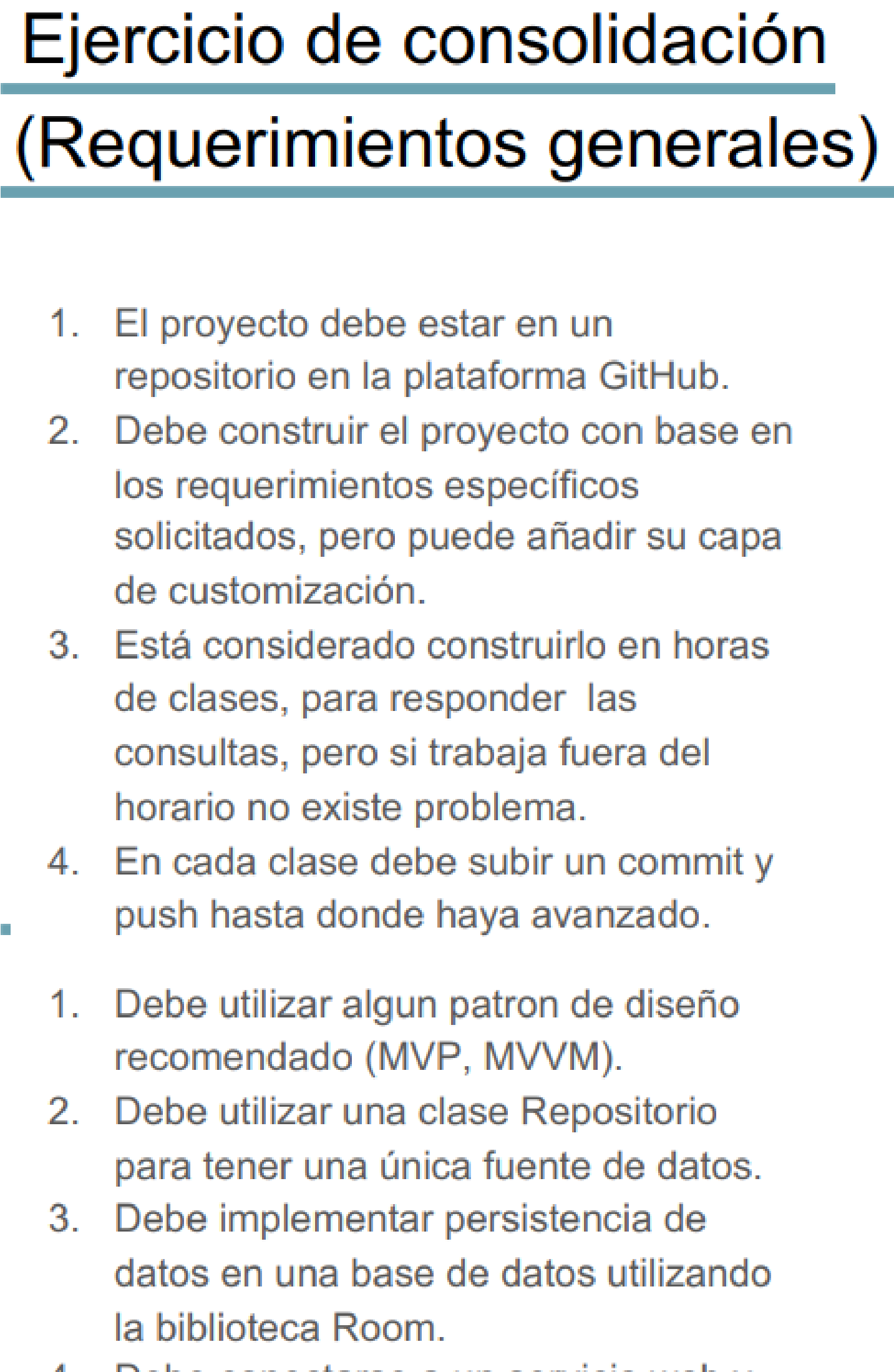
