ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

TẠ XUÂN KHIÊM

TÌM HIỂU VÀ ĐÁNH GIÁ KỸ THUẬT MÔ HÌNH HÓA LUỒNG TƯƠNG TÁC IFML TRONG PHÁT TRIỂN ỨNG DỤNG DI ĐỘNG

LUẬN VĂN THẠC SĨ CÔNG NGHỆ THÔNG TIN

Hà Nội - 2016

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

TẠ XUÂN KHIÊM

TÌM HIỂU VÀ ĐÁNH GIÁ KỸ THUẬT MÔ HÌNH HÓA LUỒNG TƯƠNG TÁC IFML TRONG PHÁT TRIỂN ỨNG DỤNG DI ĐỘNG

Ngành: Công nghệ thông tin

Chuyên ngành: Kỹ thuật Phần mềm

Mã số: 60480103

LUẬN VĂN THẠC SĨ CÔNG NGHỆ THÔNG TIN

NGƯỜI HƯỚNG DẪN KHOA HỌC: TS. ĐẶNG ĐỨC HẠNH

Hà Nội - 2016

LỜI CAM ĐOAN

Tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi, được thực hiện dưới sự hướng dẫn khoa học của ***Ts. Đặng Đức Hạnh.***

Nội dung nghiên cứu và kết quả nêu trong luận văn là hoàn toàn trung thực, được tôi tổng hợp, bổ sung và biên soạn theo sự hiểu biết của mình sau khi nghiên cứu được từ các tài liệu tham khảo như sách, bài báo khoa học, luận văn và dữ liệu từ các trang Web uy tín.

Hà nội, tháng 10, năm 2016

Học viên

(Ký và ghi rõ họ tên)

LỜI CẢM ƠN

Lời đầu tiên, tôi xin dành lời cảm ơn sâu sắc nhất đến **TS. Đặng Đức Hạnh** - Giảng viên bộ môn Công nghệ Phần mềm - Khoa Công nghệ Thông tin - Trường Đại học Công nghệ - Đại học Quốc gia Hà Nội, người đã trực tiếp định hướng và hướng dẫn tôi hoàn thành luận văn này. Từ những ngày đầu còn mơ hồ với lĩnh vực nghiên cứu mới, tôi đã được thầy tận tình quan tâm, hướng dẫn để có thể tiếp cận nhanh với lĩnh vực mới, công nghệ mới. Và bây giờ, sau khi trải qua giai đoạn nghiên cứu, tìm hiểu và vẫn dụng tôi cũng đã thu được những kiến thức mới mẻ và bổ ích được trình bày trong luận văn này.

Ngoài ra, trong khoảng thời gian học tập và nghiên cứu tại Trường Đại học Công nghệ - ĐHQGHN, với sự giảng dạy, chỉ bảo tận tình của các Thầy/Cô và các bạn học viên, tôi đã học được rất nhiều điều bổ ích, không chỉ trong kiến thức công việc, học tập mà còn trong cuộc sống. Tôi đặc biệt ấn tượng với khả năng phân tích vấn đề, đưa ra lời khuyên, kết luận đúng đắn một cách nhanh chóng và khoa học của các Thầy/Cô và các bạn. Tôi xin được gửi lời cảm ơn chân thành và sâu sắc tới các Thầy/Cô và các bạn. Tôi cũng xin được gửi lời cảm ơn tới gia đình đã luôn động viên tôi hoàn thành tốt nhiệm vụ học tập được giao.

Do lĩnh vực nghiên cứu được đề cập trong luận văn còn mới - đang trong quá trình phát triển, chưa được ứng dụng rộng rãi ở Việt Nam và trên thế giới, cho nên tôi đã gặp không ít khó khăn trong việc nghiên cứu, vận dụng. Giới hạn về thời gian, áp lực công việc cũng là vấn đề lớn khiến tôi chưa tập trung tâm huyết, trí lực để khai thác các vấn đề một cách chuyên sâu hơn nữa. Vì vậy mà chắc chắn luận văn sẽ còn nhiều điều thiếu sót, rất mong nhận được ý kiến đóng góp quý báu của các Thầy/Cô và bạn đọc quan tâm. Mọi ý kiến đóng góp xin vui lòng về địa chỉ thử điện tử : khiemtx@gmail.com.

Xin chân thành cảm ơn

Học viên

Tạ Xuân Khiêm

MỤC LỤC

DANH MỤC CÁC KÝ HIỆU VÀ CÁC CHỮ VIẾT TẮT

API	Application Programming Interface
CIM	Computation Indepedent Model
CSS	Cascading Style Sheets
DSL	Domain Specific Language
HTML	HyperText Markup Language
IDE	Integrated Development Environment
IFML	Interaction Flow Modeling Language
M2M	Model to Model
M2T	Model to Text
MDA	Model Driven Architecture
MDD	Model Driven Development
MDE	Model Driven Engineering
MDSD	Model Driven Software Development
MOF	Meta Object Facility
MVC	Model View Controller
OMG	Object Management Group
PDM	Platform Definition Model
PIM	Platform Independent Model
PSM	Platform Specific Model
SDK	Software Development Kit
UI	User Interface
UML	Unified Modeling Language
UX	User eXperience
XML	eXtensible Markup Language

DANH MỤC CÁC HÌNH ẢNH VÀ ĐỒ THỊ

DANH MỤC CÁC BẢNG BIỂU

MỞ ĐẦU

Đặt vấn đề

Trong bối cảnh bùng nổ về công nghệ thông tin hiện nay, ngành công nghiệp phần mềm đang ngày càng phát triển mạnh mẽ ở Việt Nam và trên toàn thế giới. Đặc biệt trong lĩnh vực lập trình ứng dụng di động, số lượng thiết bị điện thoại di động thông minh đã đạt tới 1.9 tỷ thiết bị vào năm 2015 [26] và vẫn trên đà phát triển nhanh chóng. Cộng đồng lập trình viên hoạt động trên lĩnh vực ứng dụng di động chiếm một phần không nhỏ trong đồng phát triển phần mềm nói chung. Nhu cầu sử dụng điện thoại di động tăng cao mang đến lợi nhuận to lớn và kéo theo tính cấp thiết về vấn đề cải thiện quy trình, phương pháp, tốc độ và chi phí phát triển phần mềm. Tuy nhiên, vẫn còn nhiều trở ngại còn tồn tại trong quá trình phát triển phần mềm nói chung và phần mềm di động nói riêng:

- *Sự đa dạng về nền tảng*: Số lượng thiết bị phần cứng khổng lồ được hoạt động trên hơn 10 nền tảng khác nhau như: Android, iOS, Windows Phone... Điều này kéo theo sự không nhất quán trong quá trình phát triển phần mềm. Các nhà phát triển phải duy trì nhiều đội ngũ độc lập nhau cho việc phát triển ứng dụng trên mỗi nền tảng.
- *Tốc độ phát triển nhanh chóng của công nghệ*: Các công nghệ mới liên tục được giới thiệu dẫn đến sự lạc hậu nhanh chóng của các ứng dụng phần mềm. Điều này đòi hỏi nhà phát triển phải thực sự linh động trong quá trình nâng cấp, bảo trì sản phẩm.
- *Sự thay đổi về tương tác người dùng*: Theo đà phát triển công nghệ điện thoại di động thông minh, các tính năng mới liên tục được giới thiệu cùng với sự đổi mới trong tương tác người dùng. Các tương tác này ngày càng đa dạng và chuyên biệt trong lĩnh vực di động, việc giao tiếp giữa người dùng và hệ thống không còn chỉ gói gọn trong các thao tác cơ bản mà còn bao gồm nhiều tương tác phức tạp khác như: trượt (slide), chạm (touch), cảm ứng lực (force touch), nhấn giữ (long press), điều khiển bằng ánh mắt (eye control)...
- *Khó nắm bắt yêu cầu của khách hàng*: Các nhà phát triển và khách hàng thường khó đồng nhất về yêu cầu ứng dụng. Điều này xuất phát từ đặc thù lĩnh vực của hai bên, nhà phát triển không nắm rõ nghiệp vụ khách hàng để

đưa ra những đề xuất đúng đắn, khách hàng không hiểu rõ về kiến thức phần mềm dẫn đến sự nhập nhằng trong yêu cầu.

- *Sự thay đổi về yêu cầu*: Sự thay đổi, chỉnh sửa yêu cầu có thể diễn ra bất cứ lúc nào, cả trước và sau khi phát triển sản phẩm nhằm thay đổi, tích hợp những tính năng mới để đáp ứng với sự thay đổi của nghiệp vụ, công nghệ. Sự thay đổi này thường khiến cho thời gian, chi phí phát triển sản phẩm nâng cao lên nhiều so với ước tính ban đầu.

Ngoài ra, các ứng dụng, ý tưởng đến từ cộng đồng phát triển to lớn được giới thiệu tính bằng giờ, thời gian phát triển kéo dài quá lâu có thể mất đi lợi thế về ý tưởng, sản phẩm đi đầu, thời gian phát triển ứng dụng quá dài có thể dẫn đến sự lạc hậu về công nghệ. Mục tiêu cần đạt được là xây dựng một ứng dụng cho tất cả các thiết bị, hệ điều hành với chi phí thấp nhất, thời gian phát triển nhanh nhất mà vẫn đảm bảo được chất lượng sản phẩm.

Mô hình hóa luồng tương tác IFML là kỹ thuật mới được giới thiệu bởi WebRatio và trở thành chuẩn OMG vào năm 2013, công cụ cho phép xây dựng ứng dụng di động bằng kỹ thuật IFML được WebRatio ra mắt vào cuối năm 2014 trở thành một xu hướng mới trong phát triển ứng dụng di động đa nền tảng. Là một kỹ thuật cho phép phát triển ứng dụng di động hướng mô hình và thừa hưởng toàn bộ ưu điểm của phương pháp này, IFML dần trở nên phổ biến trong phát triển phần mềm, đặc biệt là trên lĩnh vực di động. Tuy nhiên, do thời gian giới thiệu tới cộng đồng phát triển ứng dụng còn ngắn, chưa có nhiều tài liệu, nghiên cứu về IFML và tính ứng dụng trong phát triển phần mềm di động trên toàn thế giới. Đây là một trở ngại không nhỏ cho các nhà nghiên cứu, phát triển muốn tìm hiểu và ứng dụng IFML. Điều này dẫn đến một yêu cầu cấp thiết cho việc ra đời một nghiên cứu nhằm tìm hiểu bản chất và nguyên lý của công nghệ mô hình hóa luồng tương tác IFML cũng như khảo sát khả năng vận dụng của IFML trong thực tế, đặc biệt là lớp ứng dụng di động. Tất cả các vấn đề trên là động lực thúc đẩy tôi lựa chọn và nghiên cứu đề tài *"Tìm hiểu và đánh giá kỹ thuật mô hình hóa luồng tương tác IFML trong phát triển ứng dụng di động"*

Phạm vi nghiên cứu

Luận văn tập trung nghiên cứu và trình bày về các nguyên lý cơ bản và khả năng vận dụng trong lớp ứng dụng di động của kỹ thuật mô hình hóa luồng tương tác IFML. Qua đó đề xuất một số tiêu chí nhằm đánh giá kỹ thuật mô hình hóa luồng tương tác IFML trong phát triển ứng dụng di động.

Để thực hiện đánh giá các tiêu chí được đề xuất, luận văn trình bày các kiến thức chung về lập trình ứng dụng di động trên nền tảng Android, thực hiện xây dựng ứng dụng di động MealNote sử dụng hai phương pháp song song: Phương pháp truyền thống với kỹ thuật xây dựng ứng dụng gốc sử dụng mã nguồn Java trên công cụ Android Studio và phương pháp hướng mô hình với kỹ thuật mô hình hóa luồng tương tác IFML. Qua đó đưa ra các đánh giá chi tiết và trực quan theo các tiêu chí đã đề cập về kỹ thuật mô hình hóa luồng tương tác IFML trong phát triển ứng dụng di động.

Cấu trúc luận văn

Luận văn được cấu trúc như sau:

Phần mở đầu: Giới thiệu về tình hình phát triển phần mềm di động hiện nay, đặt vấn đề và đưa ra định hướng nghiên cứu.

Chương 1: Giới thiệu về phát triển ứng dụng hướng mô hình và lập trình ứng dụng di động. Trình bày về xu thế lập trình di động đa nền tảng hiện hay và các công cụ hỗ trợ.

Chương 2: Nghiên cứu kỹ thuật mô hình hóa luồng tương tác nói chung và kỹ thuật mô hình hóa luồng tương tác trong phát triển ứng dụng di động nói riêng, phạm vi ứng dụng và khả năng ứng dụng trong phát triển phần mềm đa nền tảng cho điện thoại di động thông minh.

Chương 3: Vận dụng, thực nghiệm và đánh giá kỹ thuật mô hình hóa luồng tương tác IFML trong phát triển ứng dụng di động so sánh với phương pháp truyền thống - Lập trình Android sử dụng Java trên công cụ Android Studio. Xây dựng ứng dụng di động sử dụng hai kỹ thuật song song: Xây dựng ứng dụng di động với IFML và xây dựng ứng dụng gốc (Java - Android Studio).

Kết luận: Đưa ra tổng kết về đề tài, các đề xuất và hướng nghiên cứu tiếp theo.

CHƯƠNG 1: KIẾN THỨC NỀN TẢNG

Chương 1 giới thiệu cơ sở lý thuyết cho luận văn bao gồm phương pháp phát triển phần mềm hướng mô hình và kiến thức về lập trình ứng dụng di động trên nền tảng Android nhằm vận dụng, xây dựng ứng dụng thực nghiệm phục vụ cho kết quả chính của luận văn.

1.1. Giới thiệu

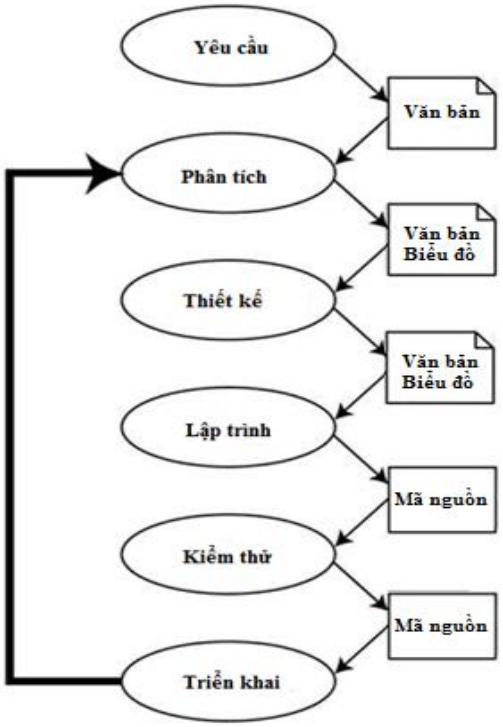
Phương pháp phát triển phần mềm hướng mô hình (MDD - Model Driven Development) được biết đến như một phương pháp giúp nâng cao hiệu quả trong phát triển phần mềm. Phương pháp này được áp dụng rộng rãi trong phát triển phần mềm nói chung nhưng chưa thực sự phổ biến trong lĩnh vực lập trình ứng dụng di động. MDD được giới thiệu trong chương này nhằm cung cấp kiến thức nền tảng về kỹ thuật xây dựng ứng dụng hướng mô hình.

Thêm vào đó, tác giả thực hiện đánh giá kỹ thuật mô hình hóa luồng tương tác bằng các xây dựng ứng dụng di động Android song song: xây dựng ứng dụng gốc và xây dựng ứng dụng với IFML. Kiến thức về lập trình ứng dụng di động nói chung và lập trình ứng dụng di động trên Android nói riêng được đề cập đến với mục tiêu phục vụ quá trình xây dựng ứng dụng gốc trong các phần tiếp theo.

1.2. Tổng quan phương pháp phát triển phần mềm truyền thống

Theo sự phát triển của công nghệ, nhu cầu về phần mềm ngày càng gia tăng, mô hình phát triển phần mềm truyền thống vẫn được áp dụng phổ biến trong quá trình xây dựng, phát triển phần mềm.

Phát triển phần mềm theo phương pháp truyền thống là sự lựa chọn chính cho các nhà phát triển phần mềm do đã dần hoàn thiện qua quá trình dài sử dụng và kiểm chứng. Tuy nhiên, với sự phát triển mạnh mẽ của công nghệ phần mềm, phương pháp phát triển truyền thống dần lộ ra những điểm bất cập cần phải khắc phục. Các giai đoạn phát triển một phần mềm theo phương pháp truyền thống được mô tả trong Hình 1.1.



Hình 1.1: Quy trình phát triển phần mềm theo phương pháp truyền thống

Theo Hình 1.1, các tài liệu thể hiện trong các giai đoạn phát triển phân tích, thiết kế đều ở dạng văn bản, hình ảnh minh họa hay một số biểu đồ UML như biểu đồ ca sử dụng, biểu đồ tương tác, biểu đồ lớp, biểu đồ hoạt động... Giai đoạn lập trình với mã nguồn được tham chiếu theo các tài liệu phân tích thiết kế, tuy nhiên, việc thay đổi yêu cầu là vấn đề luôn gặp phải trong phát triển phần mềm. Các yêu cầu thay đổi về nghiệp vụ, tính năng đi theo trong suốt quá trình xây dựng hệ thống. Việc thay đổi này dẫn đến phải sửa đổi toàn bộ tài liệu từ những giai đoạn ban đầu, dẫn đến sự chậm trễ về tiến độ phát triển, sự tiêu tốn về chi phí, sự không đồng nhất giữa tài liệu phân tích thiết kế và ứng dụng thực tế. Vấn đề này còn tiếp tục nảy sinh trong quá trình triển khai, bảo trì và nâng cấp phần mềm.

