UF1_Pt1 - Projecte

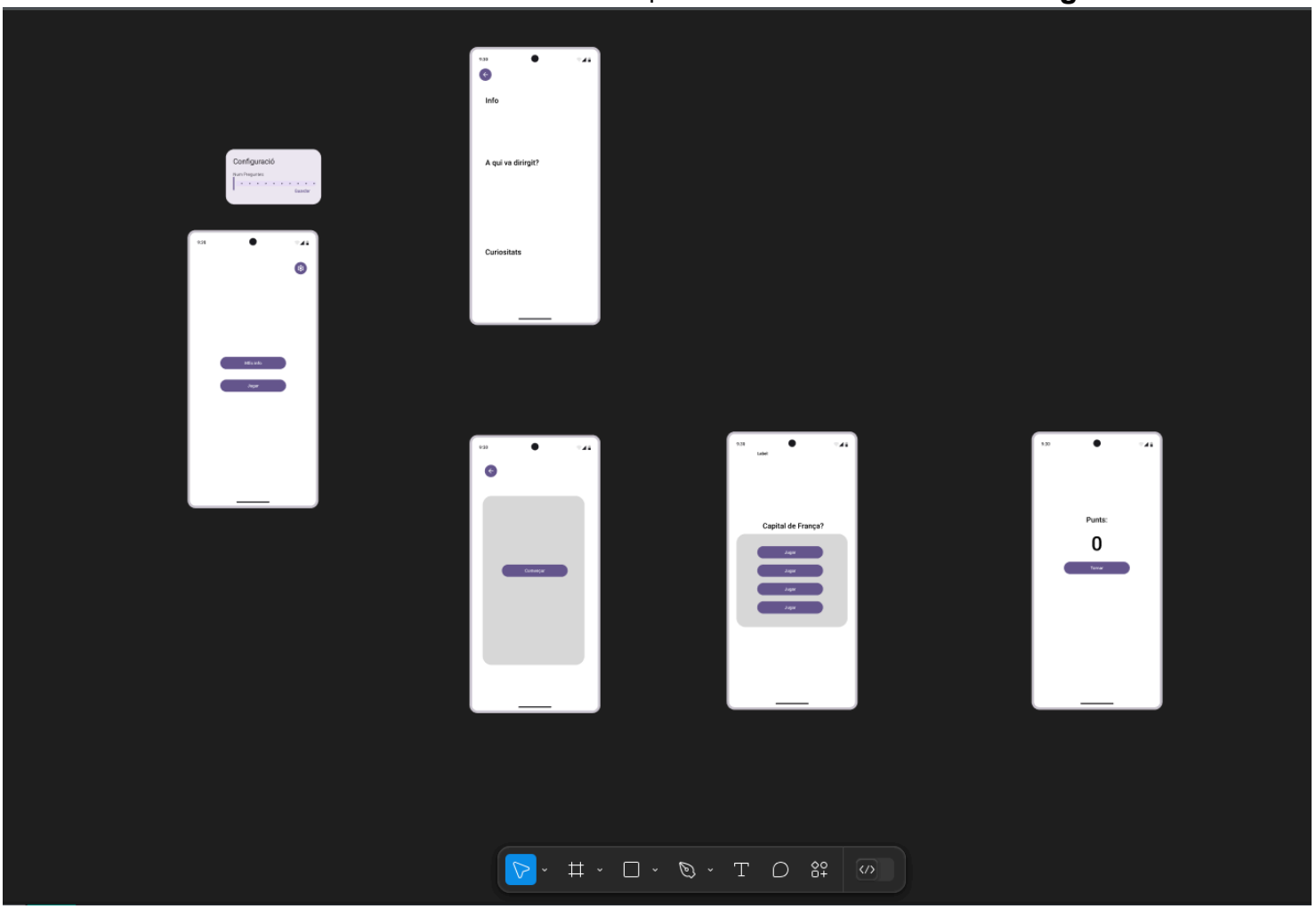
Ferran Garcia Garcia

Curs: 2024-2025

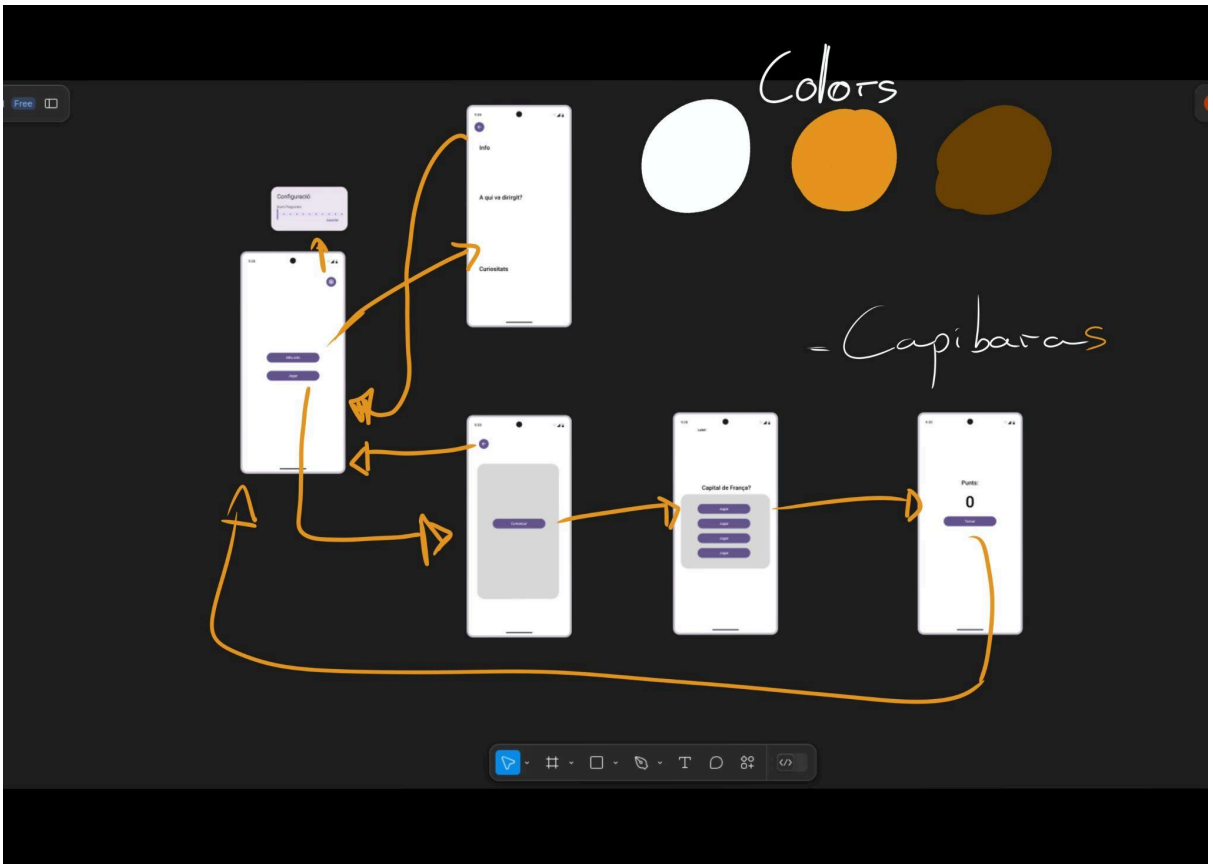
Index

Dissenys

- Primer he realitzat un esbós molt simple de les view necessaries amb **Figma**.

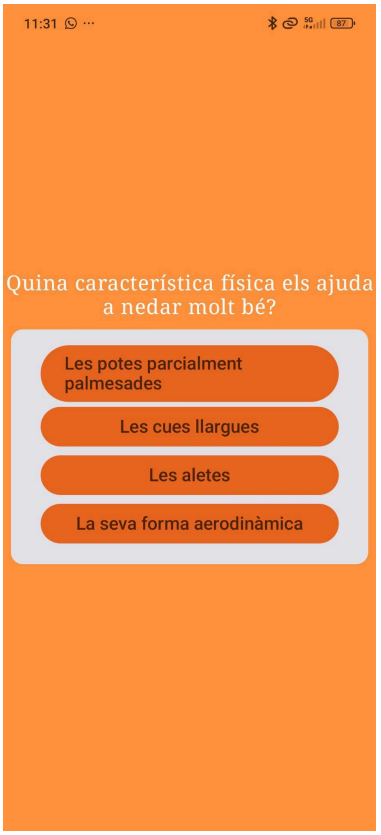


- Segon a la meva tauleta digital he realitzat les connexions i disseny de colors entre altres coses.