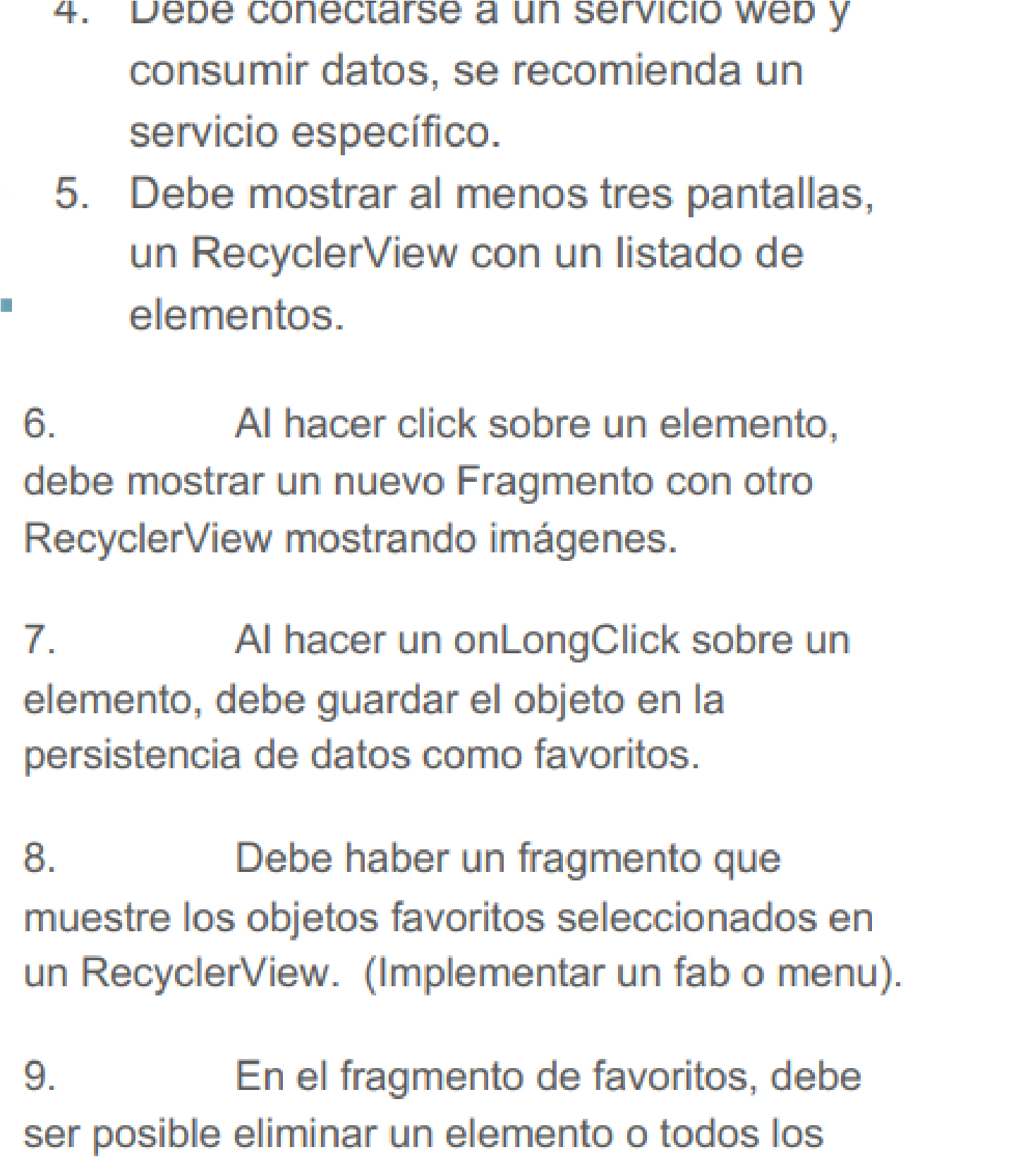
Evidencia día 05 - Semana 15