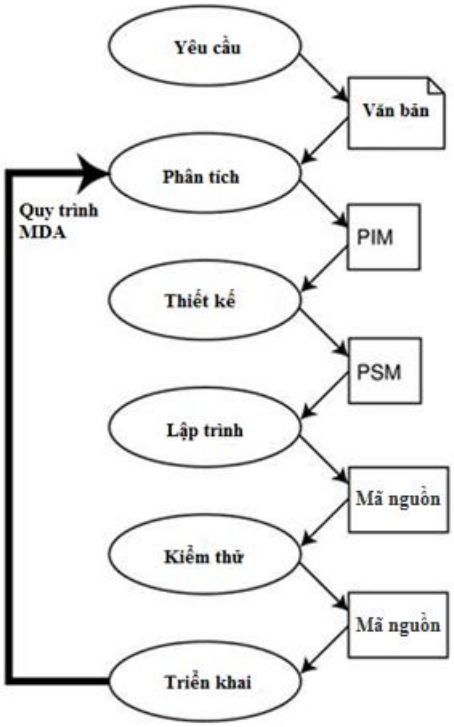
Thêm vào đó, một hạn chế rất lớn của phương pháp phát triển phần mềm truyền thống nằm ở khả năng tương thích với môi trường cài đặt. Với sự tồn tại của nhiều nền tảng hiện nay, nhà phát triển phải xây dựng mỗi nền tảng một phần mềm tương ứng do một sản phẩm phần mềm hầu như không có khả năng tương thích trên nhiều nền tảng. Sự tốn kém về chi phí, thời gian, công sức phát triển dẫn đến nhu cầu về một phương pháp phát triển phần mềm mới cho phép cải thiện các vấn đề còn tồn tại đã nêu.

1.3. Phương pháp phát triển phần mềm hướng mô hình

Với phương pháp phát triển truyền thống, mỗi nền tảng quy định mô hình xây dựng gốc của nó sử dụng các mã nguồn gốc như Java, Objective - C, C/C++... Sự đặc trưng của các ngôn ngữ gốc ảnh hưởng trực tiếp đến sự phát triển, chất lượng, hiệu quả, khả năng bảo trì, khả năng tương tác và tính khả chuyển của phần mềm là những yếu tố quan trọng để đánh giá tổng thể một ứng dụng phần mềm. Mức độ trừu tượng của ngôn ngữ gốc là rất thấp dẫn đến việc cụ thể hóa mạnh mẽ ở cấp độ ngôn ngữ. Đây cũng là một trong những khó khăn lớn nhất khi nhà phát triển muốn xây dựng ứng dụng phần mềm trên các nền tảng khác nhau.

1.3.1. Giới thiệu

Sự ra đời của phương pháp phát triển phần mềm hướng mô hình (MDSD - Model - Driven Software Development) như là một giải pháp cho các vấn đề gặp phải trong quá trình phát triển phần mềm theo phương pháp truyền thống. Sử dụng kiến trúc hướng mô hình (MDA - Model Driven Development) cho phương pháp MDSD, các giai đoạn phát triển phần mềm được thể hiện trong Hình 1.2:



Hình 1.2: Quy trình phát triển phần mềm hướng mô hình

Các giai đoạn phát triển phần mềm với MDA không khác biệt so với phương pháp truyền thống, điều cốt lõi là các kết quả của mỗi giai đoạn, các tài liệu được đặc tả bởi các mô hình hình thức, do vậy tính khả chuyển giữa các mô hình cao cho

phép rút ngắn thời gian, chi phí của mỗi giai đoạn. Thông qua các công cụ chuyển đổi mô hình, từ mô hình độc lập nền (PIM - Platform Independent Model) có thể chuyển đổi thành các mô hình phụ thuộc nền (PSM - Platform Specific Model) và chuyển đổi thành các mã nguồn hoặc tài liệu.

Cách tiếp cận phát triển phần mềm hướng mô hình là việc sử dụng các mô hình trừu tượng của hệ thống phần mềm trong quá trình xây dựng phần mềm. Các mô hình có tính trừu tượng và hình thức hơn giúp thể hiện bản chất của vấn đề. Quy trình này được vận dụng trong hai cách cơ bản: Tạo ra các thể hiện cuối như ngôn ngữ lập trình và lược đồ dữ liệu (M2T - Model to Text) hoặc chuyển đổi thành các mô hình khác có ngữ nghĩa rõ ràng và sẵn sàng tạo thành các hệ thống hoàn chỉnh (M2M - Model to Model). Điều này không chỉ dừng lại ở việc tạo ra "bộ khung" của ứng dụng mà còn hướng tới mục tiêu tạo thành một ứng dụng hoàn chỉnh.

Ưu điểm

MDSD mang đến nhiều lợi ích khắc phục các nhược điểm của phương pháp phát triển phần mềm truyền thống, các ưu điểm chính của MDSD có thể kể đến như sau:

- Giảm chi phí phát triển: Giảm thời gian phát triển dẫn đến giảm chi phí phát triển phần mềm. Ưu điểm này có thể thấy được qua khả năng tạo ra các mã nguồn có chả năng chạy (runable code) từ các mô hình hình thức sử dụng một hoặc nhiều bước chuyển (M2M và M2T).
- Tăng chất lượng phần mềm: MDSD sử dụng các bước chuyển tự động và ngôn ngữ hình thức giúp tăng chất lượng phần mềm, đặc biệt là khi có thể sử dụng các kiến trúc, thành phần phần mềm có khả năng tái sử dụng đã được sử dụng và kiểm chứng trước đó một hoặc nhiều lần.
- Tính khả chuyển cao: Các kiến trúc, ngôn ngữ mô hình hóa và bộ chuyển đổi có thể được sử dụng để sản xuất các hệ thống phần mềm khác nhau. Điều này nâng cao khả năng tái sử dụng và rút ngắn một bước nữa trong việc giảm thiểu chi phí và thời gian phát triển phần mềm.
- Sự độc lập về nền tảng: MDSD mang đến khả năng vượt trội - sự độc lập về nền tảng. Nhà phát triển có thể xây dựng sản phẩm hướng mô hình mà không cần quan tâm đến nền tảng của sản phẩm. Sau đó, thực hiện chuyển đổi sang các nền tảng tương ứng nhờ các bộ chuyển đổi.

Nhược điểm

- Sự hoàn thiện của sản phẩm: Các sản phẩm của các quá trình chuyển đổi trong MDA thường chưa hoàn chỉnh. Đòi hỏi sự cải thiện, bổ sung của nhà phát triển trước khi đưa tới người dùng cuối.
- Sự hỗ trợ các tính năng gốc: Do các công cụ mô hình hóa không phải là công cụ gốc, được phát triển bởi bên thứ ba nên việc cập nhật tính năng nhằm hỗ trợ các công nghệ mới thường chậm trễ hơn công cụ sử dụng mã nguồn gốc.
- Giới hạn về công cụ**:** Các tính năng, loại phần mềm có thể xây dựng phụ thuộc hoàn toàn và nhà cung cấp công cụ phát triển, hay các bộ chuyển đổi.

Tồn tại không ít nhược điểm bên cạnh các ưu điểm nổi trội, MDD vẫn được xem như phương pháp phát triển phần mềm ưu việt cho xu thế phát triển phần mềm hiện tại. Tuy nhiên, tùy thuộc vào đặc thù của mỗi hệ thống, nhà phát triển cần xem xét và cân nhắc kỹ lưỡng trước khi đưa ra phương pháp tối ưu nhất.

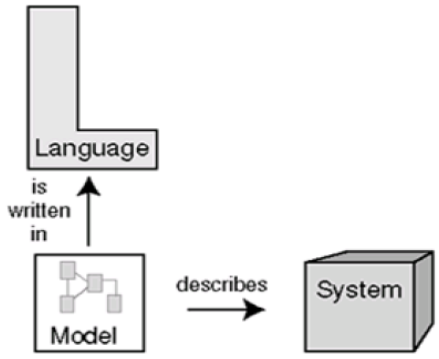
1.3.2. Các khái niệm chính

1.3.2.1. Mô hình

Mô hình là một biểu diễn trừu tượng của cấu trúc, tính năng và hành vi của hệ thống. Mô hình có thể được biểu diễn bằng các ký hiệu đồ họa và diễn tả bằng ngôn ngữ đặc tả miền cụ thể dưới dạng ngôn ngữ hình thức. AnneKe đã định nghĩa mô hình như sau [4]:

"Một mô hình là một mô tả (hoặc một phần) của một hệ thống được viết bởi một ngôn ngữ hình thức". "Ngôn ngữ hình thức là ngôn ngữ với mẫu được xác định rõ ràng và ngữ nghĩa phù hợp với việc biên dịch tự động bởi máy tính".

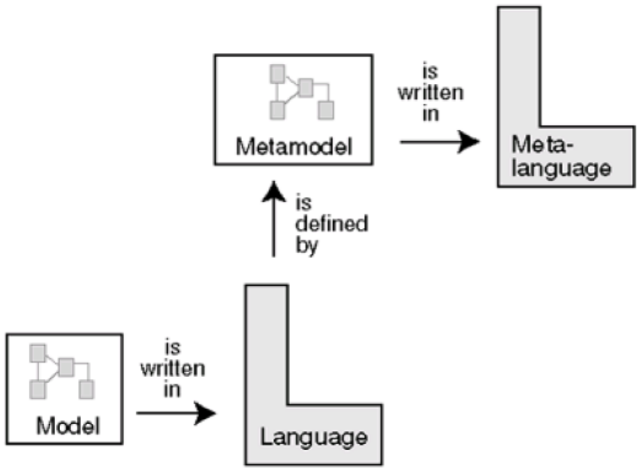
Mô tả về mô hình được biểu diễn cụ thể hơn trong Hình 1.3:



Hình 1.3: Mô hình được viết bởi ngôn ngữ hình thức nhằm biểu diễn hệ thống.

1.3.2.2. Meta-model

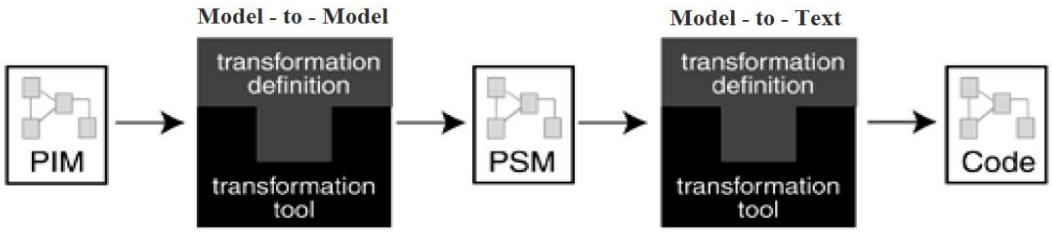
Meta-model là một mô hình ở mức trừu tượng hơn và sử dụng để biểu diễn mô hình. Metamodel được viết bởi ngôn ngữ gọi là Meta-language. Meta-model được biểu diễn trong Hình 1.4 [4]:



Hình 1.4: Meta-model định nghĩa model được viết bởi Meta-language.

1.3.2.3. Chuyển mô hình

Chuyển mô hình bao gồm hai phương thức cơ bản sau : Mô hình thành Mô hình (M2M) và Mô hình thành Văn bản (M2T). Các phương thức này dùng các công cụ chuyển đổi (transformation tool) để thực hiện chuyển mô hình, bộ chuyển đổi bao gồm một tập hợp các định nghĩa chuyển đổi (transformation definition). Mô hình được chuyển đổi tuân theo chặt chẽ các luật chuyển này. Luật chuyển là thành phần cốt lõi của các bộ chuyển đổi mô hình. Hình 1.5 [4] mô tả các bước chuyển đổi mô hình:



Hình 1.5: Các bước chuyển mô hình trong MDA.

1.3.2.4. Luật chuyển mô hình

Luật chuyển mô hình được định nghĩa như sau [4]: *"Một luật chuyển đổi mô hình là sự mô tả cách một hay nhiều cấu trúc trong ngôn ngữ nguồn có thể thay đổi thành một hoặc nhiều cấu trúc trong ngôn ngữ đích"*. Luật chuyển mô hình mô tả cách một phần của mô hình nguồn có thể được chuyển đổi thành một phần của mô hình đích. Các cấu trúc thể hiện các phần trong mô hình nguồn được ánh xạ tới các cấu trúc thể hiện các phần trong mô hình đích tương ứng.

1.3.3. Phát triển phần mềm hướng mô hình trong lập trình di động

Ngày nay, điện thoại di động thông minh (smartphone) là một thiết bị phổ biến gắn bó với người dùng phần lớn thời gian trong ngày. Người dùng sử dụng smartphone không chỉ đơn giản để gọi điện, nhắn tin hay các tính năng cơ bản mà còn sử dụng điện thoại để giải trí và sử dụng các ứng dụng khác. Từ thực trạng này, xu hướng phát triển ứng dụng nhanh nhằm đáp ứng nhu cầu của thị trường đang dần trở nên phổ biến. Các nhà phát triển luôn hướng tới mục tiêu giảm thiểu thời gian, chi phí phát triển mà vẫn đảm bảo chất lượng sản phẩm.

Phương pháp phát triển phần mềm truyền thống dường như bị "rút ngắn" trong quá trình ứng dụng trong lĩnh vực di động khi các yêu cầu thay đổi liên tục, thời gian phát triển sản phẩm phải đặc biệt ngắn. MDD được đề xuất như là phương pháp thích hợp nhất cho các mục tiêu và thực trạng hiện tại. Một số dự án áp dụng MDD trong phát triển ứng dụng trên nền tảng di động có thể kể đến như: The Simple Mobile Service [11] ứng dụng MDD để xây dựng dịch vụ cho điện thoại (Mobive service). PervML [13] với mục tiêu tạo ra các hệ thống thông qua việc áp dụng MDD. Ngôn ngữ mô hình hóa đa phương tiện (MML - Multimedia Modeling Language) [2] và một số nghiên cứu khác được Florence đề cập trong báo cáo khoa học của mình [9] cũng hướng tới việc sử dụng MDD trong lĩnh vực di động.

Tuy nhiên, chưa có kỹ thuật, công cụ nào được cộng đồng phát triển ứng dụng di động chào đón và sẵn sàng sử dụng như một phương pháp hoàn hảo thay thế kỹ thuật xây dựng ứng dụng gốc. Với mục tiêu giới thiệu một kỹ thuật mới ứng dụng phương pháp MDD trong lĩnh vực di động. Tác giả sẽ giới thiệu về lập trình ứng dụng di động trong phần tiếp theo như là kiến thức nền tảng nhằm làm rõ hơn nội dung, kết quả chính của luận văn này.

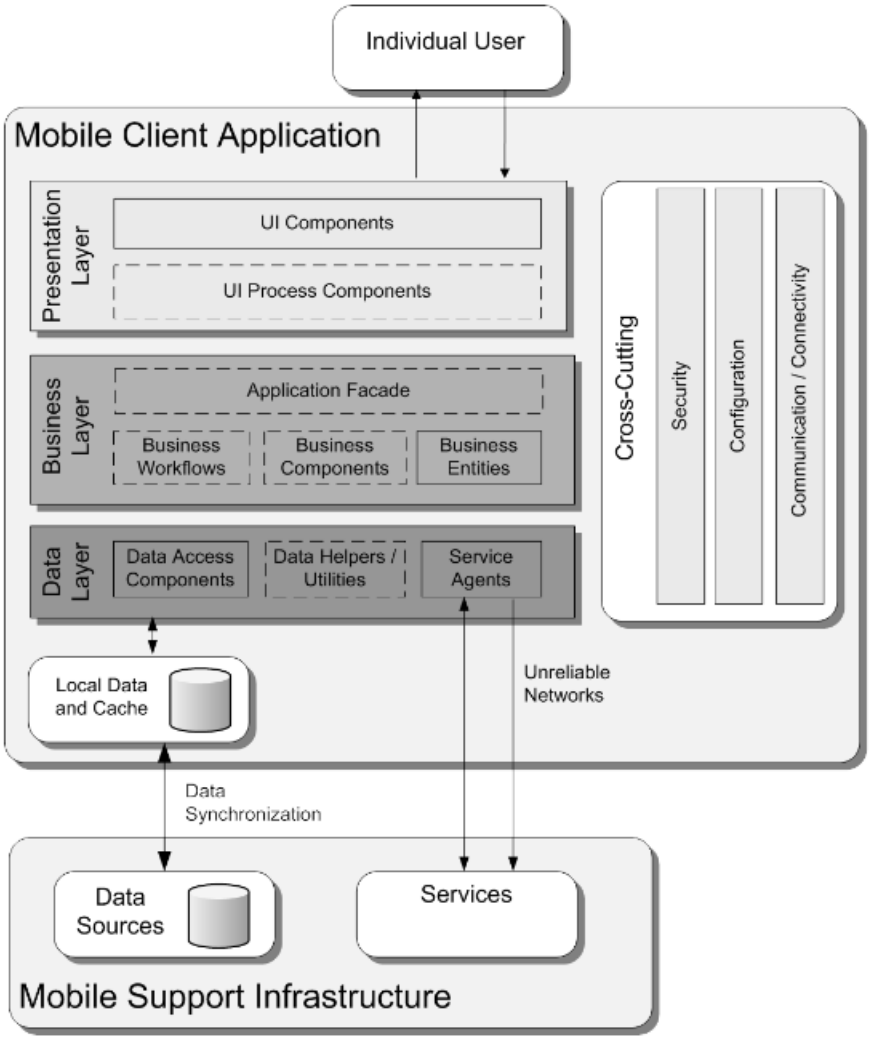
1.4. Lập trình ứng dụng di động

Lập trình ứng dụng di động là một trong các lĩnh vực phần mềm có tốc độ phát triển nhanh nhất hiện nay. Cộng đồng lập trình viên phát triển ứng dụng di động ngày càng trở nên đông đảo và lớn mạnh. Nhu cầu sử dụng phần mềm trên nền tảng di động tăng cao theo sự phát triển của công nghệ, thiết bị di động. Tuy nhiên, kéo theo đó là sự phức tạp trong thiết kế, xây dựng ứng dụng di động.

1.4.1. Nguyên tắc thiết kế ứng dụng di động

Thiết kế điện thoại di động là rất khác nhau, không chỉ về kích thước mà còn về các đặc điểm kỹ thuật như: Tỉ lệ điểm ảnh, thống số cấu hình, độ phân giải màn hình, hệ điều hành...Xu hướng thiết kế luôn làm người dùng cảm thấy thoải mái nhất, cùng với điều này, thiết kế ứng dụng di động cũng phải hướng đến mục tiêu làm thỏa mãn người dùng.

