей для обох архітектур у стабільному середовищі. Дані, отримані в експерименті, аналізуються для порівняння архітектур і відповіді на дослідницькі питання. Дане дослідження вимірює такі властивості, як продуктивність, тестованість і змінюваність.
Основною властивістю архітектури проекту є ремонтопридатність. Тестованість і змінюваність визначають цю властивості, тому вони будуть вимірюватися.

Продуктивність повинна вимірюватися, оскільки архітектурні вимоги свідчать, що програмна система має належно керувати пам'яттю, а інтерфейс користувача не повинен показувати будь-які затримки.

6

Дослідження. Вимірювання продуктивності

Стандарт ISO / IES 25010 представляє модель якості програмного забезпечення, де продуктивність групується за трьома категоріями; потенціалу, часового поведінки та використання ресурсів.
Продуктивність буде вимірюватися за допомогою Android Profiler. Він був розроблений для вимірювання різних областей продуктивності для додатків Android, таких як CPU, GPU, пам'ять, акумулятор або мережа. Час, необхідний для виконання сценарію використання та споживання пам'яті, є виміряними показниками.

7

Дослідження. Вимірювання змінюванності

З моделі якості програмного забезпечення ISO / IES 25010 змінюваність визначається як "Ступінь, до якої продукт або система можуть бути ефективно і ефективно змінені без введення дефектів або погіршення існуючої якості продукції".
Щоб оцінити змінність, слід зібрати наступні показники:

- Які і скільки класів потрібно модифікувати?
- Які і скільки нових класів потрібно додати?
- Скільки нових рядків коду (LOC) потрібно додати?

Всі випадки використання розділені на менші функції. Ці показники збираються для кожної функції.

8

Дослідження. Вимірювання тестованості

ISO / IES 25010 визначає тестованість як "Ступінь ефективності та ефективності, з якою можуть бути встановлені критерії випробування для системи, продукту або компонента, і тести можуть бути виконані для визначення того, чи були ці критерії виконані".
Покриття тесту або покриття коду - це показник, який визначає, яку частину системи можна перевірити тестами.

9

Дослідження. Сценарії використання

Для оцінки архітектур розроблено набір сценаріїв використання.
Кожен сценарій використання складається з декількох дій, таких як перемикання екранів, введення тексту, доступ до бази даних тощо.
Кожен сценарій використання реалізується з використанням як архітектурних підходів MVP, так і VIPER, а потім виконуються тести на кожному випадку використання, і збираються метрики.
Для кожного випадку використання логіка програми залишається такою ж, середовище виконання тесту постійне, єдиною відмінністю є архітектура.

10

Результати. Змінюваність системи

Feature	VIPER	MVP
Infrastructure layer	5 classes	5 classes
Domain layer	5 classes	5 classes
Base architecture	21 classes	24 classes
Main screen	8 classes	4 classes
Create note	9 classes	5 classes
View note	7 classes	7 classes
Edit note	2 classes	0 classes
Delete note	2 classes	0 classes
View list of all notes	10 classes	6 classes
Create category	9 classes	5 classes
Edit category	9 classes	5 classes
Delete category	2 classes	0 classes
Set category for a note	1 class	1 class
View notes by category	10 classes	6 classes

Кількість класів, які необхідно створити для реалізації конкретної функції

11

Результати. Змінюваність системи

Feature	VIPER	MVP
Infrastructure layer	0 classes	0 classes
Domain layer	0 classes	0 classes
Base architecture	0 classes	0 classes
Main screen	4 classes	4 classes
Create note	7 classes	8 classes
View note	4 classes	6 classes
Edit note	4 classes	5 classes
Delete note	5 classes	6 classes
View list of all notes	5 classes	5 classes
Create category	5 classes	8 classes
Edit category	5 classes	6 classes
Delete category	4 class	5 classes
Set category for a note	6 class	6 class
View notes by category	8 classes	9 classes

Кількість класів, які необхідно відредагувати для реалізації конкретної функції

12

Результати. Змінюваність системи

Feature	VIPER	MVP
Infrastructure layer	216 LOC	216 LOC
Domain layer	270 LOC	270 LOC
Base architecture	631 LOC	659 LOC
Main screen	242 LOC	185 LOC
Create note	356 LOC	336 LOC
View note	278 LOC	266 LOC
Edit note	102 LOC	74 LOC
Delete note	110 LOC	80 LOC
View list of all notes	293 LOC	267 LOC
Create category	336 LOC	300 LOC
Edit category	390 LOC	344 LOC
Delete category	51 LOC	35 LOC
Set category for a note	123 LOC	107 LOC
View notes by category	452 LOC	405 LOC