**Cristian Maturana Ibarra**

**Ejercicio:**





Por mi parte usaré la siguiente API:

[Últimos 15 Sismos en Chile - Centro Sismológico U de Chile](https://api-sismologia-chile.herokuapp.com/)

Muestro acá parte del avance del proyecto, para ver el avance completo revisar en el siguiente [link.](https://github.com/Leonardo-Rodenas/EjercicioActividad12_Consolidacion)

***Modelo:***

@Entity

(tableName =

"tabla\_sismos"

)

data

class

SismosModel

(

@PrimaryKey(autoGenerate = true)

val

id:

Int

,

//Agregada por mi para llevar una id

val

horaLocal: String,

val

horaUtc: String,

val

latitud: String,

val

longitud: String,

val

magnitud: String,

val

mapa: String,

val

profundidad: String,

val

referencia: String,

val

favorito:

Boolean

//agregado por mí para guardar en favorito

)

: Serializable

***Service:***

interface

SismoService

{

@GET(

"."

)

fun

obtenerSismos

()

:

Call<List<SismosModel

>>

}

***Dao:***

@Dao

interface

SismoDao

{

@Insert(onConflict = OnConflictStrategy.REPLACE)

suspend

fun

insertarTodosLosSismos

(

listaDeSismos:

List

<

SismosModel

>)

@Query(

"SELECT \* FROM tabla\_sismos"

)

fun

obtenerTodosLosSismosDeLaBD

()

:

LiveData<List<SismosModel

>>

//1 = true || 0 = false --> revisar las comillas o no

@Query(

"SELECT \* FROM tabla\_sismos where favorito='1'"

)

fun

obtenerLosSismosFavoritosDeLaBD

()

:

LiveData<List<SismosModel

>>

//ejecutado en consecuencia de si es favorito o no

@Update

fun

cambiaEstadoFavorito

(

favorito:

SismosModel

)

//este borra

@Query(

"UPDATE tabla\_sismos set favorito='0' where favorito='1'"

)

fun

borraFavorito

()

}

***Repositorio:***

class

SismosRepositorio

(

private

val

sismosDao: SismoDao) {

private

val

service = SismoCliente.obtenCliente()

val

miLiveData = sismosDao.obtenerTodosLosSismosDeLaBD()

//De Api a la base de datos

fun

obtenDataDelServer

()

{

val

call = service.obtenerSismos()

call.enqueue(

object

: Callback<List<SismosModel>> {

override

fun

onResponse

(

call:

Call

<

List

<

SismosModel

>>

,

response:

Response

<

List

<

SismosModel

>>

)

{

CoroutineScope(Dispatchers.IO).launch {

response.body()?.let {

Log.v(

"logenrepo"

, response.body().toString())

sismosDao.insertarTodosLosSismos(it)

}

}

}

override

fun

onFailure

(

call:

Call

<

List

<

SismosModel

>>

, t:

Throwable

)

{

call.cancel()

}

})

}

//para mostrar los sismos en la base de datos

fun

exponeSismosDeLaBaseDeDatos

()

:

LiveData<List<SismosModel>>

{

return

sismosDao.obtenerTodosLosSismosDeLaBD()

}

//para mostrar los FAVORITOS en la base de datos

fun

exponeFavoritosDeLaBaseDeDatos

()

:

LiveData<List<SismosModel>>

{

return

sismosDao.obtenerLosSismosFavoritosDeLaBD()

}

//este cambia el estado de si es favorito o no, pero tendria que cambiarlo acá

no? o en el viewmodel?

fun

cambiaeEstadoDeFavorito

(

sismo

:

SismosModel

)

{

sismosDao.cambiaEstadoFavorito(sismo)

}

//para BORRAR los FAVORITOS en la base de datos

fun

borrarFavoritosDeLaBaseDeDatos

()

{

sismosDao.borraFavorito()

}

}

**Reflexión:**

Clase de finalización de presentación del tema hilos - corrutinas. Se presenta y se da la clase para realizar el ejercicio de la Lección 12 - Consolidación. Por mi parte usé una Api distinta a la del ejercicio para dar un poco de variación a los proyectos. Estuvo buena las clases, pero la modalidad de preguntar unas cosas y que me respondan con 5 y quedar con más dudas de las que tenía no me agrada del todo.