Các thiết bị di động nhỏ gọn và có tính di động hơn máy tính rất nhiều, do đó rất thuận tiện để sử dụng. Ngày nay, hầu hết các thiết bị di động sử dụng màn hình cảm ứng, giúp người dùng tương tác với thiết bị dựa trên các cử chỉ và yếu tố giao diện đơn giản. Kích thước nhỏ hơn yêu cầu cấu trúc nội dung nhỏ và đơn giản hơn để có thể hiển thị. Ngoài ra, giới hạn băng thông và kết nối yêu cầu tối ưng hóa thiết kế của điện thoại với yêu cầu dữ liệu ít. Người dùng có xu hướng sử dụng điện thoại di động thường xuyên hơn: trên xe buýt, khi đi bộ, trong công viên... Chúng thường được sử dụng trong thời gian chờ đợi, thậm chí là trong khi đang làm việc khác, có nghĩa là chúng được sử dụng trong điều kiện khó khăn với một loạt vấn đề phiền phức. Điều này dẫn đến nhà phát triển phải tuân thủ những quy tắc ngặt nghèo nhằm làm thỏa mãn người dùng. Hình 1.6 mô tả cấu trúc chung của một ứng dụng di động [14]:



Hình 1.6: Cấu trúc chung của một ứng dụng di động.

Một ứng dụng di động nói chung bao gồm các thành phần giao diện người dùng trên lớp trình diễn (Presentation Layer). Lớp nghiệp vụ (Business Layer) thường chứa các thành phần logic nghiệp vụ, tất cả các thành phần luồng xử lý nghiệp vụ (Business workflows) và thực thể nghiệp vụ (Business Entities) được yêu cầu bởi ứng dụng. Các lớp dữ liệu (Data Layer) bao gồm các dữ liệu truy cập và các thành phần dịch vụ (Service Agents).

Nhằm mục đích xây dựng một ứng dụng phần mềm tốt nhất, các yếu tố dưới đây cần được xem xét một cách cẩn thận trong quá trình thiết kế:

- Quyết định loại ứng dụng di động sẽ xây dựng, nhà phát triển cần xem xét loại ứng dụng di động sẽ phát triển trước khi bắt tay vào xây dựng ứng dụng di động. Nếu ứng dụng yêu cầu các quá trình xử lý trên thiết bị và ít kết nối

tới hệ thống hay dịch vụ, và yêu cầu trải nghiệm người dùng tốt với hiệu suất cao, nhà phát triển nên phát triển theo hướng ứng dụng gốc. Nếu ứng dụng là đơn giản, thường xuyên sử dụng kết nối tới hệ thống qua internet, không yêu cầu khắt khe về trải nghiệm người dùng, nhà phát triển nên xây dựng dưới dạng ứng dụng lai hay ứng dụng Web nhằm khai thác triệt để những lợi ích của phương pháp này.

- Quyết định loại thiết bị sẽ hỗ trợ. Khi chọn loại thiết bị để hỗ trợ, nhà phát triển cần cân nhắc những yếu tố then chốt như độ phân giải màn hình, kích cỡ màn hình, đặc điểm hiệu suất bộ vi xử lý, bộ nhớ và dung lượng bộ nhớ, công cụ và môi trường phát triển. Có thể yêu cầu GPS hay các loại cảm biến đối với từng loại ứng dụng.
- Xem xét loại ứng dụng có yêu cầu kết nối tới băng thông internet giới hạn hay không. Điều này giúp quản lý tốt các nghiệp vụ cần kết nối mạng hay chế độ hoạt động mà không cần kết nối.
- Thiết kế những giao diện người dùng thích hợp cho loại thiết bị di động sẽ hỗ trợ.
- Thiết kế kiến trúc phù hợp cho thiết bị di động sao cho tăng khả năng tái sử dụng và khả năng bảo trì.
- Thiết kế ứng dụng xem xét tới tài nguyên ràng buộc của thiết bị như lượng pin, kích cỡ bộ nhớ và tốc độ xử lý. Mỗi nhà phát triển luôn phải quyết định xem ứng dụng của mình có phù hợp với các tài nguyên của thiết bị hay không. Hệ điều hành di động thông minh ngày nay luôn xuất hiện các cơ chế cho phép thiết bị dừng các ứng dụng gây lãng phí tài nguyên có thể đe dọa đến các tính năng sống còn như gọi điện, truy cập internet, quay phim...

Với việc cân nhắc kỹ các yếu tố và đưa ra giải pháp phù hợp nhất, nhà phát triển có thể cân bằng các điều kiện quan trọng như chi phí/thời gian phát triển ứng dụng và chất lượng sản phẩm, tránh được các rủi ro nghiêm trọng phát sinh trong quá trình phát triển sản phẩm.

1.4.2. Lập trình ứng dụng di động trên Android

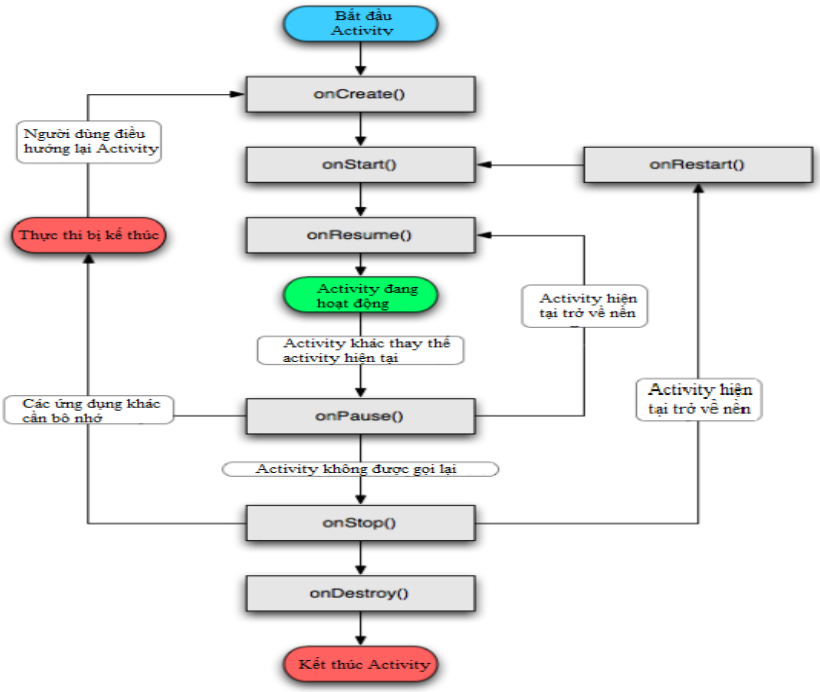
Android là hệ điều hành mã nguồn mở dựa trên dựa trên nền tảng Linux được thiết kế dành cho các thiết bị di động có màn hình cảm ứng như điện thoại thông minh và máy tính bảng. Android được Google ra mắt vào năm 2007 với giấy phép Apache, sự thuận lợi về mã nguồn mở và một giấy phép ít ràng buộc đã cho phép các nhà phát triển thiết bị, mạng di động và các lập trình viên được điều chỉnh

và sử dụng Android một cách tự do. Ngoài ra, cộng đồng phát triển Android đang là cộng động phát triển ứng dụng di động đông đảo nhất hiện nay.Tính đến quý 2 năm 2016, số lượng thiết bị chạy Android chiếm 87.6% lượng thiết bị di động thông minh toàn cầu[27].

Ứng dụng Android được xây dựng với ngôn ngữ Java bằng các công cụ phổ biến như Eclipse ADT hay Android Studio đi cùng với bộ công cụ phát triển ứng dụng phần mềm Android - Android Software Development Kit (Android SDK).Các ứng dụng Android sau khi hoàn thành sẽ được đóng gói dưới dạng .apk file, có thể được cài trực tiếp trên thiết bị hoặc tải về từ kho ứng dụng (Google Play, Amazon Store). Các thành phần chính cần lưu ý khi phát triển ứng dụng Android bao gồm:

Hoạt động (Activities)

Là một thành phần thuộc tầng trình diễn của ứng dụng, giao diện người dùng của ứng dụng được xây dựng dựa trên một hoặc nhiều Activity. Activity là một trong những thành phần cơ bản của Android chịu trách nhiệm tương tác với người dùng bằng những cửa sổ hiển thị tương ứng. Hình 1.1 mô tả vòng đời của một Activity trong Android.



Hình 1.7: Vòng đời của một Activity

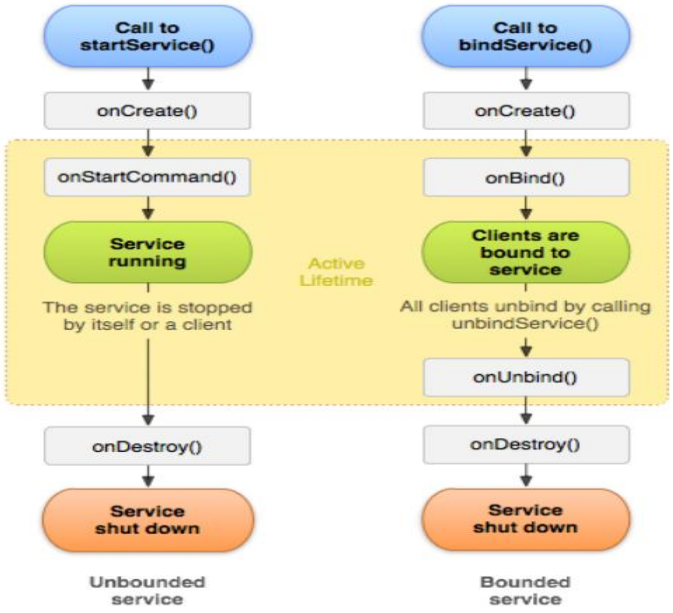
Trong đó các hàm sự kiện như onCreate(), onStart(), onResume(), onPause(), onStop(), onDestroy(), onRestart() đều được khai báo trong mã nguồn lớp của Activity tương ứng.

Dịch vụ (Service)

Là các chương trình chạy ẩn để thực hiện các thao tác mà không cần tương tác với người dùng. Ví dụ Service có thể mở một bản nhạc trong khi người dùng đang sử dụng ứng dụng khác, hay trả về các vị trí thông qua GPS hoặc truy cập dữ liệu qua mạng mà không ảnh hưởng đến người dùng. Service có hai loại:

- Không ràng buộc: Một Service được bắt đầu giống thành phần khác như Activities, bắt đầu Service bằng cách gọi hàm startService(). Thông thường, một Service đang chạy thực hiện một hành động đơn lẻ và không trả về kết quả cho đối tượng gọi. Ví dụ, nó có thể tải xuống hoặc tải lên một file thông qua kết nối mạng và tự động dừng lại khi hoàn thành.
- Ràng buộc: Một Service được ràng buộc khi các thành phần ràng buộc thông qua gọi hàm bindService(). Service ràng buộc thường là kiểu giao diện khách – chủ (client-server), nó cho phép các thành phần tương tác với Service, gửi yêu cầu và nhận kết quả trả về. Một Service ràng buộc có thể chạy với nhiều thành phần ràng buộc đến khi tất các các thành phần không còn ràng buộc nữa thì Service sẽ bị hệ thống hủy.

Hình 1.8 mô tả vòng đời của service ràng buộc và service không ràng buộc[25].



Hình 1.8: Vòng đời của Service

Trong đó hàm onStartCommand() được hệ thống gọi khi có một thành phần hoặc một Activity yêu cầu bắt đầu Service bằng cách gọi hàm startService().

Hàm onBind() được hệ thống gọi khi một thành phần muốn ràng buộc với Service bằng cách gọi hàm bindService().

Hàm onCreate() được hệ thống gọi khi lần đầu tiên Service chạy trước khi chạy hàm onStartCommand và onBind. Nếu Service đã hoạt động thì hàm này sẽ không được gọi nữa.

Hàm onDestroy() được hệ thống gọi khi Service không được sử dụng nữa và cần gọi hàm này để giải phóng toàn bộ tài nguyên cần thiết liên quan đến Service.

Ý định (Intent)

Là một hạt nhân rất quan trọng của Android, Intent thường được dùng để trao đổi dữ liệu giữa hai Activity hay giữa Activity với ứng dụng khác. Intent cũng có thể được hiểu như là một cấu trúc dữ liệu mô tả cách thức, đối tượng thực hiện của một Activity.

Intent có thể gọi startActivity() để hiển thị một Activity, broadcastIntent() để gửi nó tới bất kỳ thành phần Broadcast Receiver liên quan nào hay gọi startService(), bindService() để giao tiếp với các service chạy nền. Nhiệm vụ quan

trọng nhất của nó là việc gọi tới các Activity, nó như một kết nối giữa các Activity với nhau.

Bộ nhận tín hiệu (Broadcast Receivers)

Là một trong các thành phần chính của Android, có thể được hiểu như một bộ thu các bản tin cần thiết cho ứng dụng. Các bản tin được thu ở đây chính là các Intent, có thể thu các Intent sẵn có của hệ điều hành như: tin nhắn đến, cuộc gọi đến, trạng thái của điện thoại hoặc các Intent do chính ứng dụng tạo ra để làm các nhiệm vụ chuyên biệt.

Bộ cung cấp nội dung (Content Provider)

Content Provider quản lý truy cập đến tập cấu trúc dữ liệu, chúng đóng gói dữ liệu và cung cấp cơ chế để bảo mật dữ liệu. Content Provider có thể cung cấp dữ liệu từ một ứng dụng này đến ứng dung khác dựa trên các yêu cầu. Một Content Provider có thể sử dụng các cách lưu trữ dữ liệu khác nhau, các dữ liệu này có thể được lưu trức trong cơ sở dữ liệu, file hay thông qua mạng kết nối.

Mỗi ứng dụng Android chạy trong các tiến trình riêng của chính mình và có các điều khoản của riêng nó, điều này cho phép giữ dữ liệu của chúng ẩn với các ứng dụng khác. Tuy nhiên, đôi khi dữ liệu được yêu cầu chia sẻ đến các ứng dụng khác, Content Provider cũng được sử dụng cho mục đích này.

1.4.3. Lập trình ứng dụng di động đa nền tảng

Cách phổ biến nhất để xây dựng các ứng dụng di động là sử dụng các công cụ gốc (native) đi cùng với nền tảng đó. Đối với Android thì nó là Java và Eclipse ADT hoặc Android Studio đi cùng với Android SDK (Software Development Kit). Đối với iOS thì nó là Objective-C hoặc Swift và Xcode. Không có khả năng sử dụng phần mã nguồn Android để tái phái triển ứng dụng trên nền tảng khác như iOS. Đồng nghĩa với điều này, nhà phát triển này cần phải có năng lực gần như gấp đôi so với ban đầu để có thể phát triển ứng dụng trên một nền tảng khác, chi phí cũng tăng theo yêu cầu về nhân lực và thiết bị.

Đứng trước tình hình đó, lập trình di động đa nền tảng xuất hiện như một giải pháp cho vấn đề trên. Các ứng dụng có thể được xây dựng một lần cho tất cả các hệ điều hành di động giúp tiết kiệm đáng kể chi phí và thời gian phát triển. Chỉ với ưu điểm vượt trội này, lập trình đa nền tảng đã xứng đáng được xem xét một cách nghiêm túc như một hướng phát triển phần mềm chính thức bên cạnh phương pháp

truyền thống sử dụng công cụ gốc. Ưu điểm và nhược điểm chính của lập trình đa nền tảng được thể hiện như sau:

Ưu điểm

Ưu điểm lớn nhất của lập trình đa nền tảng là tiết kiệm chi phí và thời gian phát triển ứng dụng. Ưu điểm này càng tăng lên với mỗi nền tảng mà nhà phát triển sẽ phát hành sản phẩm. Anmol và cộng sự đưa ra so sánh chi tiết về chi phí phát triển ứng dụng di động theo hai phương pháp : Đa nền tảng (ứng dụng lai - Hybrid app) và ứng dụng gốc (Native app) được thể hiện theo Bảng 1.1 [3]:

Bảng 1.1: So sánh chi phí xây dựng ứng dụng di động gốc và ứng dụng lai.

Description	Small to Medium Size Projects Average Cost Native App Development	Small to Medium Size Projects Average Cost Hybrid App Development
Back-end programming, APIs, admin and cloud deployment	$10,000 - $20,000	$10,000 - $20,000
iOS native development	$15,000 - $30,000	$0
Android native development	$10,000 - $20,000	$0
Hybrid app development using PhoneGap technology	$0	$10,000 - $20,000
Total	$35,000 - $70,000	$20,000 - $40,000

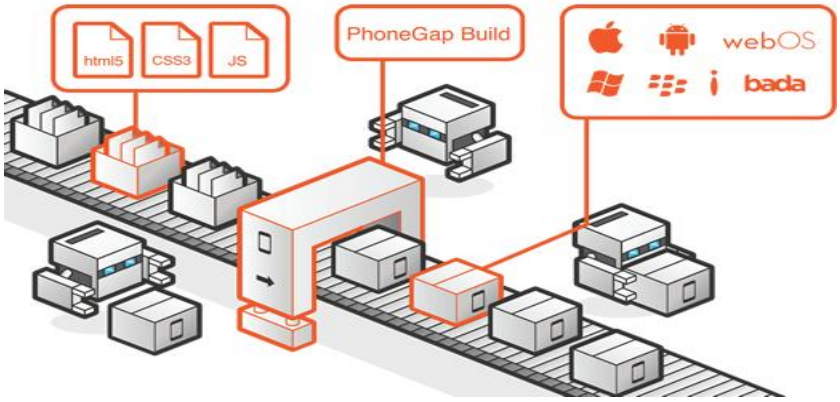
Theo báo cáo khoa học của Anmol, chi phí khi xây dựng ứng dụng lai đa nền tảng chỉ xấp xỉ 60% chi phí phát triển ứng dụng gốc. Một con số không hề nhỏ cho số lượng dự án phát triển ứng dụng di động hiện nay.

