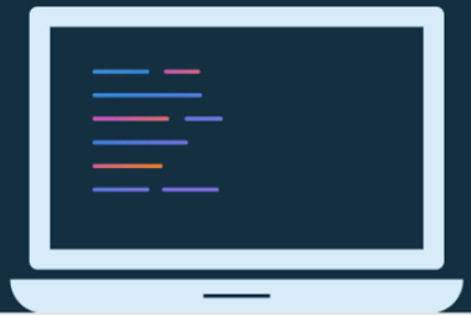
Bài học 12: Mẫu kho lưu trữ và WorkManager



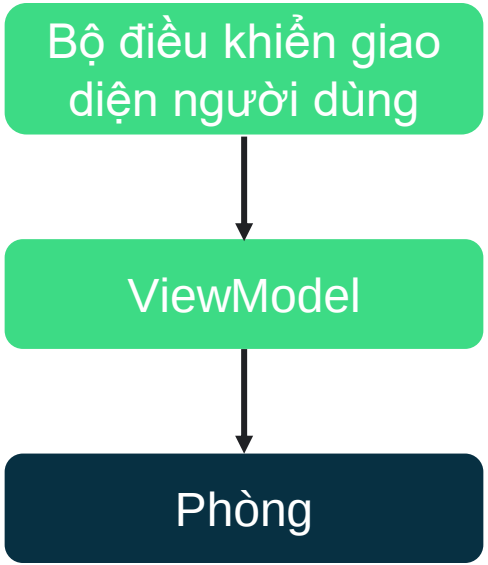
Giới thiệu về bài học này

Bài học 12: Mẫu kho lưu trữ và WorkManager

- Mẫu kho lưu trữ
- WorkManager
- Đầu vào và đầu ra của tác vụ
- Các hạn chế đối với WorkRequest
- Tóm tắt

Mẫu kho lưu trữ

Cấu trúc hiện có của ứng dụng



Tốc độ tương đối của dữ liệu

Thao tác	Tốc độ tương đối
Đọc từ `LiveData`	NHANH
Đọc từ cơ sở dữ liệu `Phòng`	CHẬM
Đọc từ mạng	CHẬM NHẤT

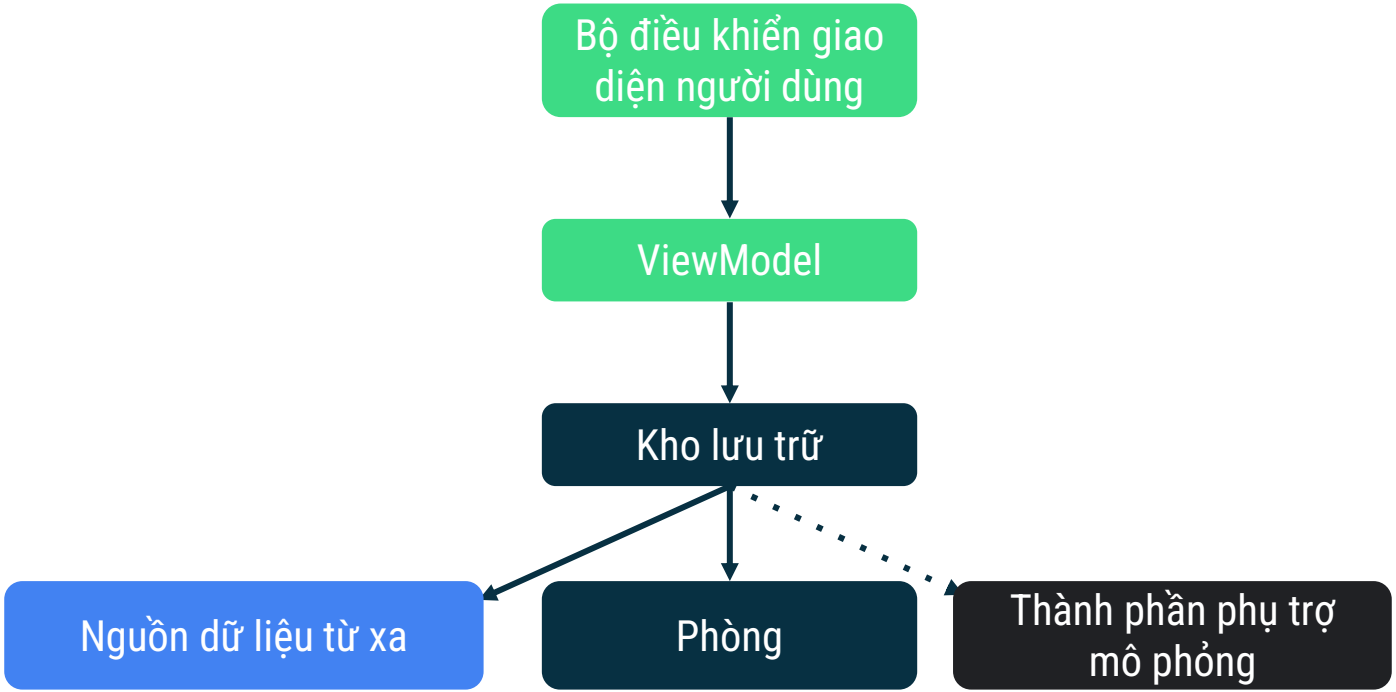
Lưu các phản hồi của mạng vào bộ nhớ đệm