- Tercer pluja d'idees.

- Disseny final



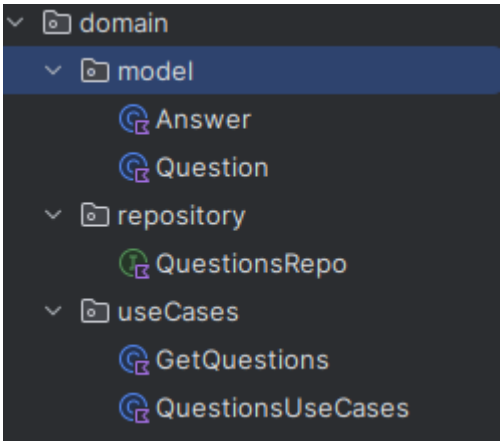
**Depenen de la Puntuació surt

,

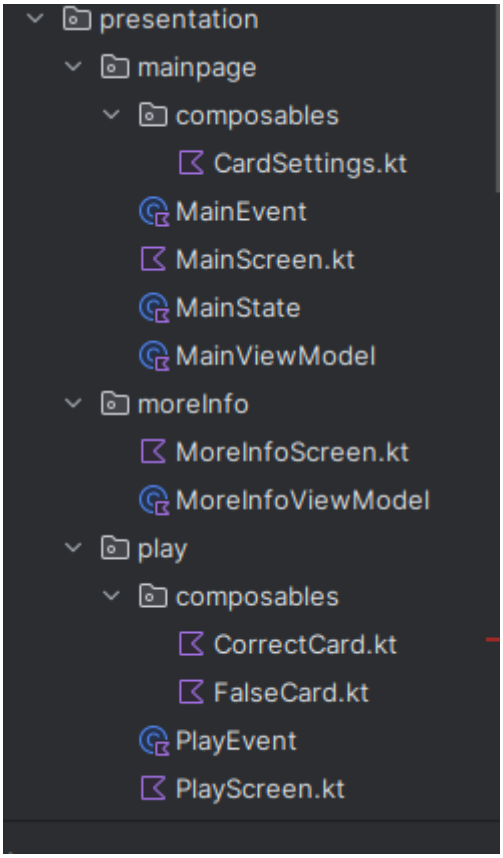
un capi o un altre.

Codi

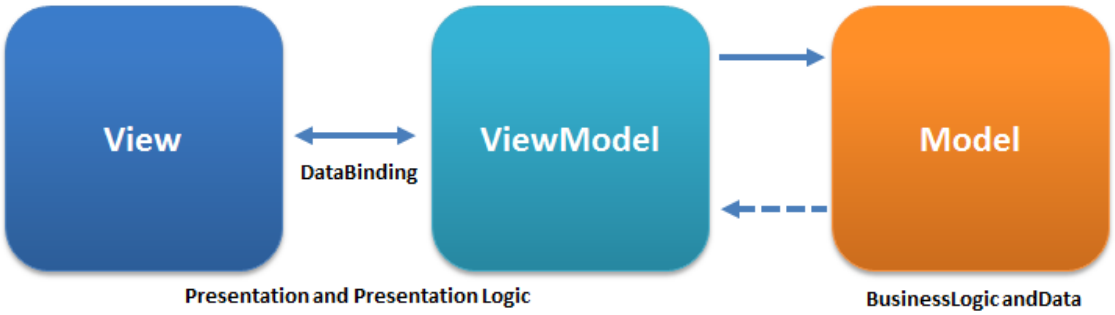
El codi utilitza una Estructura molt coneguda en el món de la programació mòbil. MVVM
Es caracteritza per separar tot en tres classes o grups.
El model, acostumen a ser interfícies amb les estructures de les classes que crearàs.(en el projecte la part model el tinc anomenat com a domini).