Nhược điểm

Lập trình đa nền tảng sử dụng các ngôn ngữ thứ hai (không phải ngôn ngữ gốc) và các bộ chuyển để có thể hoạt động trên thiết bị. Các bước chuyển gián tiếp này cũng như các hạn chế về khả năng hỗ trợ tính năng gốc làm ảnh hưởng xấu đến hiệu suất cũng như trải nghiệm người dùng.

Tiếp theo, luận văn sẽ đề cập đến một số công cụ hỗ trợ lập trình đa nền tảng nổi bật nhất hiện nay.

1.4.3.1. PhoneGap



Hình 1.9: PhoneGap Build

Mô hình PhoneGap được thể hiện trong Hình 1.9 [24]. PhoneGap về cơ bản là một tập các hàm JavaScript API, nó cho phép truy cập những khả năng gốc trong thiết bị di động. Nó cũng là một bộ đóng gói và cho phép xây dựng một ứng dụng Web mà sẽ được cài đặt cục bộ trên thiết bị đó.

Xây dựng một ứng dụng sử dụng PhoneGap đồng nghĩa với việc xây dựng một trang Web di động sử dụng HTML5 và JavaScript, giống như việc xây dựng những trang Web khác hiện nay, nhưng đặt mã HTML5 và JavaScript đó trên thiết bị di động. Các ứng dụng PhoneGap chạy trên trình duyệt cục bộ trên điện thoại và có một số tính năng trợ giúp (hook) để triệu gọi vào các thư viện gốc thông qua các giao diện lập trình ứng dụng JavaScript API.

Điều đó có nghĩa là việc phát triển một ứng dụng PhoneGap tương đương việc phát triển một trang web di động đa nền tảng. Tất cả mã nguồn trong ứng dụng là HTML5 và JavaScript nên phần lớn sẽ có thể chia sẻ được, nhưng sẽ không thể viết một ứng dụng dạng gốc.

Do ứng dụng PhoneGap sẽ chạy trong một trình duyệt nên sẽ giống một ứng dụng Web hơn là một ứng dụng dạng gốc. Giao diện nhà phát triển đã thiết kế sẽ không sử dụng những điều khiển gốc và sẽ phụ thuộc vào giới hạn và tốc độ của trình duyệt Web. Điều này có nghĩa là có thể phải viết một số mã nguồn cho một nền tảng xác định nào đó để phù hợp với các loại trình duyệt khác nhau, nhưng hầu hết những đoạn mã nguồn đó đều có khả năng chia sẻ.

Một lợi ích lớn của PhoneGap là cho phép tải dự án vào bất cứ môi trường nào khác môi trường ban đầu đã tạo nó, và có thể xây dựng một cách tự động cho những nền tảng khác.

1.4.3.2. Xamarin

Các công cụ của Xamarin về cơ bản cho phép phát triển các ứng dụng Android hoặc iOS bằng ngôn ngữ C# và có thể chia sẻ rất nhiều phần mã nguồn giữa các ứng dụng với nhau.

Khi viết một ứng dụng sử dụng bộ công cụ Xamarin là việc sử dụng một lớp trừu tượng phía trên các SDK thực sự của iOS và Android. Điều này có nghĩa là sẽ thu được kết quả là một ứng dụng gốc hoàn toàn cùng với giao diện người dùng gốc trên mỗi nền tảng.

Nhưng điều này cũng có nghĩa là việc chia sẻ mã nguồn giữa các nền tảng này sẽ bị giới hạn. Điển hình là khi phát triển một ứng dụng sử dụng công cụ của Xamarin, nhà phát triển sẽ xây dựng một phần lõi của ứng dụng mà mã nguồn của nó có thể chia sẻ giữa hai nền tảng iOS và Android, thậm chí là cả phiên bản trên Windows Phone của ứng dụng cũng dựa trên thư viện lõi này. Với hướng tiếp cận này, nhà phát triển có khả năng sử dụng mã nguồn của ứng dụng cho các nền tảng khác.

1.4.3.3. Appcelerator Titanium

Appcelerator Titanium sử dụng một tùy chỉnh API phát triển di động đa nền tảng để xây dựng ứng dụng. Điều này khác với PhoneGap hoặc Xamarin, Xamarin sử dụng một bộ đóng gói (wapper) xoay quanh các SDK gốc thực thụ, với PhoneGap nhà phát triển có thể sử dụng bất cứ thứ gì họ muốn để xây dựng một ứng dụng Web HTML5.

Với Titanium, nhà phát triển viết tất cả mã nguồn của mình dựa trên SDK của nó bao gồm các thành phần giao diện UI. Điều này có nghĩa là khi viết một ứng dụng Titanium đồng nghĩa với việc đang viết một giao diện người dùng đa nền tảng.

Các ứng dụng Appcelerator Titanium được biên dịch xuống tới các ứng dụng gốc hoàn toàn và sử dụng những điều khiển gốc thực sự trên nền tảng đó.

Một ví dụ điển hình, trong Titanium nhà phát triển có thể khai báo một nút (button) và bố cục (layout) xác định của nó cũng như một số thuộc tính cho button đó. Khi biên dịch ứng dụng, button sẽ xuất hiện như một button gốc của Android khi chạy trên thiết bị Android và như một button gốc của iOS khi chạy trên iOS. Điều này cho phép ứng dụng sử dụng rất nhiều tính năng gốc trên thiết bị. Nhiều thành phần UI và mô hình tương tác là đa nền tảng, tuy nhiên không phải là toàn bộ.

Một trong những ưu điểm của Titanium là đưa ra các tính năng về máy chủ đám mây (cloud server). Titanium cho phép thể truy cập tới các dịch vụ đám mây (cloud services) một cách đầy đủ. Có thể sử dụng các dịch vụ đám mây này để quản lý và xác thực người dùng, lưu trữ dữ liệu.

1.5. Tổng kết chương

Chương 1 đã đề cập tổng quan đế kỹ thuật phát triển hướng mô hình, các thành phần cũng, vai trò cũng như lợi ích của chúng trong phát triển phần mềm. Ngoài ra, nội dung chương cũng đề cập đến lĩnh vực lớn hiện nay: Lập trình di động. Các nguyên tắc thiết kế ứng dụng di động cũng như việc lập trình ứng dụng di động trên Android.

Một xu hướng lập trình di động khác mới nổi lên gần đây và thực sự được xem xét như một hướng phát triển khác cho việc xây dựng ứng dụng di động bên cạnh phương pháp xây dựng ứng dụng gốc truyền thống: Lập trình di động đa nền tảng. Nội dung chương cũng trình bày lợi thế, ưu điểm và nhược điểm cũng như một số công cụ lập trình di động đa nền tảng phổ biến hiện nay.

Phát triển phần mềm hướng mô hình là phương pháp phổ biến và tính ứng dụng cao trong phát triển phần mềm nói chung, tuy nhiên lại chưa được sử dụng triệt để và hiệu quả trong phát triển ứng dụng di động. Phương pháp truyền thống xây dựng ứng dụng gốc và các ứng dụng đa nền tảng sử dụng mã nguồn là sự lựa chọn tối ưu khi lập trình ứng dụng di động. Vậy có kỹ thuật nào giúp các nhà phát triển tận dụng tối đa ưu điểm của MDD? Chúng ta hay cùng tìm hiểu câu trả lời cho vấn đề này ở các chương tiếp theo.

CHƯƠNG 2 : KHẢO SÁT KỸ THUẬT MÔ HÌNH HÓA LUỒNG TƯƠNG TÁC

Chương 2 tập trung tìm hiểu kỹ thuật mô hình hóa luồng tương tác nói chung và kỹ thuật IFML nói riêng. Nội dung chương cũng đề cập tới tính ứng dụng của IFML trong phát triển phần mềm trên nền tảng di động qua công cụ WebRatio Mobile Platform và đề xuất các tiêu chí đánh giá kỹ thuật mô hình hóa luồng tương tác trong phát triển ứng dụng di động.

2.1. Giới thiệu

Tương tác bao gồm tương tác giữa các tác nhân với hệ thống và tương tác của hệ thống,là thành phần quan trọng và cần được hiểu rõ trong quá trình đặc tả yêu cầu cũng như phát triển hệ thống. Mô hình luồng tương tác giúp nhà phát triển hiểu rõ cách thức hệ thống hoạt động và tương tác với người dùng, là tài liệu giúp xây dựng "bộ khung" hoàn chỉnh của hệ thống trong giai đoạn phân tích, thiết kế.

IFML là kỹ thuật mô hình hóa luồng tương tác mới giúp nhà phát triển thể hiện rõ không chỉ những tương tác thông thường của hệ thống mà còn thể hiện cả những tương tác người dùng đang ngày một phức tạp theo sự phát triển của công nghệ.

Ngoài ra, việc ứng dụng IFML cũng được phát triển một cách đáng kể khi WebRatio giới thiệu công cụ WebRatio Mobile Platform cho phép ứng dụng IFML trong phát triển phần mềm trên nền tảng di động. Giúp nhà phát triển có thể phát triển ứng dụng di động trực tiếp từ mô hình IFML mà không cần thao tác với mã nguồn.

Bên cạnh lợi ích của IFML trong phát triển ứng dụng di động, kỹ thuật này cũng tồn tại nhiều hạn chế. Tác giả đề xuất một số tiêu chí và thực hiện đánh giá kỹ thuật IFML trong phát triển ứng dụng di động dựa vào việc tham khảo các nghiên cứu, bài báo khoa học trong các phần tiếp theo.

2.2. Hướng tiếp cận mô hình hóa luồng tương tác

Mô hình hóa luồng tương tác không phải là lĩnh vực mới, UML cho phép nhà phát triển xây dựng mô hình luồng tương tác với Biểu đồ trình tự, Biểu đồ cộng tác, Biểu đồ tương tác, Biểu đồ thời gian hay sự kết hợp của chúng. Tuy nhiên việc thực hiện mô hình tương tác với các biểu đồ UML chưa thực sự thể hiện tốt hệ thống dưới góc nhìn tổng quát, nhằm biểu diễn nội dung thể hiện của hệ thống qua các giao diện, các tương tác của người dùng và các hành vi điều khiển của hệ thống

được kích hoạt bởi tương tác người dùng. Đặc biệt là khả năng phát triển phần mềm hướng mô hình trực tiếp từ các PIM được xây dựng bởi ngôn ngữ mô hình hóa.

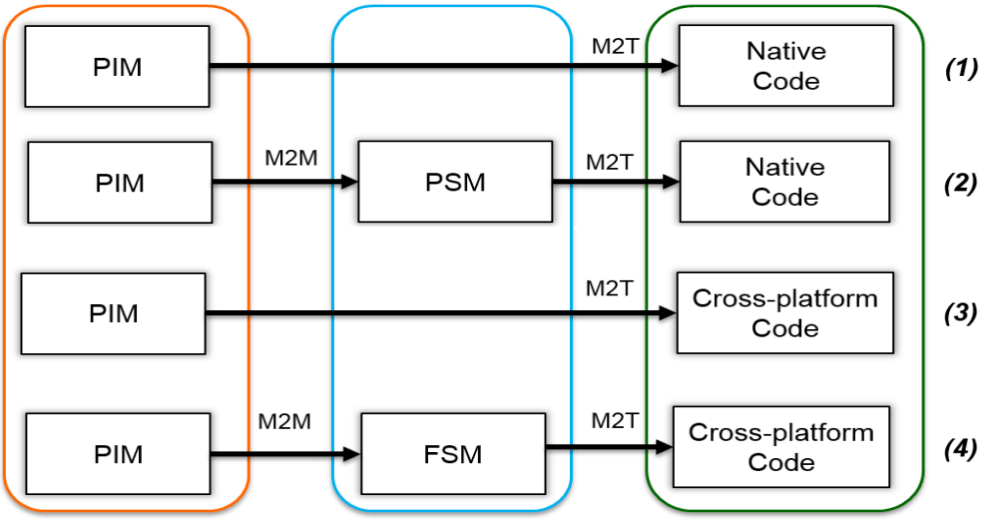
Với tình trạng thực tại, phương pháp phát triển phần mềm truyền thống còn tồn tại các hạn chế cơ bản như : khả năng tái sử dụng thấp, tỉ lệ rủi ro cao do lỗi, chi phí phát triển lớn cho đa nền tảng. Với mục tiêu khắc phục những nhược điểm trên, nội dung luận văn đề cập đến hướng tiếp cận mới cho kỹ thuật mô hình hóa luồng tương tác nhằm thể hiện mạnh mẽ các nội dung, tương tác người dùng và các hành vi điều khiển của một hệ thống đầu cuối và việc áp dụng kỹ thuật này trong phát triển phần mềm hướng mô hình.

Mô hình hóa luồng tương tác (Interaction Flow Modeling – IFM) là cách tiếp cận sử dụng phương pháp phát triển phần mềm hướng mô hình trong phát triển phần mềm hướng đến các mục tiêu:

- Mô hình hóa một lần và sinh mã (ứng dụng) cho các nền tảng được lựa chọn.
- Cải thiện quy trình phát triển phần mềm.
- Cho phép thể hiện giao tiếp giữa giao diện và sự tương tác hay cách thức hệ thống hoạt động tới các bên liên quan không nắm rõ về kỹ thuật như khách hàng.
- Cho phép đánh giá và kiểm thử yêu cầu từ các pha phát triển sớm hơn trong quá trình phát triển phần mềm.

IFM nhằm thể hiện trực quan nội dung của các giao diện người dùng, các sự kiện được kích hoạt bởi các tương tác của người dùng và các hành vi điều khiển của hệ thống phần mềm.

Việc áp dụng IFM vào quy trình phát triển phần mềm sử dụng kỹ thuật sinh mã tự động nhằm tạo ra ứng dụng trực tiếp từ mô hình cũng được các nhà phát triển quan tâm. Ngôn ngữ mô hình hóa luồng tương tác là một ngôn ngữ cấp độ ngôn ngữ độc lập nền (PIM) trong kiến trúc hướng mô hình [22]. Eric Umuhoza [8] đưa ra bốn cách tiếp cận cho quá trình sinh mã tự động trong phát triển phần mềm từ PIM, các cách tiếp cận này phù hợp để có thể được ứng dụng cho kỹ thuật IFM như biểu diễn trong Hình 2.1.



Hình 2.1: Các hướng tiếp cận phát triển ứng dụng hướng mô hình với kỹ thuật mô hình hóa luồng tương tác.

1. PIM - Mã nguồn gốc (Native Code): Sinh mã nguồn gốc qua bộ chuyển M2T tương ứng với từng nền tảng từ mô hình độc lập nền của ứng dụng. Cách tiếp cận này cho phép sinh ra mã nguồn gốc từ PIM, tuy nhiên chưa hoàn hảo để có thể sinh ứng dụng trực tiếp.
2. PIM - PSM - Mã nguồn gốc: Tương đồng với cách tiếp cận 1 nhưng sử dụng M2M để sinh PSM như một bước trung gian. Cách tiếp cận này có thể sinh mã nguồn gốc hiệu quả hơn nhưng đầu ra là mã nguồn chưa hoàn chỉnh và cần được chỉnh sửa thêm bởi nhà phát triển.
3. PIM - Mã nguồn đa nền tảng: Các mã nguồn đa nền tảng được sinh ra tuân theo các cấu trúc của framework đa nền tảng cụ thể. Sau đó, các framework này sẽ chịu trách nhiệm sinh ra các ứng dụng đa nền tảng. Để làm được điều này, framework thường sử dụng các mã nguồn đa nền tảng được sinh để tạo ra các file chạy ứng dụng cho mỗi nền tảng thông qua một quá trình tự động.
4. PIM - Framework Specific Model(FSM) - Mã nguồn đa nền tảng: Tương tự như 3, tuy nhiên có thêm một bước trung gian sử dụng M2M để chuyển đổi từ PIM sang FSM. Việc chuyển đổi này có thể tạo ra ứng dụng hiệu quả hơn do mỗi FSM chỉ phục vụ cho một nền tảng chuyên biệt.

Với mục tiêu ứng dụng mô hình hóa luồng tương tác trong phát triển ứng dụng phần mềm, WebRatio giới thiệu kỹ thuật với ngôn ngữ mô hình hóa luồng tương tác (được gọi là IFML - Interaction Flow Modeling Language) cùng các

công cụ hỗ trợ cho phép nhà phát triển thực hiện mô hình hóa luồng tương tác, xây dựng ứng dụng trực tiếp từ mô hình mà không cần viết hay chỉnh sửa mã nguồn.

2.3. Tổng quan kỹ thuật mô hình hóa luồng tương tác IFML

Sự ra đời của IFML là một bước thay thế cho các kỹ thuật mô hình hóa không còn phù hợp, việc ứng dụng IFML không chỉ giới hạn trên nền tảng Web mà còn được xây dựng nhằm tạo ra các ứng dụng di động trên nhiều nền tảng khác nhau.

2.3.1. Giới thiệu

Tiền thân của IFML là một ngôn ngữ mô hình hóa gọi là Ngôn ngữ mô hình hóa Web (Web Modelling Language - WebML), được hình thành trong dự án nghiên cứu Cơ sở hạ tầng thông tin thông minh nền tảng Web(Web-based Intelligent Information Infrastructures - W3I3 1998 - 2000) [17] được hỗ trợ bởi Ủy ban Châu âu. Từ năm 1999, WebML đã được sử dụng cho sử dụng để phát triển các ứng dụng Web công nghiệp như là các thỏa thuận nghiên cứu với các công ty như Microsoft và Cisco Systems.

Vào năm 2001, đội ngũ các lập trình viên và kỹ sư thiết kế đã thành lập một công ty khởi nghiệp với mục tiêu phát triển, phân phối và giới thiệu WebRatio, một công cụ thích hợp dựa trên WebML. Kể từ đó, phát triển hướng mô hình cho ứng dụng Web với WebRatio đã được áp dụng cho hàng nghìn ứng dụng trên toàn thế giới bao gồm cả các dự án lớn trong ngành công nghiệp như các tiện ích (nước và năng lượng), tài chính, hậu cần, thương mại điện tử...