- Xem xét đến thời gian yêu cầu mạng lâu, trong khi vẫn phải nhanh chóng phản hồi người dùng
- Việc dùng `Phòng` cục bộ khi truy xuất từ mạng có thể gây tốn kém hoặc khó khăn
- Có thể lưu vào bộ nhớ đệm dựa trên giá trị ít dùng gần đây nhất (LRU), giá trị truy cập thường xuyên (FRU) hoặc thuật toán khác

Mẫu kho lưu trữ

- Có thể trừu tượng hóa nhiều nguồn dữ liệu trong phương thức gọi
- Hỗ trợ truy xuất nhanh bằng cách dùng cơ sở dữ liệu cục bộ trong khi gửi yêu cầu mạng để làm mới dữ liệu (có thể mất nhiều thời gian hơn)
- Có thể kiểm tra nguồn dữ liệu riêng biệt với các chương trình thành phần khác của ứng dụng

Cấu trúc ứng dụng có mẫu kho lưu trữ



Triển khai một lớp kho lưu trữ

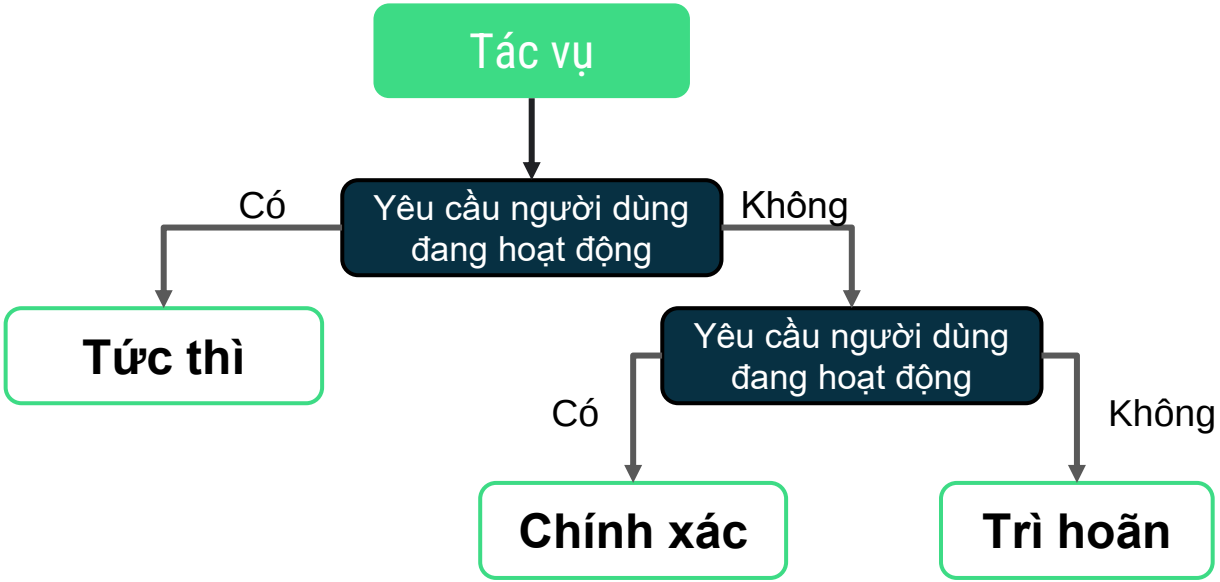
- Cung cấp một giao diện chung để truy cập vào dữ liệu:
  - Hiển thị các hàm để truy vấn và sửa đổi dữ liệu cơ bản
- Tùy vào các nguồn dữ liệu của bạn, kho lưu trữ có thể:
  - Lưu giữ thông tin tham chiếu đến DAO, nếu dữ liệu của bạn nằm trong cơ sở dữ liệu
  - Thực hiện các yêu cầu nối mạng nếu bạn kết nối với một dịch vụ web