Рядки коду, які потрібно додати / відредагувати для реалізації особливої функції

13

Результати. Тестованість

masters.vlad.humeniuk.notesviper.database.entity		99%
masters.vlad.humeniuk.notesviper.domain.mappers		98%
masters.vlad.humeniuk.notesviper.presentation.editnote.presenter		100%
masters.vlad.humeniuk.notesviper.presentation.createnote.presenter		100%
masters.vlad.humeniuk.notesviper.presentation.noteslist.presenter		100%
masters.vlad.humeniuk.notesviper.presentation.categories.presenter		100%
masters.vlad.humeniuk.notesviper.presentation.main.presenter		100%
masters.vlad.humeniuk.notesviper.presentation.main.router		100%
masters.vlad.humeniuk.notesviper.presentation.categories.router		100%
masters.vlad.humeniuk.notesviper.presentation		100%
masters.vlad.humeniuk.notesviper.presentation.createnote.router		100%
masters.vlad.humeniuk.notesviper.presentation.editcategory.router		100%
masters.vlad.humeniuk.notesviper.presentation.editnote.router		100%
masters.vlad.humeniuk.notesviper.presentation.createcategory.router		100%
masters.vlad.humeniuk.notesviper.presentation.createnote.view.spinner		100%
Total	1,094 of 7,355	85%

VIPER
AndroidTest

14

Результати. Тестованість

masters.vlad.humeniuk.notesmvp.domain.utils		83%
masters.vlad.humeniuk.notesmvp.database.entity		99%
masters.vlad.humeniuk.notesmvp.domain.mappers		98%
masters.vlad.humeniuk.notesmvp.domain.repositories.implementation		100%
masters.vlad.humeniuk.notesmvp.presentation.editnote.presenter		100%
masters.vlad.humeniuk.notesmvp.presentation.createnote.presenter		100%
masters.vlad.humeniuk.notesmvp.presentation.categories.presenter		100%
masters.vlad.humeniuk.notesmvp.presentation.noteslist.presenter		100%
masters.vlad.humeniuk.notesmvp.presentation.editcategory		100%
masters.vlad.humeniuk.notesmvp.presentation.editnote		100%
masters.vlad.humeniuk.notesmvp.presentation.createcategory		100%
masters.vlad.humeniuk.notesmvp.presentation		100%
masters.vlad.humeniuk.notesmvp.presentation.createnote.view.spinner		100%
Total	969 of 6,093	84%

MVP
AndroidTest

15

Результати. Продуктивність

Швидкість введення та натискання елементів керування була незмінною, оскільки виконувалася за допомогою автоматизованої системи. Швидкість нанесення та завантаження була постійною, оскільки установка пристрою була постійною. Єдиним фактором, що впливає на час, було кількість операцій, які потрібно було завантажувати, що відрізняється для архітектур додатків. У таблиці наведено середні 1000 зібраних результатів.

Nexus 5X 8.1.0, time(ms)	Create note	Edit Note	Create category	Edit category	Create category note	Edit category note	Delete note	Delete category	Delete category note	Show info
Viper	2916	2746	2309	2486	3564	2751	1422	1611	1854	1076
Mvp	3162	2996	2475	2681	3763	2920	1517	1689	2027	1143

16

Результати. Продуктивність

Налаштування пристрою було постійним. Єдиним фактором, що впливає на обсяг пам'яті, необхідної для програми, є кількість операцій, які потрібно було завантажити, що відрізняється для архітектур додатків. У таблиці наведено середні 1000 зібраних результатів.

Nexus 5X 8.1.0, memory (mb)	Create note	Edit Note	Create category	Edit category	Create category note	Edit category note	Delete note	Delete category	Delete category note	Show info
Viper	120.6	106.3	87.2	101.2	128.3	113.5	80.4	83.0	87.2	60.8
Mvp	110.1	120.4	112.8	122.7	145.1	124.7	90.2	94.1	99.7	66.5

17

Результати

RQ1. Яка архітектура Android є більш доступною для обслуговування, MVP або VIPER?

Достовірність оцінюється як склад тестованості і модифікованості.

Архітектура VIPER показала кращі результати у тестованості, але різниця була незначною.

Архітектура MVP вимагала меншої кількості LOC для реалізації кожної функції в середньому на 10%.

RQ2. Яка архітектура Android призводить до кращої продуктивності, MVP або VIPER?

Продуктивність оцінюється шляхом вимірювання часу, необхідного для завершення сценарію використання та споживання пам'яті.

VIPER був кращим за обома показників.

18

Результати

RQ3. Яка архітектура Android краще підходить для різних типів проектів, MVP або VIPER?

Архітектура MVP вимагає менше ліній коду, що пишуться, ніж VIPER, тому розробнику потрібно менше часу для завершення програми. Це також призводить до зниження вартості.

Архітектура VIPER демонструє кращу продуктивність і перевірочність, тому ця архітектура буде кращою, якщо продуктивність і якість програми є більш важливими, ніж вартість і час.

Архітектура VIPER призводить до більш чистого коду, ніж MVP, оскільки класи менші, а обов'язки розділені краще між класами.

19

Висновки

Обидві архітектури життєздатні.

Ця робота надає інформацію для двох популярних архітектур андроїд-проектів MVP і VIPER.

Майбутня робота може включати вимірювання продуктивності на більшій кількості пристроїв Android, збирання показників на великих комерційних проектах, оцінка нових архітектур.

20