Bước cuối trong lịch sử phát triển là quá trình chuẩn hóa IFML tại OMG. Việc được chuẩn hóa và công nhận là một tiêu chuẩn của OMG. IFML được OMG giới thiệu lần đầu vào tháng 3 năm 2013 và được công bố bản Beta 2 như một bản chính thức vào tháng 2 năm 2014.

IFML hỗ trợ đặc tả kỹ thuật của các ứng dụng đầu cuối độc lập với công nghệ và nền tảng sẽ triển khai. Trọng tâm của IFML nhằm mô tả về cấu trúc, hành vi của ứng dụng cung như các tương tác của người dùng cuối, các mô tả về cấu trúc và hành vi của hệ thống được giới hạn trong những khía cạnh mà nó ảnh hưởng trực tiếp đến trải nghiệm người dùng. IFML đề cập đến những vấn đề như sau của việc mô hình hóa ứng dụng đầu cuối:

- Thành phần của khung nhìn (View): Những thành phần nào của giao diện có thể trực quan hóa, làm thể nào để tổ chức chúng, chúng sẽ được hiển thị đồng thời hay loại trừ lẫn nhau.
- Các nội dung của khung nhìn: Những thành phần nội dung nào được hiển thị từ ứng dụng cho người dùng, các tham số đầu vào nào người dùng sẽ cung cấp cho ứng dụng.
- Các sự kiện: Những sự kiện tương tác nào được hỗ trợ.
- Các hành động: Các thành phần nghiệm vụ nào được kích hoạt bởi các sự kiện.
- Các hiệu ứng của sự tương tác: Các sự kiện trên tác động đến trạng thái của giao diện như thế nào.
- Tham số ràng buộc: Những dữ liệu nào được truyền giữa các thành phần của giao diện và các hành động.

IFML thể hiện những khía cạnh trên sử dụng một ngôn ngữ mô hình hóa trực quan dựa trên các tiêu chuẩn Kiến trúc hướng mô hình MDA của OMG. Với những ứng dụng sử dụng mô hình Model-View-Controller (MVC) điển hình như các ứng dụng di động, IFML trọng tâm vào thành phần View. Hơn nữa, IFML mô tả cách cách View tham hiếu hay phụ thuộc vào thành phần Model và Control của ứng dụng.

Cú pháp trừu tượng của IFML được đặc tả bởi bốn thành phần sau [17]:

- IFML Metamodel
- IFML UML Profile
- IFML Visual Syntax
- IFML XMI

Các thành phần và vai trò của chúng sẽ được đề cập một cách khái quát trong các mục tiếp theo.

2.3.2. Cú pháp trừu tượng của IFML

2.3.2.1. IFML Metamodel

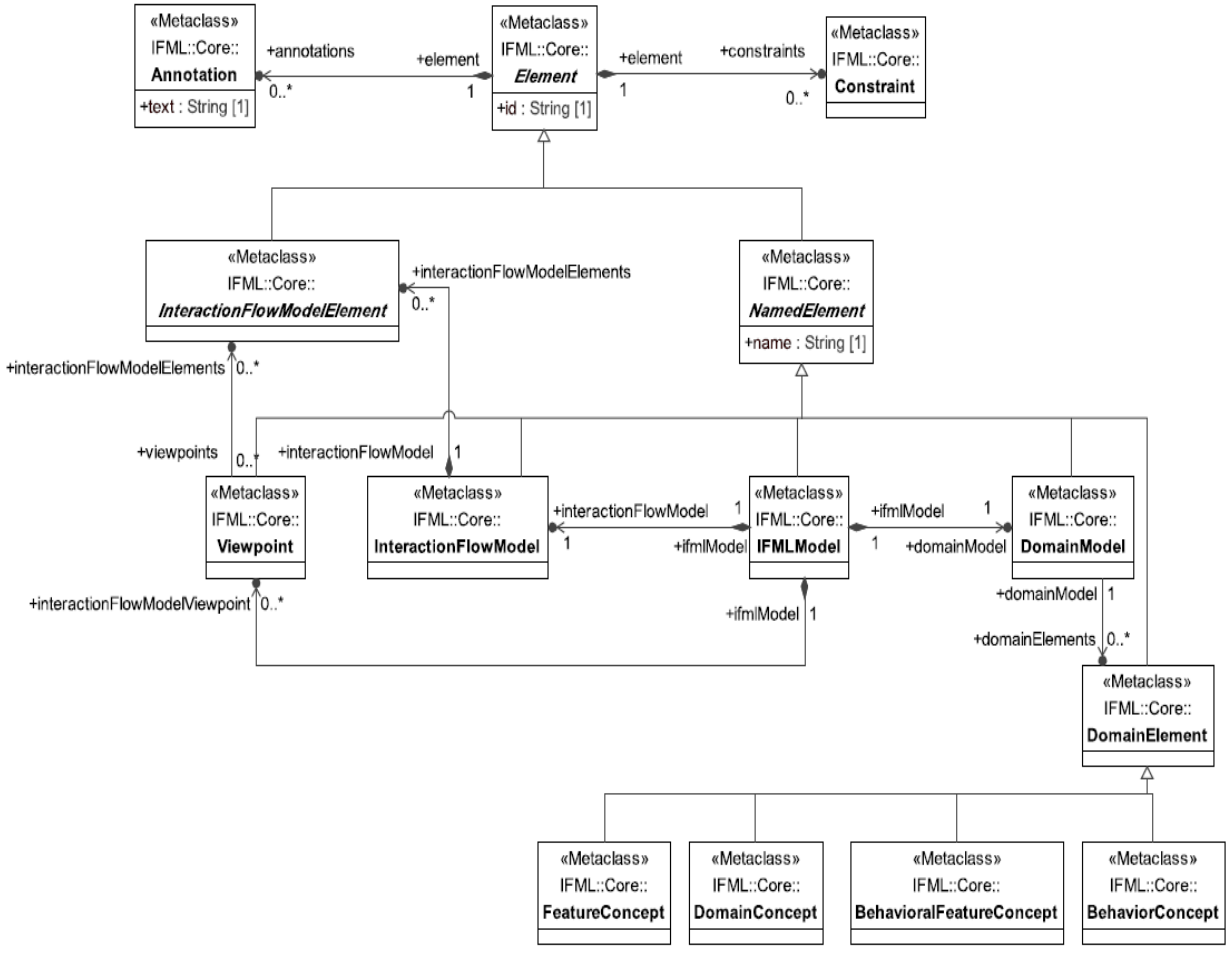
Metamodel của IFML được chia làm ba gói (packages) : Lõi (Core), Mở rộng (Extension) và Kiểu dữ liệu (DataType). Core chứa các khái niệm xây dựng các tương tác cơ bản của ngôn ngữ hình thức của InteractionFlowElements, InteractionFlows và Parameters. Core được mở rộng bởi các khái niệm cụ thể trong

gói Extension với các hành vi cụ thể và phức tạp hơn. DataType chưa các kiểu dữ liệu tùy biến được định nghĩa bởi IFML.

Các mô tả mức độ cao của IFML Metamodel được cấu trúc theo các tiêu chí chính như sau:

- Mô hình IFML (IFML Model)
- Mô hình luồng tương tác (Interaction Flow Model)
- Các phần tử luồng tương tác (Interaction Flow Elements)
- Các phần tử View (View Elements)
- Các sự kiện (Events)
- Tham số (Parameters)
- Các nội dung ràng buộc (Content Bindings)

IFML Model



Hình 2.2: Mô hình IFML (IFML Model)

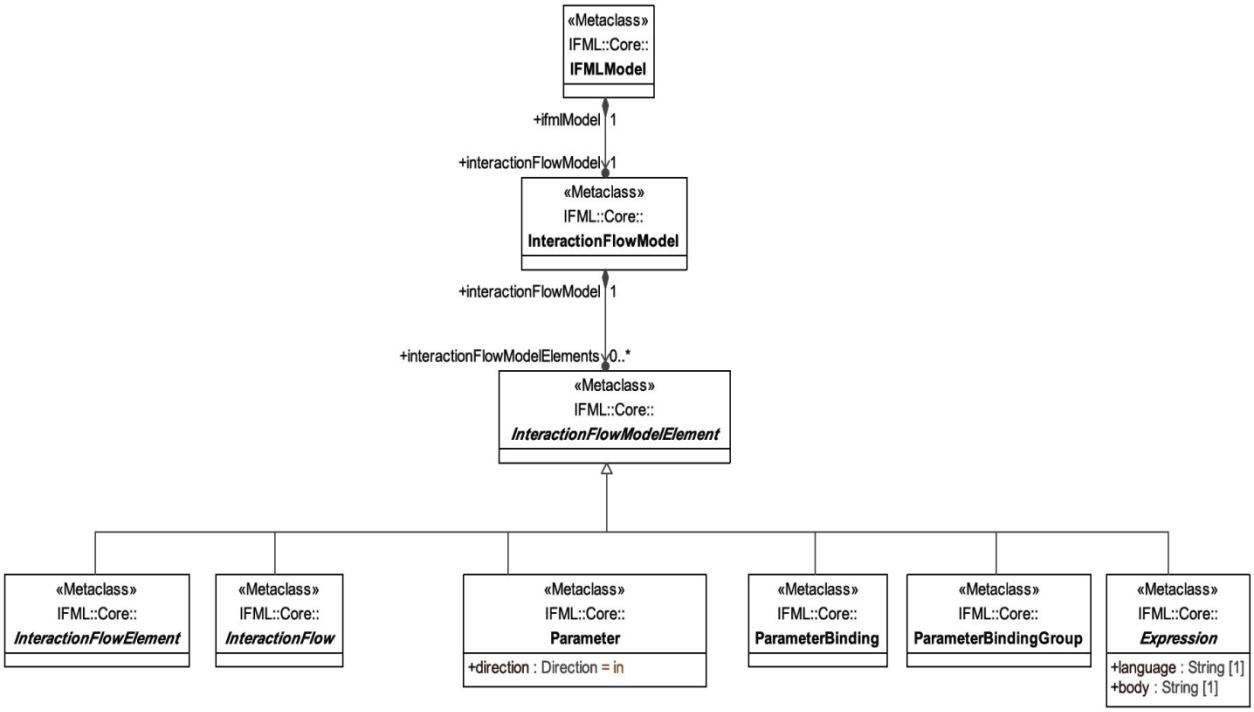
IFML Model thể hiện một mô hình IFML ở mức độ cao của tất cả các phần tử mô hình còn lại. Nó chứa một Mô hình luồng tương tác (InteractionFlowModel), một mô hình miền (DomainModel) và có thể có thêm điểm nhìn (ViewPoints).

Mô hình luồng tương tác đại diện khía cạnh người sử dụng của toàn bộ ứng dụng trong khi đó ViewPoints trình bày các khía cạnh cụ thể của hệ thống bằng các tham chiếu tới các phần tử luồng tương tác (InteractionFlowElements) như một định nghĩa đầy đủ toàn bộ tính năng của hệ thống.

Mô hình miền thể hiện mô hình nghiệp vụ của ứng dụng như các mô tả về nội dung và hành vi được xử lý (hay tham chiếu) trong InteractionFlowModel. DomainModel bao gồm các DomainElements như các khái niệm, tính năng, hành vi và phương pháp (DomainConcept, FeatureConcept, BehaviorConcept, và BehavioralFeatureConcept tương ứng).

NamedElement là một lớp trừu tượng thể hiện lớp phần tử (lớp chung nhất trong mô hình) biểu thị tên của phần tử đó.

Interaction Flow Model

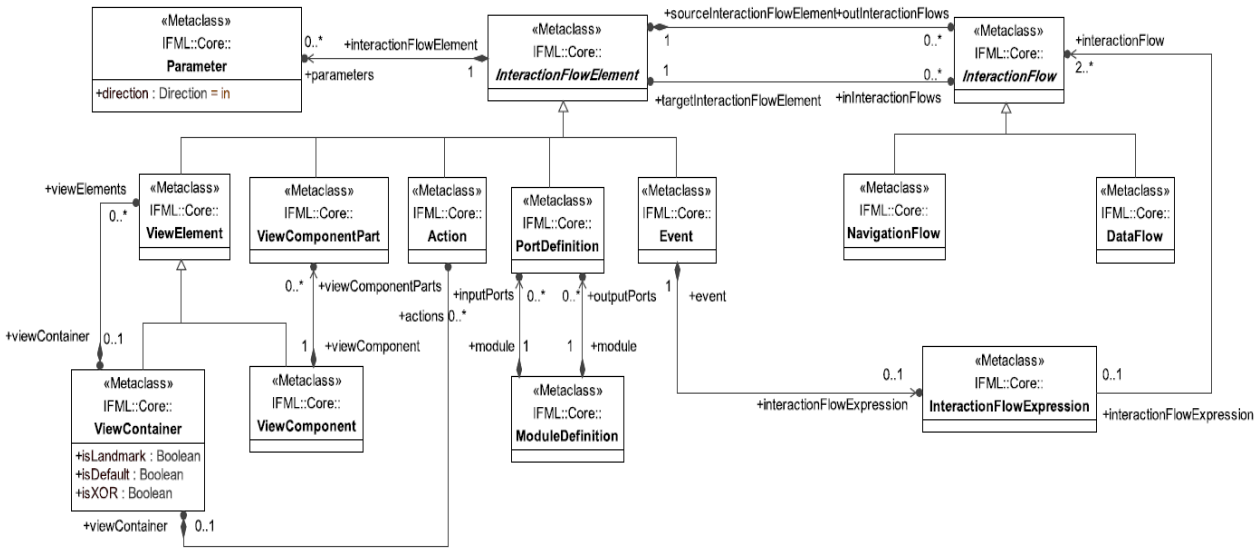


Hình 2.3: Mô hình luồng tương tác (Interaction Flow Model)

Một Mô hình luồng tương tác chứa tất cả các yếu tố của ứng dụng về khía cạnh người dùng được thể hiện bởi các Phần tử mô hình luồng tương tác

(InteractionFlowModelElement). Phần tử luồng tương tác (InteractionFlowElement) bao gồm: Phần tử luồng tương tác (InteractionFlowElement), Luồng tương tác (InteractionFlow), Tham số (Parameter), Tham số ràng buộc (ParameterBinding), Tập tham số ràng buộc (ParameterBindingGroup) và Thể hiện (Expression). Interaction Flow Elements là các phần tử xây dựng nên các tương tác, đại diện cho một phần của hệ thống tham gia vào các tương tác được kết nối bằng luồng tương tác.

Interaction Flow Elements



Hình 2.4: Các phần tử luồng tương tác (InteractionFlowElements).

Các phần tử luồng tương tác là một trong những khái niệm chính quan trọng của IFML đại diện cho một phần của hệ thống như các Phần tử khung nhìn (ViewElements), Thành phần khung nhìn (ViewComponentPart), Hành động (Action), PortDefinition và Sự kiện (Event). Các phần tử này tham gia kết nối các luồng tương tác.

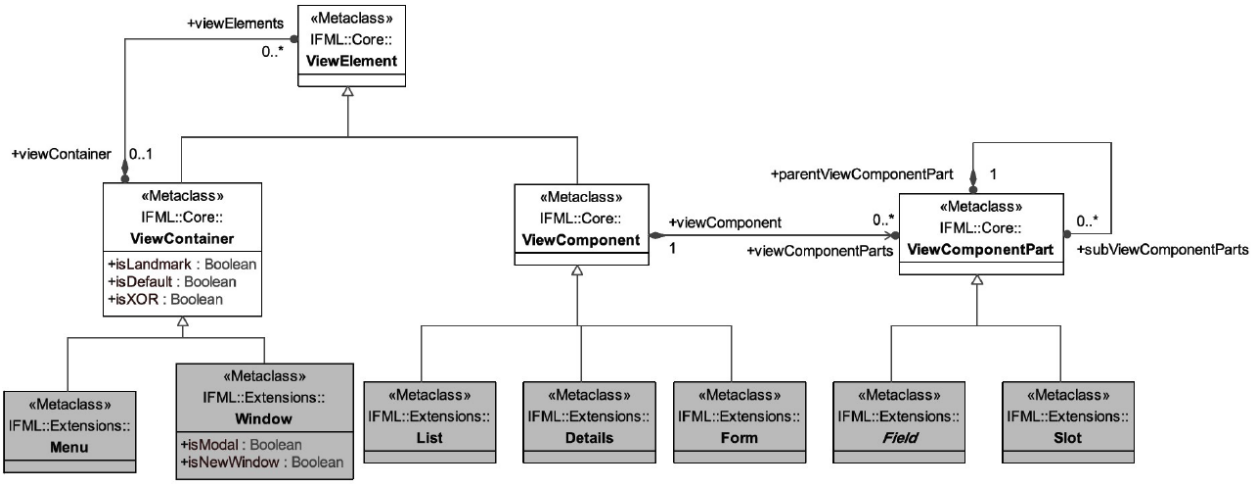
Tham số (Parameter) là thành phần được trao đổi trực tiếp giữa luồng tương tác của các phần tử luồng tương tác như các sự kiện, các hành động hay các luồng tương tác.

Luồng tương tác (InteractionFlow) bao gồm Luồng điều hướng (NavigationFlow) và luồng dữ liệu (DataFlow). Luồng điều hướng kết nối các sự kiện và các khung nhìn như một sự đáp ứng của tương tác người dùng. Luồng dữ

liệu giúp kết nối nội dung, dữ liệu hiển thị là kết quả xử lý của Sự kiện, Hành động giữa các khung nhìn khác nhau.

ViewContainer có thể chứa nhiều thành phần khung nhìn khác nhau hay các hành động.