WorkManager

WorkManager

- Thành phần cấu trúc của Android Jetpack
- Giải pháp đề xuất để thực thi tác vụ trong nền (tức thì hoặc trì hoãn)
- Thực thi theo cơ hội và được đảm bảo
- Việc thực thi có thể dựa trên các điều kiện nhất định

Trường hợp sử dụng WorkManager



Khai báo các phần phụ thuộc của WorkManager

```
implementation "androidx.work:work-runtime-ktx:$work_version"
```

Các lớp quan trọng cần biết

- `Worker` – thực hiện tác vụ trên một luồng trong nền, ghi đè phương thức `doWork()`
- `WorkRequest` – yêu cầu thực hiện tác vụ nào đó
- `Constraint` – điều kiện về thời điểm có thể chạy tác vụ
- `WorkManager` – lên lịch chạy `WorkRequest`

Xác định tác vụ

```
class UploadWorker(appContext: Context, workerParams: WorkerParameters) :
        Worker(appContext, workerParams) {

    override fun doWork(): Result {

        // Do the work here. In this case, upload the images.
        uploadImages()

        // Indicate whether work finished successfully with the Result
        return Result.success()
    }
}
```

Mở rộng CoroutineWorker thay vì Worker

```
class UploadWorker(appContext: Context, workerParams: WorkerParameters) :
        CoroutineWorker(appContext, workerParams) {

    override suspend fun doWork(): Result {

        // Do the work here (in this case, upload the images)
        uploadImages()

        // Indicate whether work finished successfully with the Result
        return Result.success()
    }
}
```

WorkRequest

- Có thể được lên lịch để chạy một lần hoặc nhiều lần
  - `OneTimeWorkRequest`
  - `PeriodicWorkRequest`
- Vẫn giữ nguyên trong các lần khởi động lại thiết bị
- Có thể được xâu chuỗi để chạy tuần tự hoặc song song
- Có thể chứa các hạn chế đối với việc chạy

Lên lịch OneTimeWorkRequest

Tạo `WorkRequest`:

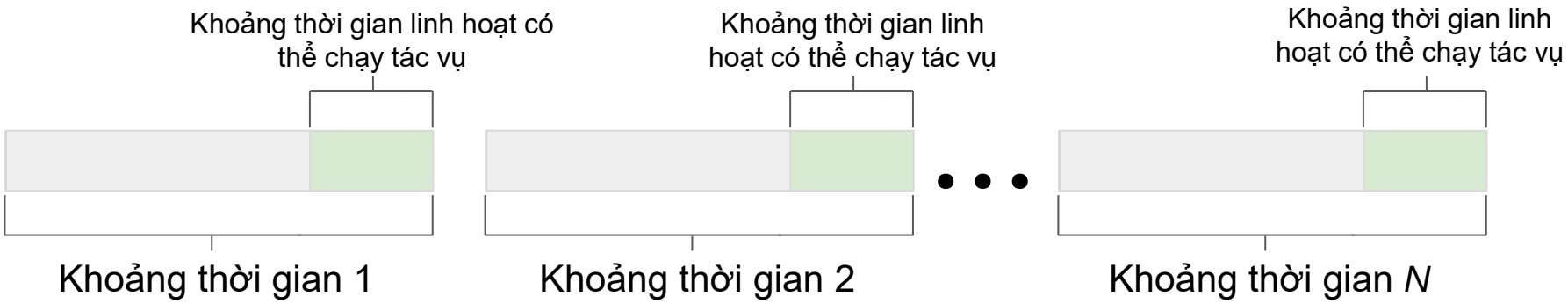
```
val uploadWorkRequest: WorkRequest =
   OneTimeWorkRequestBuilder<UploadWorker>()
       .build()
```

Thêm tác vụ vào hàng đợi `WorkManager`:

```
WorkManager.getInstance(myContext)
    .enqueue(uploadWorkRequest)
```