La vista o la part visual.



I el ViewModel. Les views són estàtiques i el view model és el que s'encarrega de modificar les views.



Gràfic de com funciona.

(com que el codi és molt extens, aquí explicaré tan sols les coses que més m'han agradat o que més se m'han complicat, per veure més comentaris revisar el codi)

https://github.com/Ferran05/QuizCapibaraApp_MVVM

```
object NumQuestionState {
    var npreguntes: Int by mutableIntStateOf(10)

    fun updateValue(newValue: Int){
        npreguntes = newValue
    }
}
```

He creat un objecte Singleton que totes les views poden veure. Així no es té que passar sempre per variable i no es té que crear una base de dades tant sols per això.

```
package com.amazingapps.appquizmvvm.feature_quiz.presentation.play

class PlayViewModel(
    private val useCases: QuestionsUseCases
): ViewModel() {

    private val _state = mutableStateOf(PlayState())
    val state: State<PlayState> = _state


    private var _currentIndex = MutableStateFlow(0)
    val currentIndex: StateFlow<Int> = _currentIndex

    private val _currentQuestion =
MutableStateFlow(_state.value.questions.firstOrNull())
    val currentQuestion: StateFlow<Question?> = _currentQuestion

    init {
        getQuestions()
    }
```

```
    fun onEvent(event: PlayEvent){
        when(event){
            is PlayEvent.NextQuestio -> {
                nextQuestion()
            }
            is PlayEvent.CardCorrectV -> {
                _state.value = state.value.copy(
                    isCorrectVisible =
!_state.value.isCorrectVisible
                )
            }
            is PlayEvent.CardFalseV -> {
                _state.value = state.value.copy(
                    isFalseVisible = !_state.value.isFalseVisible
                )
            }

            is PlayEvent.CorrectAnswer -> {
                _state.value = state.value.copy(
                    correct =  state.value.correct + 1
                )
            }

            is PlayEvent.ResetPoints -> {
                _state.value = state.value.copy(
                    correct =  0
                )
            }
        }
    }
    private fun getQuestions(){
        viewModelScope.launch {
            var result =
useCases.getQuestions(NumQuestionState.npreguntes)
            Log.w("A", NumQuestionState.npreguntes.toString())
            _state.value = state.value.copy(
                questions = result
            )
        }
    }

    fun nextQuestion(){
        if (_state.value.start){
            _state.value = _state.value.copy(
                start = false
            )
            _currentIndex.value = 0
```

```
            _currentQuestion.value =
_state.value.questions.firstOrNull()
        }
        else if (_currentIndex.value < _state.value.questions.size
- 1){
            _currentIndex.value++
            _currentQuestion.value =
_state.value.questions[_currentIndex.value]
        } else {
            _currentQuestion.value = null
        }
    }

}
```

Aquesta és la classe PlayViewModel. Amb totes les coses que ha de consultar i gestionar trobo que ha quedat molt neta i fàcil d'entendre.
Està composta de tres variables que guarden tres classes d'estats diferents.
La raó per la qual està separat en més d'una Classe és perquè no tots s'actualitzen al mateix temps. Per exemple: No vull que quan és suma un a l'índex de preguntes es carreguin de nou les preguntes. En una aplicació amb tan sols 10 preguntes no donaria problemes, però a la llarga pot arribar a alentir el programa molt.

Tot el tema de la barreja de les preguntes i respostes s'encarrega l'api-servidor d'aquesta forma si algun dia es distribuís l'app i es volgués canviar no s'hauria d'actualitzar en tots els mòbils

Conclusió

Ha sigut un projecte molt divertit, ja que he pogut aprendre algunes coses que no havia aplicat mai en cap altre programa com per exemple crear singletons per guardar variable en temps d'execució. A més de reforçar el meu coneixement en la creació d'app amb l'estructura MVVM. Tinc la confiança de dir que l'aplicació està preparada per ser escalable.