View Elements



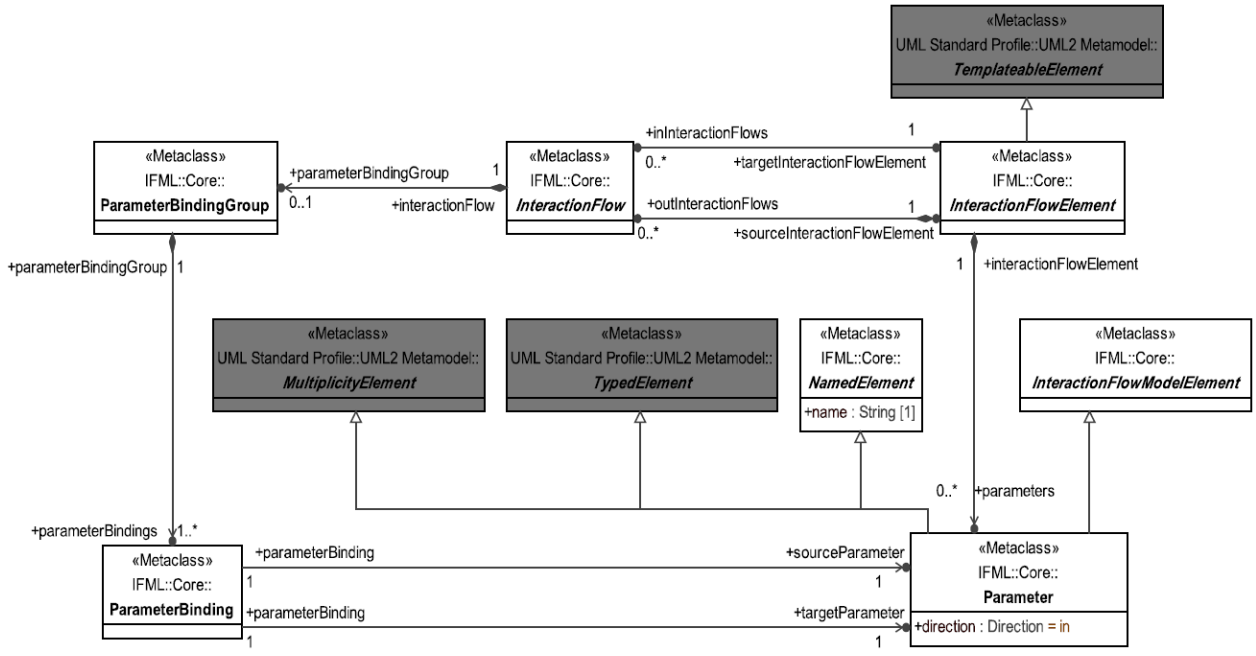
Hình 2.5: Các phần tử khung nhìn (ViewElement)

Các phần tử của một mô hình IFML có thể nhìn thấy ở mức độ giao diện được gọi là View Elements bao gồm View Container và View Component. View Container là các khung nhìn chứa các View Component khác. Các View Component giúp hiển thị nội dung, dữ liệu hay các tương tác đầu vào từ người sử dụng.

Một View Container có thể là cố định (landmark), là mặc định(default) hay loại trừ lẫn nhau(XOR). Một dạng của View Container là Menu, thể hiện tập hợp các phần tử có khả năng tương tác hoặc liên kết đến các View Container khác. Menu không thể chứa các View Container con hay View Component.

View Component có thể được cụ thể hóa như các danh sách(List), chi tiết(Details) và biểu mẫu(Form).

Parameters



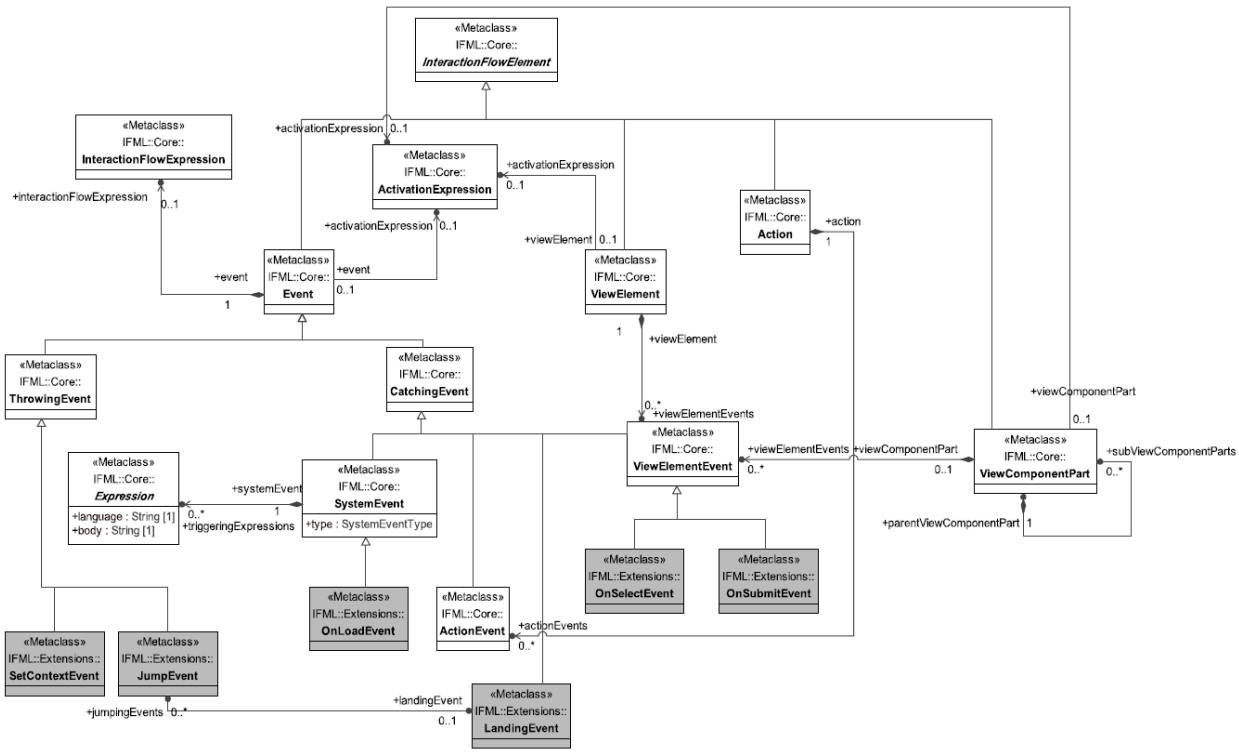
Hình 2.6: Các tham số (Parameters)

Parameter là các phần tử lưu giữ giá trị được sử dụng bởi các luồng tương tác giữa các phần tử luồng tương tác. Các tham số có thể được ánh xạ tới các yếu tố giao diện người dùng như ViewComponent. Các tham số ràng buộc giúp kết nối dữ liệu, nội dung hiển thị giữa các ViewComponent khác nhau như là kết quả của các sự kiện được kích hoạt bởi người dùng hoặc hệ thống.

Tham số có thể là đầu vào, đầu ra hoặc cả hai. Tham số mặc định được hiểu là tham số đầu vào, Một tham số đầu vào cho phép một phần tử luồng tương tác nhận một hoặc nhiều giá trị thông qua một luồng điều hướng hoặc một luồng dữ liệu. Một tham số đầu ra cho phép một phần tử luồng tương tác thể hiện dữ liệu còn một tham số bao gồm cả hai vai trò đầu vào/đầu ra sẽ bao gồm cả hai hành vi.

Một ràng buộc tham số luôn kèm với một luồng tương tác có thể được tập hợp thành tập ràng buộc tham số(ParameterBindingGroup), giúp ràng buộc nhiều thông tin dữ liệu theo các luồng tương tác.

Events



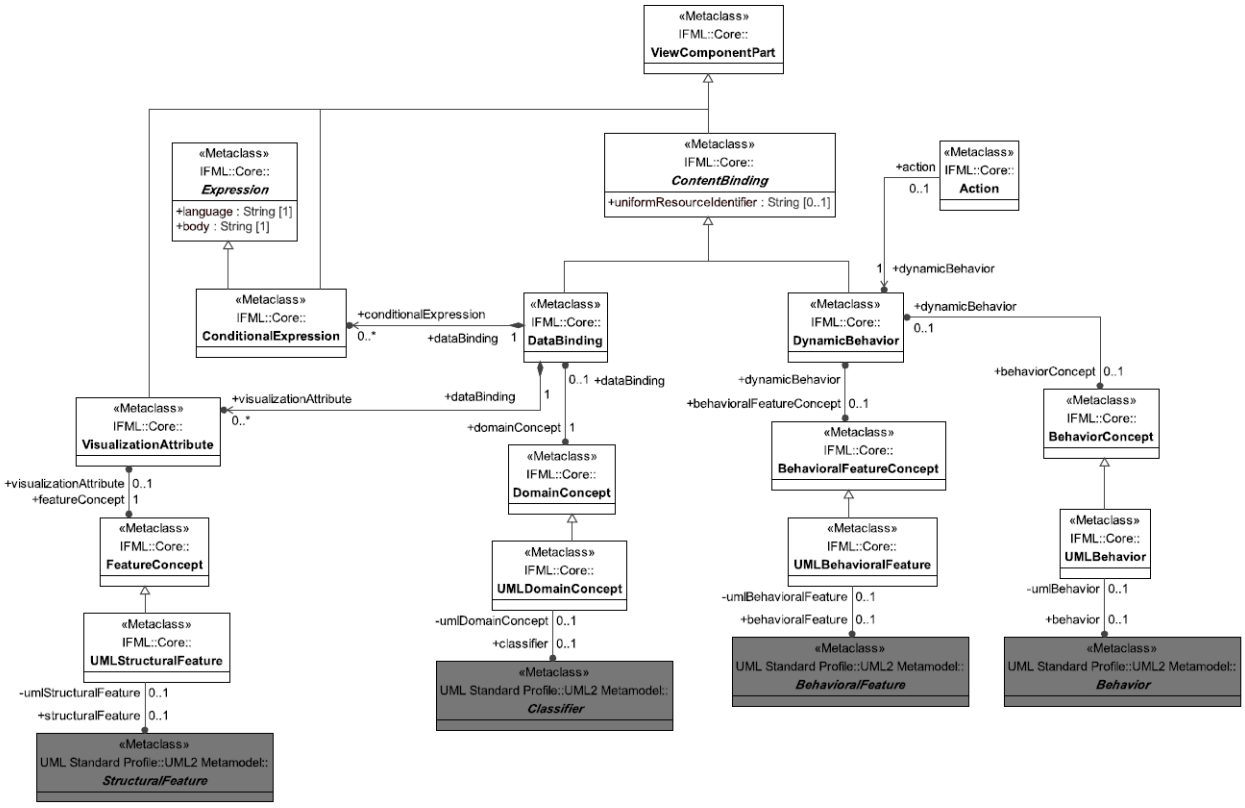
Hình 2.7: Các sự kiện (Events)

Sự kiện xảy ra có thể ảnh hưởng đến trạng thái của ứng dụng, chúng là thể hiện con của các phần tử luồng tương tác. Sự kiện bao gồm hai loại chính: Catching Events (sự kiện được kích hoạt trong giao diện người dùng là khởi đầu cho sự thay đổi về giao diện) và Throwing Events(Sự kiện được tạo ra bởi các giao diện người dùng).

Các ViewElementEvents là các tập con của các ViewElement liên quan, điều này có nghĩa là ViewElement có thể chứa các sự kiện cho phép người dùng tương tác với hệ thống bằng viện tác động vào các liên kết hay các nút. ActionEvents tạo nên các Action liên quan, một Action có thể kích hoạt ActionEvent trong quá trình thực thi của nó hoặc khi nó kết thúc.

SystemEvents là các sự kiện độc lập ở mức độ của mô hình luồng tương tác. SystemEvents là kết quả việc thực thi các Action.

Content Bindings



Hình 2.8: Các nội dung ràng buộc (Content Bindings)

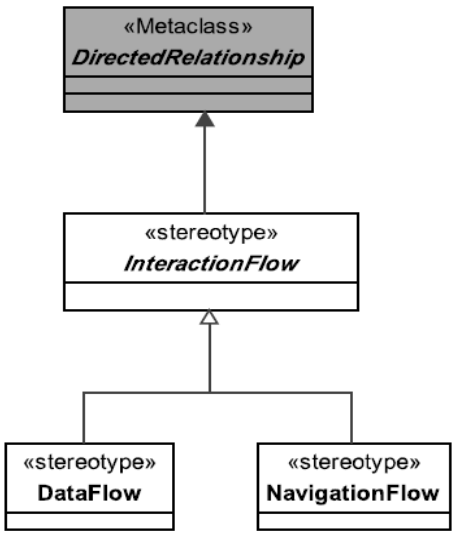
Các thành phần khung nhìn thể hiện nội dung theo các nội dung ràng buộc trong luồng tương tác, Content Binding có thể là bất kỳ kiểu dữ liệu nào. Content Bindings bao gồm hai khái niệm chính : DataBinding (Các ràng buộc dữ liệu có thể là bất kỳ loại dữ liệu nào như file XML, các trường trong một cơ sở dữ liệu...) và Dynamic Behavior (Thể hiện cho một nội dung truy cập, một nội dung logic nghiệp vụ như một service trả về kết quả sau khi gọi tới).

2.3.2.2. IFML UML Profile

IFML UML Profile định nghĩa cú pháp trên cơ sở UML nhằm thể hiện các mô hình IFML. Đặc biệt, UML profile mở rộng các khái niệm về lược đồ UML bao gồm : Class Diagram, State Machine và Composite Structure Diagram. Mục đích của UML Profile nhằm mở rộng các UML Metamodel bằng cách tùy biến nó theo cấu trúc IFML.

UML Profile cho IFML dựa trên việc sử dụng các thành phần của UML bao gồm cả thành phần cơ bản và thành phần đóng gói (packaging components), các lớp và các khái niệm khác.

Một ví dụ về IFML UML Profile được thể hiện trong Hình 2.9.



Hình 2.9: Khuôn mẫu thành phần luồng tương tác

Các mô tả và thể hiện đồ họa được thể hiện trong Bảng 2.1.

Bảng 2.1: Bảng khuôn mẫu thành phần luồng tương tác.

Stereotype	UML Metaclass	Tagged Values	Constraints	Icon
«DataFlow»	UML::Kernel::DirectedRelationShip		Must be associated with a ParameterBinding	
«InteractionFlow»	UML::Kernel::DirectedRelationShip			
«NavigationFlow»	UML::Kernel::DirectedRelationShip			

Khuôn mẫu InteractionFlow thuộc về Metaclass DirectedRelationship được thể hiện bởi hai khuôn mẫu con là DataFlow và Navigation Flow. DataFlow có ràng buộc là phải được kết hợp với các tham số ràng buộc.

2.3.2.3. IFML Visual Syntax

IFML Visual Syntax cung cấp các cú pháp trực quan chuyên biệt để thể hiện các mô hình IFML một cách ngắn ngọn. Cụ thể, nó cung cấp một lược đồ đặc biệt có khả năng thể hiện các khía cạnh của giao diện người dùng riêng biệt với các lược đồ IFML.

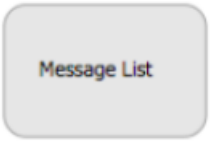
2.3.2.4. IFML XMI

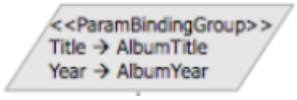
Cung cấp định dạng chuyển mô hình, dành riêng cho các công cụ độc lập. Tiêu biểu là lược đồ trao đổi (IFML Diagram Interchange - IFML DI). IFML DI tạo điều kiện trao đổi lược đồ IFML giữa các công cụ mô hình hóa hơn là dùng để thể hiện các lược đồ trong từng công cụ riêng biệt. Các Metamodel của IFML DI được định nghĩa theo cơ sở MOF metamodel, do đó, các thể hiện của nó có thể được tuần tự hóa và chuyển đổi sử dụng XMI.

2.3.3. Cú pháp cụ thể dạng đồ họa của IFML

IFML bao gồm 7 khái niệm chính và các thể hiện đồ họa tương ứng được biểu diễn trong Bảng 2.2 [22].

Bảng 2.2: Các khái niệm chính của IFML

Khái niệm	Ý nghĩa	Thể hiện đồ họa	Ví dụ
View Container	Một phần tử của giao diện bao gồm các nội dung hiển thị và hỗ trợ tương tác với các View Container khác		Màn hình nền hiển thị đăng nhập.
View Component	Một phần tử của giao diện để hiển thị các nội dung hay các tham số đầu vào		Một list view thể hiện danh sách các thành viên trong một nhóm.
Event	Một sự kiện xảy ra và tác động đến trạng thái của ứng dụng		Sự kiện đăng nhập
Action	Một phần của logic nghiệp vụ được gọi bởi Event		Hành động đăng nhập

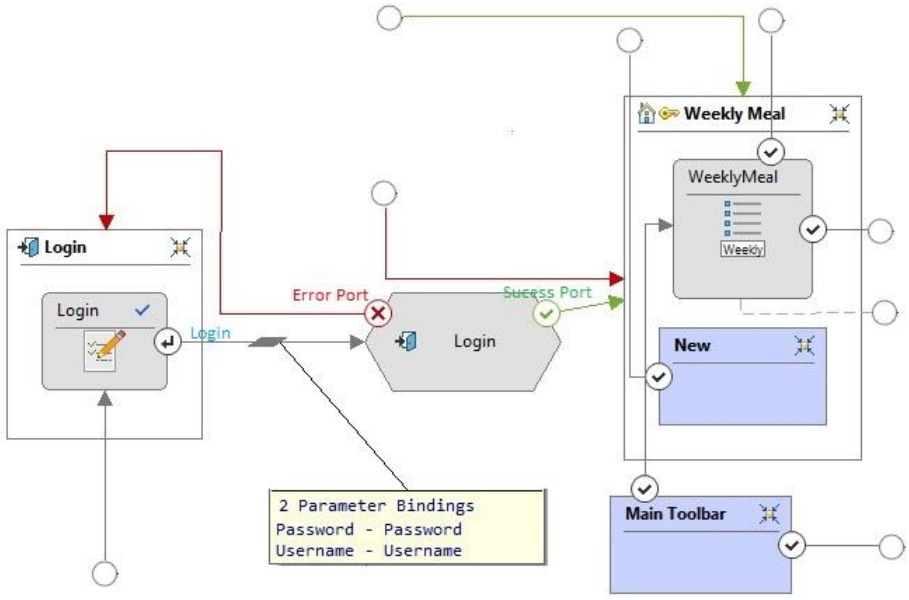
Navigation Flow	Một ràng buộc tham số đầu vào và đầu ra. Nguồn được liên kết với các tham số đầu ra mà nó nhận được và chuyển tham số đầu vào tới đích		Chuyển từ màn hình danh sách nhân viên đến màn hình thông tin chi tiết với tham số là mã nhân viên bằng Navigation Flow
Data Flow	Chuyển dữ liệu giữa các View Components hay Action như là kết quả của tương tác người dùng trước đó		
Parameter Binding Group	Tập hợp các tham số ràng buộc được kèm theo luồng tương tác (như Navigation Flow hoặc Data Flow)		

Một lược đồ IFML bao gồm một hoặc nhiều View Container là một giao diện bao gồm một hoặc nhiều phần tử cho phép thể hiện nội dung hiển thị và các tương tác hỗ trợ. Như ví dụ ca sử dụng Login trong Hình 2.10, View Container là khung nhìn Login nơi chứa một View Component bao gồm các trường để thể hiện nội dung và tương tác với người dùng (trường tài khoản - username, mật khẩu - password và nút đăng nhập - Login).