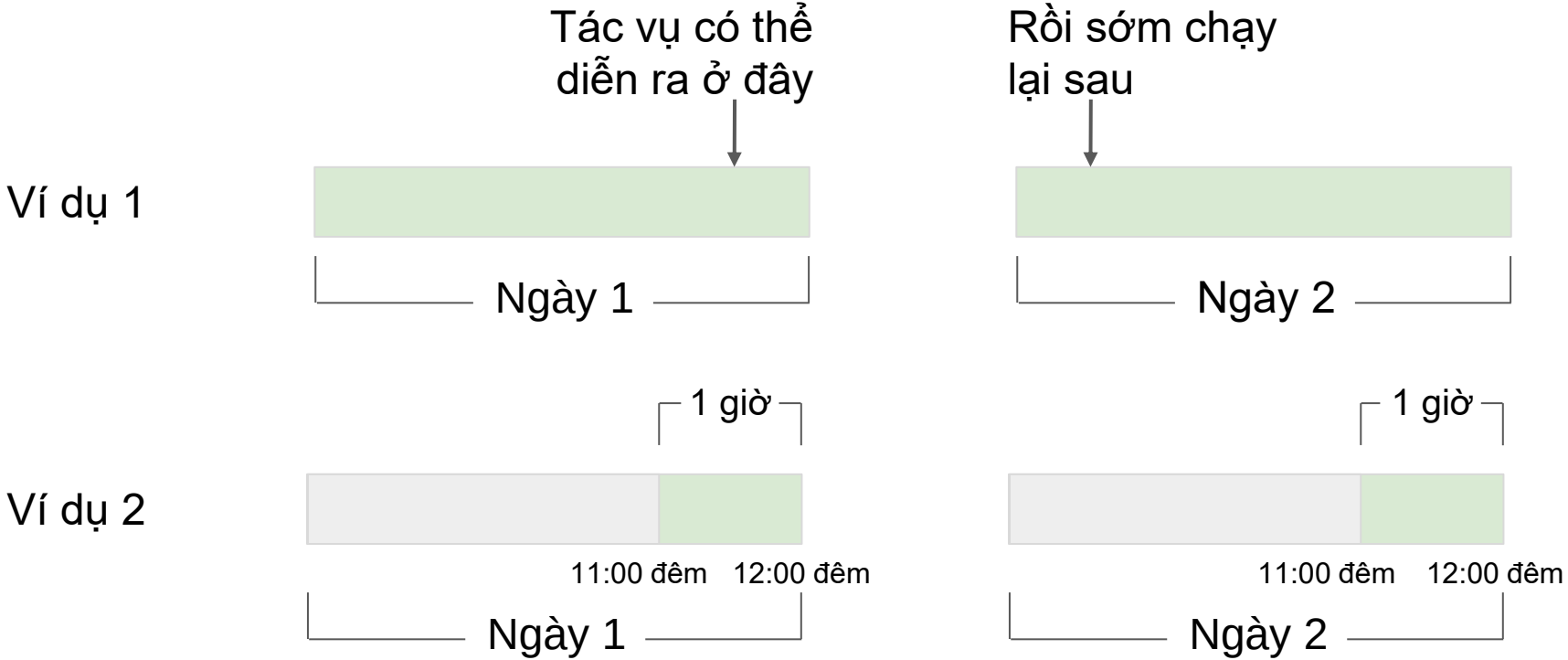
Lên lịch PeriodicWorkRequest

- Đặt khoảng thời gian lặp lại
- Đặt khoảng thời gian linh hoạt (không bắt buộc)



Chỉ định khoảng thời gian bằng `TimeUnit` (ví dụ: `TimeUnit.HOURS`, `TimeUnit.DAYS`)

Khoảng thời gian linh hoạt



Ví dụ về PeriodicWorkRequest

```
val repeatingRequest = PeriodicWorkRequestBuilder<RefreshDataWorker>(
        1, TimeUnit.HOURS,     // repeatInterval
        15, TimeUnit.MINUTES   // flexInterval
    ).build()
```

Thêm tác vụ định kỳ vào hàng đợi

```
WorkManager.getInstance().enqueueUniquePeriodicWork(
    "Unique Name",
    ExistingPeriodicWorkPolicy.KEEP, // or REPLACE
    repeatingRequest
)
```

Đầu vào và đầu ra của tác vụ

Xác định Worker với đầu vào và đầu ra

```
class MathWorker(context: Context, params: WorkerParameters):
      CoroutineWorker(context, params)  {

    override suspend fun doWork(): Result {
        val x = inputData.getInt(KEY_X_ARG, 0)
        val y = inputData.getInt(KEY_Y_ARG, 0)
        val result = computeMathFunction(x, y)
        val output: Data = workDataOf(KEY_RESULT to result)
        return Result.success(output)
    }
}
```

Đầu ra kết quả từ doWork()

Trạng thái kết quả	Trạng thái kết quả có đầu ra
`Result.success()`	`Result.success(output)`
`Result.failure()`	`Result.failure(output)`
`Result.retry()`	

Gửi dữ liệu đầu vào cho Worker

```
val complexMathWork = OneTimeWorkRequest<MathWorker>()
    .setInputData(
        workDataOf(
            "X_ARG" to 42,
            "Y_ARG" to 421,
        )
    ).build()

WorkManager.getInstance(myContext).enqueue(complexMathWork)
```

Các hạn chế đối với WorkRequest

Constraints

- `setRequiredNetworkType`
- `setRequiresBatteryNotLow`
- `setRequiresCharging`
- `setTriggerContentMaxDelay`
- `requiresDeviceIdle`

Ví dụ về Constraints

```
val constraints = Constraints.Builder()
    .setRequiredNetworkType(NetworkType.UNMETERED)
    .setRequiresCharging(true)
    .setRequiresBatteryNotLow(true)
    .setRequiresDeviceIdle(true)
    .build()
```

Tóm tắt

Tóm tắt

Trong Bài học 12, bạn đã tìm hiểu cách:

- Dùng kho lưu trữ để trừu tượng hóa lớp dữ liệu trong phần còn lại của ứng dụng
- Lên lịch các tác vụ trong nền một cách hiệu quả và tối ưu hóa bằng `WorkManager`
- Tạo các lớp `Worker` tùy chỉnh để chỉ định tác vụ cần thực hiện
- Tạo và thêm các yêu cầu tác vụ một lần hoặc định kỳ vào hàng đợi

Tìm hiểu thêm

- Tìm nạp dữ liệu
- Lên lịch các tác vụ bằng WorkManager
- Xác định yêu cầu tác vụ
- Kết nối ViewModel và kho lưu trữ
- Dùng WorkManager để thực thi tức thì trong nền

Lộ trình

Thực hành những gì bạn đã học được bằng cách hoàn thành lộ trình này:

Bài học 12: Mẫu kho lưutrữ và WorkManager