Đối chiếu với một ứng dụng Model - View - Controller truyền thống, IFML trọng tâm về phần View, IFML mô tả cách thức các View tham chiếu hay phụ thuộc vào Model và Control của ứng dụng. Mối quan hệ giữa IFML và mô hình MVC bao gồm [22]:

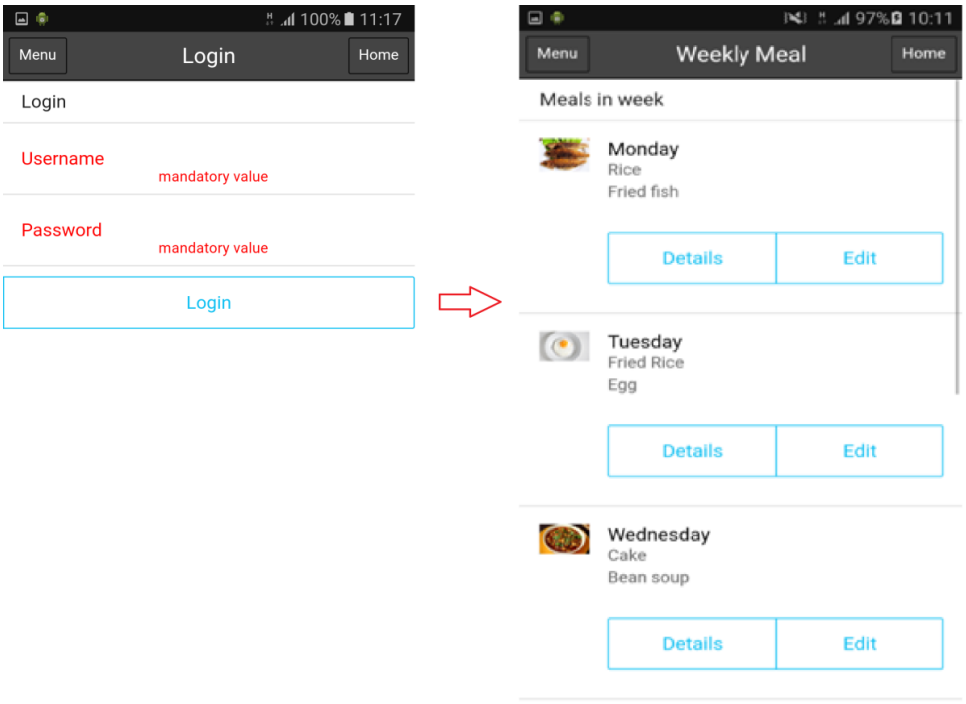
- Với thành phần View: IFML thể hiện các thành phần giao diện, các mô tả và thể hiện các dữ liệu sẽ tương tác trực tiếp với người dùng.
- Với thành phần Controller: IFML cho phép nhà phát triển xác định những ảnh hưởng của tương tác người dùng và các sự kiện hệ thống. Xây dựng các sự kiện liên quan mà các Controller sẽ điều khiển.

- Với thành phần Model: IFML cho phép đặc tả các tham chiếu đến các đối tượng dữ liệu nhằm thể hiện trạng thái của ứng dụng và đưa ra các hành động phù hợp với tương tác của người dùng.



Hình 2.10: Màn hình đăng nhập login với thể hiện IFML

Một cách trực quan, có thể quan sát dạng thể hiện của ứng dụng đầu cuối tương ứng với ca sử đụng đăng nhập như Hình 2.11.



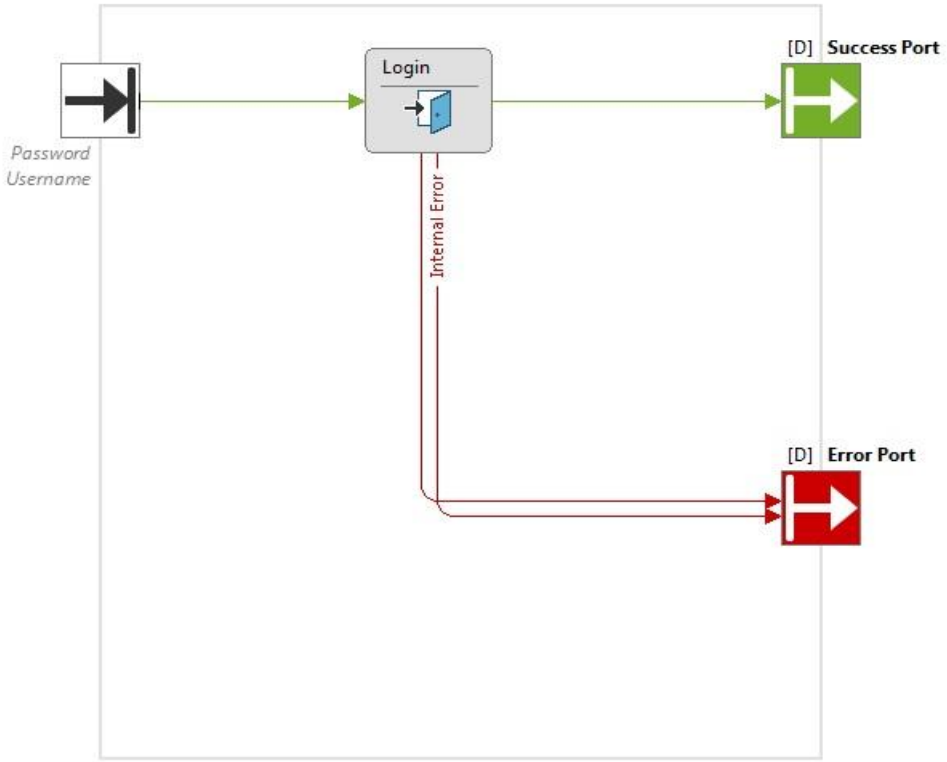
Hình 2.11: Màn hình đăng nhập login trong ứng dụng đầu cuối

Dưới thể hiện đầu cuối, **View Container** là màn hình Login và Weekly Meal sau khi loại bỏ các thành phần View Component.

View Component là thành phần thể hiện dữ liệu, tương tác trực tiếp với người dùng được chứa trong View Container. Ở ví dụ trên ứng dụng đầu cuối, View Component tương ứng với trường username/password/nút Login.

Event tương ứng với sự kiện Login có thể nhìn thấy trên View Container, tuy nhiên Event không trực quan như View Component nên không thể nhìn thấy trên giao diện ứng dụng đầu cuối. Event là sự kiện khi người dùng ấn nút Login để đăng nhập vào hệ thống.

Action tương tự như Event cũng không phải là đối tượng trực quan tương tác trực tiếp với người dùng, như ví được thể hiện trong Hình 2.10, Action ở đây là Login. Action được gọi bởi sự kiện, khi sự kiện Login được người dùng kích hoạt bằng cách ấn nút Login, sự kiện Login sẽ được kích hoạt nhằm kiểm tra xác thực của thông tin đăng nhập. Bên trong Action là các luồng xử lý logic nhằm đưa ra kết quả cuối cùng như Hình 2.12.



Hình 2.12: Minh họa hành động Login

Tại minh họa này, hành động Login được trả về giá trị thành công (success port) hay lỗi (error port) để đưa ra các xử lý tương ứng cho từng trường hợp.

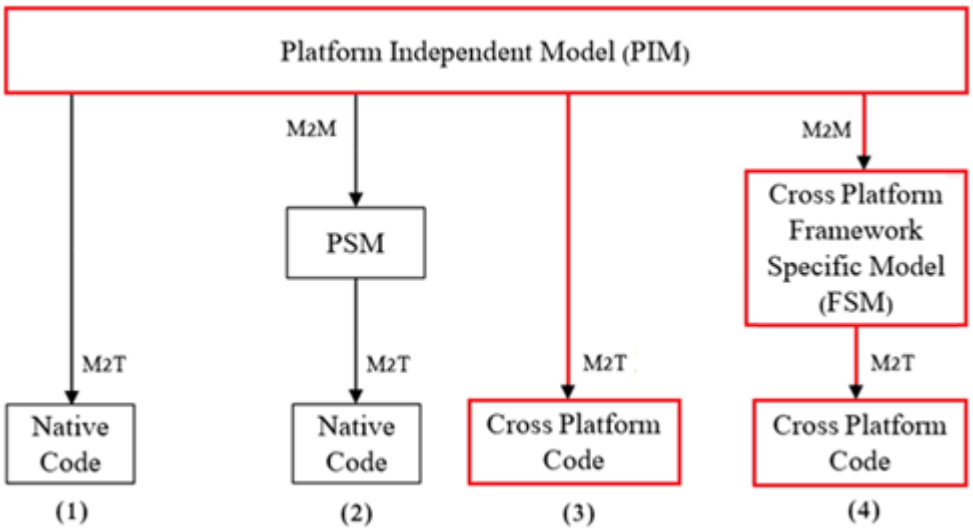
Navigation Flow giúp điều hướng xử lý khi sự kiện Login được gọi bởi người dùng, theo ví dụ Hình 2.10, Navigation Flow điều hướng từ màn hình đăng nhập tới hành động Login và điều hướng tới màn hình tiếp theo nếu xác nhận thành công (success port), trở về màn hình Login nếu xác nhận thất bại (error port).

Data Flow giúp chuyển thông tin dữ liệu giữa các thành phần View cùng tham số ràng buộc nhất định, khác biệt giữa Data Flow thường sử dụng để kết nối các thành phần cùng View Container hoặc trong cùng sự kiện với nhau.

Parameter Binding Group thường đi kèm Navigation Flow hay Data Flow, tập các tham số ràng buộc này giúp chuyển chính xác những thông tin từ nguồn đến đích. Tham chiếu trong ví dụ Hình 2.10 thì tập các tham số ràng buộc ở đây là username và password.

2.3.4. Cơ chế sinh mã nguồn

Ngôn ngữ IFML là một ngôn ngữ cấp độ ngôn ngữ độc lập nền (PIM) trong kiến trúc hướng mô hình [22]. Tác giả đưa ra hai cách tiếp cận thích hợp nhất cho quá trình sinh mã tự động trong phát triển ứng dụng với IFML như Hình 2.13.

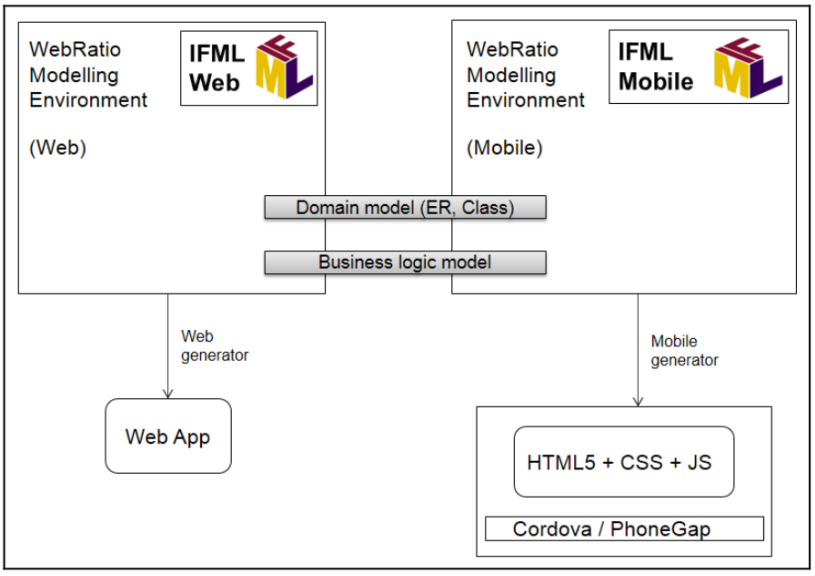


Hình 2.13: Cách tiếp cận sinh mã tự động trong phát triển ứng dụng hướng mô hình với IFML

Về lý thuyết, cách tiếp cận số 3 và số 4 là hai cách tiếp cận chính xác nhất của phương pháp phát triển phần mềm sử dụng kỹ thuật mô hình hóa luồng tương

tác IFML. Mô hình độc lập nền được xây dựng từ một DSL ở đây là ngôn ngữ mô hình hóa luồng tương tác IFML được chuyển đổi thành các mã nguồn đa nền tảng như HTML5, JavaScript... qua bộ chuyển đổi M2T (cách tiếp cận số 3) hoặc qua bước trung gian nhằm chuyển đổi thành các mô hình phụ thuộc framework đa nền tảng sử dụng bộ chuyển đổi M2M (cách tiếp cận số 4). Các mã nguồn này được sử dụng trực tiếp để xây dựng phần mềm trên nền tảng Web hoặc gửi đến framework chuyên biệt để chuyển đổi thành các ứng dụng trên từng nền tảng riêng biệt.

WebRatio đưa ra hai ứng dụng chính và cách tiếp cận của IFML trong phát trển ứng dụng Web và ứng dụng di động được mô tả trong minh họa Hình 2.14 [22]



Hình 2.14 : Ứng dụng IFML trong phát triển phần mềm

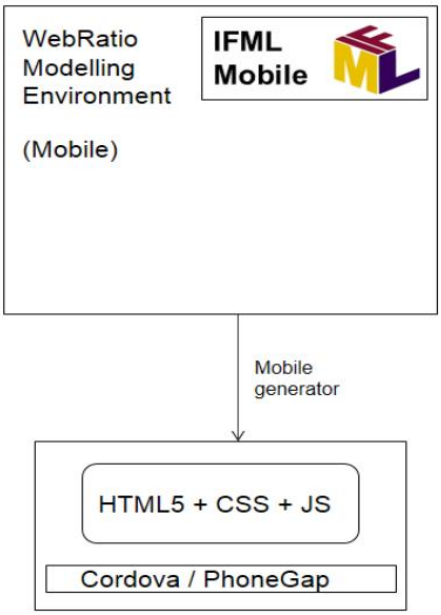
Trong pha thiết kế, mô hình IFML được nhà phát triển thiết kế trong môi trường được cung cấp bởi WebRatio cho phép mô hình hóa một ứng dụng Web thực thụ hoặc một ứng dụng di động. Nó cho phép nhà phát triển chuyển đổi các mô hình như mô hình miền (Domain Model), mô hình logic nghiệp vụ (Business logic model) từ môi trường mô hình hóa này sang môi trường mô hình hóa khác.

Các bộ sinh mã sinh ra các mã nguồn có khả năng chạy và có thể được đóng gói thành các file chạy của ứng dụng có thể chạy trực tiếp trên nền tảng tương ứng. Các ứng dụng được xây dựng trên cở sở tiêu chuẩn JEE-HTML5, iOS và Android.

2.4. Kỹ thuật IFML trong phát triển ứng dụng di động

WebRatio phát triển Webratio Mobile Platform (WMP) là một IDE dựa trên Eclipse nhằm hỗ trợ phát triển ứng dụng di động sử dụng kỹ thuật IFML. WMP sử

dụng PhoneGap phiên bản Cordova như một công cụ trung gian nhằm xây dựng ứng dụng di động như minh họa Hình 2.15 [22].



Hình 2.15: Webratio Mobile Platform và PhoneGap Cordova

Như đã trình bày trong mục *1.3.3.* Phát triển phần mềm hướng mô hình trong lập trình di động, bản chất của PhoneGap trong việc xây dựng các ứng dụng di động là sử dụng các cấu trúc Web như HTML5, CSS, JS để xây dựng một ứng dụng chạy trên nền tảng Web nên WMP cũng thừa hưởng khả năng này. Tuy nhiên, WMP và PhoneGap cũng cung cấp các API cho phép sử dụng trực tiếp tính năng gốc nhằm nâng cao hiệu suất ứng dụng và tăng trải nghiệm người dùng.

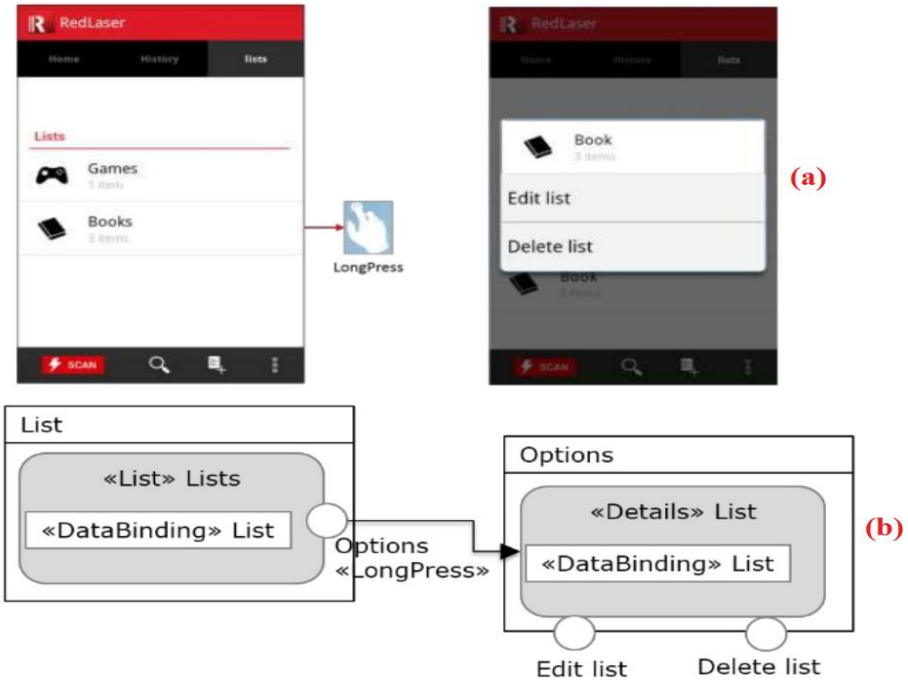
WMP tập trung vào các khía cạnh chính sau đây:

- Mô hình miền: Hỗ trợ thiết kế các mô hình miền sử dụng lược đồ lớp UML.
- Thiết kế đầu cuối: Hỗ trợ thiết kế lược đồ IFML bao gồm cả việc xây dựng cấu trúc IFML và mở rộng tùy thuộc vào các tính năng do nhà phát triển cung cấp.
- Ánh xạ dữ liệu: Cho phép khai báo và liên kết đến cơ sở dữ liệu như SQL. Các cơ sở dữ liệu thường được sử dụng dạng khách - chủ (client - server) hơn là tổ chức lưu trữ dữ liệu trên chính thiết bị.
- Thiết kế lớp giao diện: WMP cho phép nhà phát triển thiết kế các giao diện nơi người dùng trực tiếp tương tác với hệ thống. Với việc tinh

chỉnh các View Components, View Container hay các View Elements giúp xây dựng được lớp giao diện hoàn chỉnh.

- Sinh mã: WMP tự động sinh mã từ các mô hình IFML, mẫu giao diện hay các sự kiện thể hiện trên lược đồ IFML thành các ứng dụng dạng gốc như Android hay iOS. Việc sinh mã này được thực hiện trực tiếp trên máy chủ đám mây (Cloud server) được cung cấp bởi nhà phát hành công cụ.

Việc ứng dụng IFML trong lĩnh vực di động còn giúp thể hiện những tương tác phức tạp của người dùng. Hình 2.16 mô tả một trường hợp cụ thể cho tương tác "chạm giữ"(Long press):



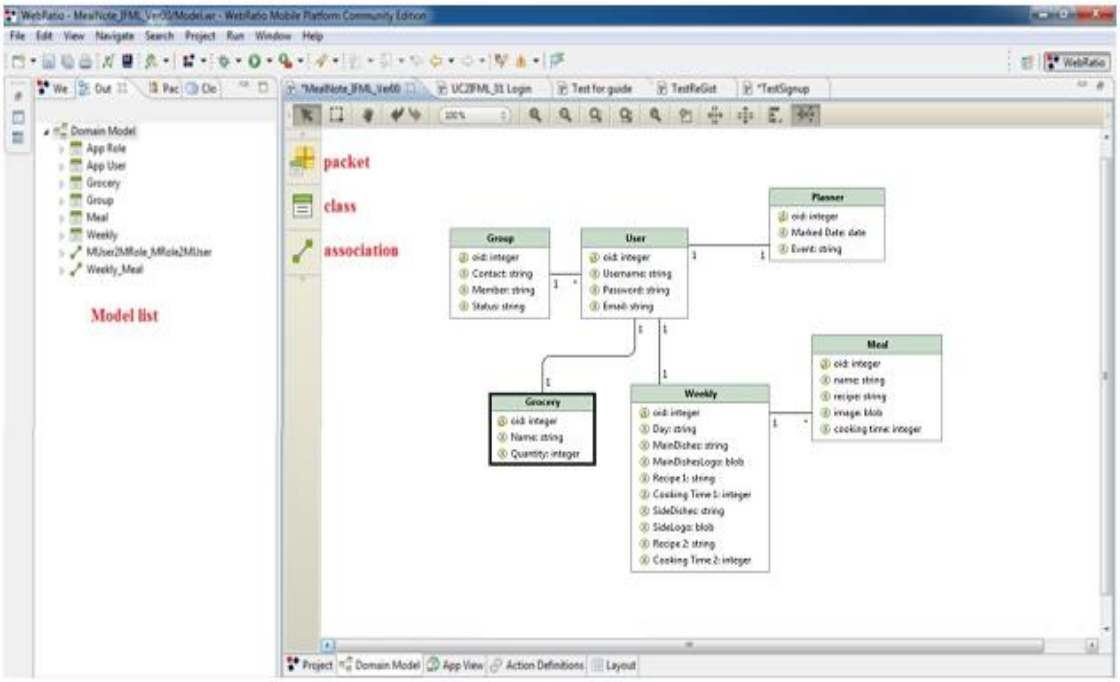
Hình 2.16: Sự kiện được sinh bởi tương tác người dùng.

Sự kiện Options được kích hoạt bởi tương tác người dùng bằng tương tác "chạm giữ" từ ứng dụng trên Hình 2.16 (a) được mô hình hóa với IFML như Hình 2.16 (b).

WebRatio cho phép nhà phát triển xây dựng lược đồ tương tác IFML của ứng dụng, từ đó sinh mã nguồn dưới dạng file chạy trực tiếp của ứng dụng. Phần tiếp theo, nội dung luận văn trình bày các bước xây dựng một ứng dụng di động sử dụng kỹ thuật IFML với công cụ mô hình hóa luồng tương tác-WMP.

2.4.1. Mô hình miền

Mô hình miền được sử dụng theo chuẩn UML, WMP cho phép xây dựng mô hình miền trực tiếp trên ứng dụng với thẻ Domain Model. Mô hình miền được xây dựng theo chuẩn UML thông thường. Đây là cơ sở thực hiện mô hình hóa luồng tương tác cho các bước tiếp theo. Hình 2.17 thể hiện màn hình xây dựng mô hình miền với WMP.

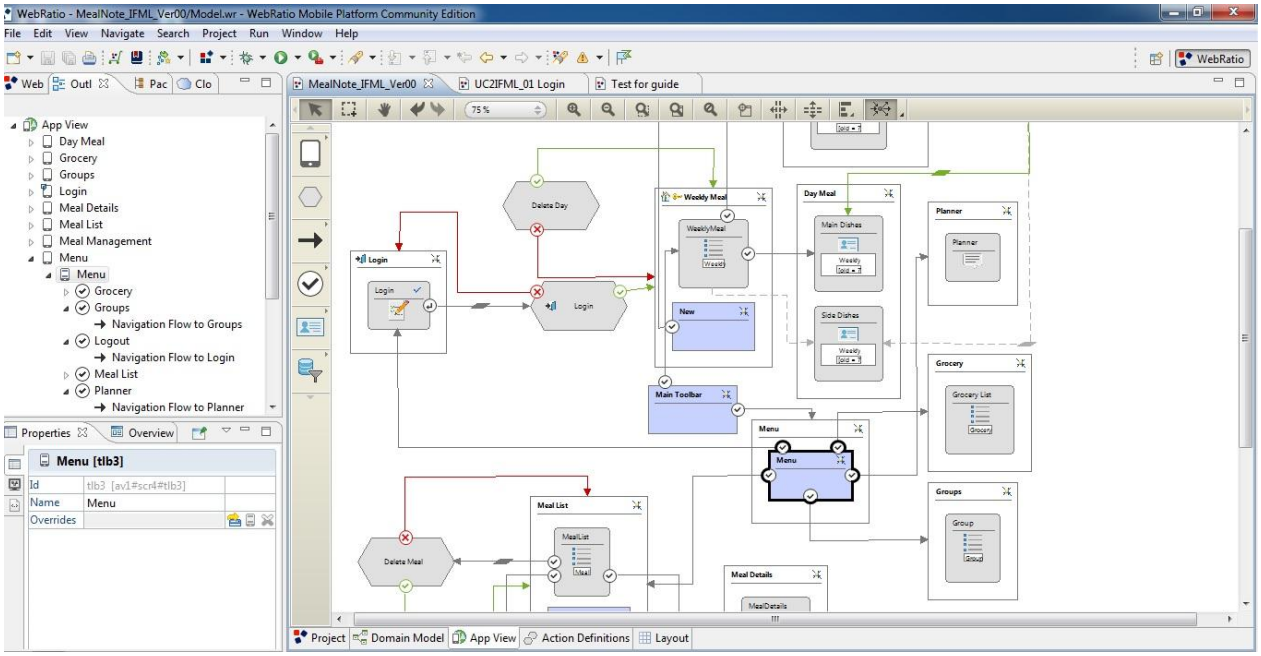


Hình 2.17: Domain Model với WMP

Trong đó, nhà phát triển có thể thêm các class, packet hay association với các tính năng được tích hợp sẵn trên WMP.

2.4.2. Mô hình hóa luồng tương tác

Thực hiện tạo mới Mobile Project và ở App View để bắt đầu xây dựng mô hình hóa. Các khái niệm chính của IFML và thành phần của WMP được mô tả trong minh họa Hình 2.18.



Hình 2.18: Mô hình hóa luồng tương tác với WMP

Các phần tử mô hình luồng tương tác IFML được thể hiện trên WMP bao gồm :

- **View Container**: bao gồm một màn hình (Screen), nhiều màn hình (Screen Set) và màn hình thanh công cụ (Toolbar)

- **View Component**: Bao gồm View chi tiết (Details) để thể hiện các thông tin cùng trong một trường, danh sách (List) để thể hiện một danh sách thông tin, biểu mẫu (Form) cho phép người dùng tương tác với thông tin và cấu trúc (Hierarchy) để thể hiện các thông tin dạng cấu trúc. Khi sử dụng View Component, nhà phát triển phải chỉnh sửa bố cục một cách hợp lý trong thẻ Layout

- **Action:** Để định nghĩa và kết nối các hành động, khi thêm một hành động mới, nhà phát triển phải khai báo hành động này trong thẻ Action Definition.

- **Flow:** Bao gồm Navigation Flow cho phép điều hướng giữa các view khác nhau kèm theo các tham số ràng buộc hoặc sử dụng tham số ràng buộc mặc định, Data Flow cho phép chuyển thông tin dữ liệu giữa các View Component. Các tham số ràng buộc được thêm vào bởi nhà phát triển, nếu không sẽ không có tham số ràng buộc cho Navigation Flow tham số ràng buộc mặc định cho Data Flow.

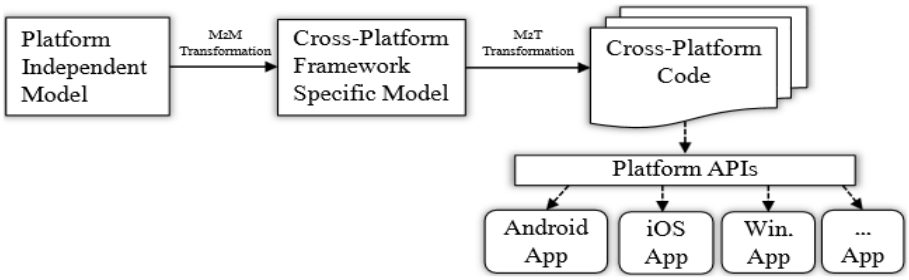
- **Events:** Bao gồm sự kiện chọn (On Select Event) xử lý những tương tác chọn tương ứng của người dùng, sự kiện thông báo (Notification Event) nhằm xử lý

những sự kiện thông báo từ hệ thống hoặc đến người dùng, sự kiện gửi đi (Submit Event) nhằm xử lý các tương tác xác nhận hoặc gửi đi thông tin từ người dùng và sự kiện trở lại (Back Event) nhằm quay lại màn hình cũ.

- **Utility Components:** Là tính năng được cung cấp nằm ngoài khuôn khổ đặc tả kỹ thuật IFML để dành riêng cho lập trình ứng dụng di động, bao gồm Selector cho phép tương tác với cơ sở dữ liệu, Barcode, Calendar và Map cho phép sử dụng trực tiếp các tính năng gốc tương ứng trên thiết bị.

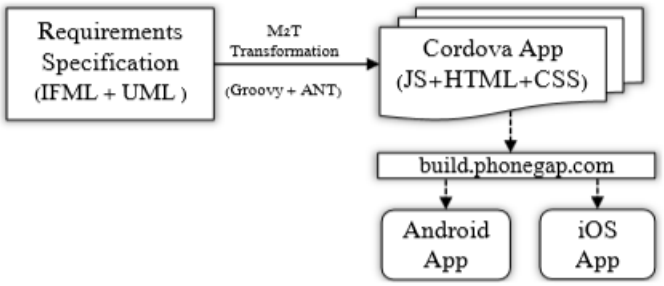
2.4.3. Cơ chế sinh mã nguồn trong lĩnh vực di động

Qua trình bày trong mục *2.3.4.* Cơ chế sinh mã nguồn, cách tiếp cận số 4 áp dụng IFML trong lĩnh vực di động được diễn giải như Hình 2.19:



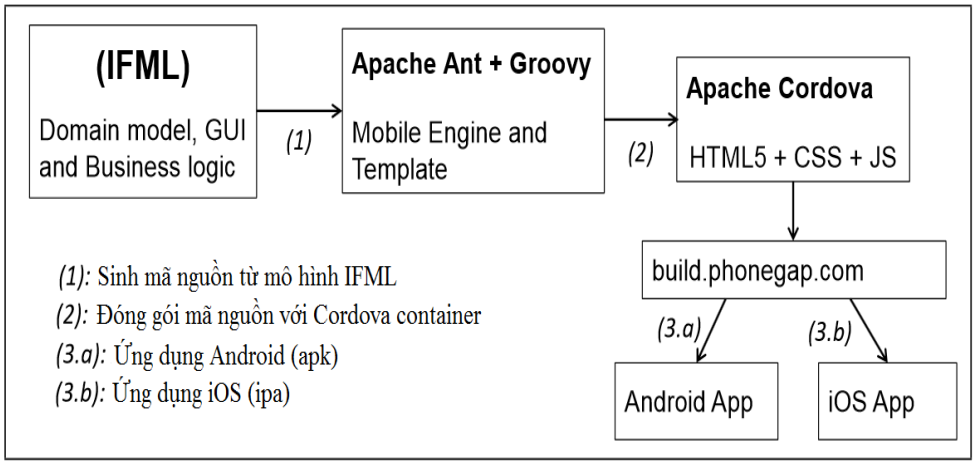
Hình 2.19: Cách tiếp cận PIM - CFS - CPC [8]

PIM biểu diễn bởi IFML được chuyển đổi thành mô hình phụ thuộc framework đa nền tảng (CFS) qua bộ chuyển đổi M2M. Sau đó, CFS được chuyển đổi thành mã nguồn đa nền tảng qua bộ chuyển đổi M2T và xây dựng thành các ứng dụng di động trên nền tảng tương ứng qua các API của framework đa nền tảng. Tuy nhiên, cách tiếp cận này chưa chính xác cho quá trình phát triển ứng dụng di động với IFML. Qua nội dung nghiên cứu của luận văn, tác giả đề xuất cách tiếp cận cho quá trình sinh mã, xây dựng ứng dụng di động với IFML được thể hiện trong Hình 2.20.



Hình 2.20: Cơ chế sinh mã của IFML trong phát triển ứng dụng di động

WebRatio sử dụng một nền tảng sẵn có cho kỹ thuật sinh mã tự động: Apache Cordova. Ý tưởng chính là sinh mã HTML5, CSS3 và JavaScript từ các mô hình độc lập nền IFML, sau đó, các mã nguồn này được đóng gói bởi framework Cordova và gửi tới máy chủ PhoneGap Build để có thể xây dựng ứng dụng trực tiếp dưới dạng file chạy cho Android (apk) và iOS (ipa). Kiến trúc có thể được mô tả cụ thể hơn trong Hình 2.21[18]:



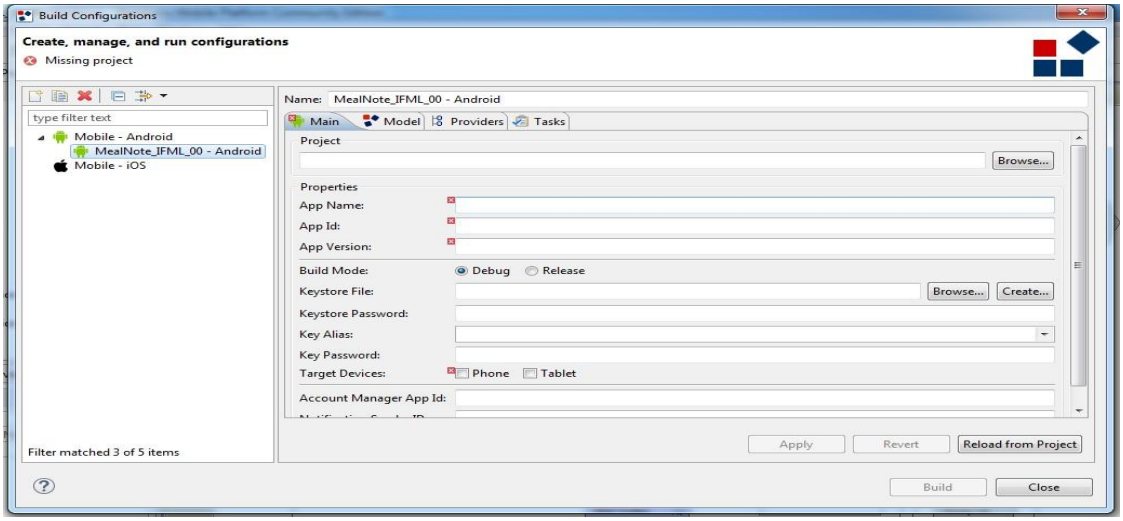
Hình 2.21: Kiến trúc phát triển ứng dụng di động đa nền tảng của IFML.

Bộ sinh mã phân tích mô hình đầu vào tuần tự hóa dưới dạng các file XML (XMI) với tất cả thông tin về dữ liệu ứng dụng, dữ liệu tương tác và tạo ra tất cả các mã JavaScript, HTML cần thiết để có thể chạy ứng dụng dưới dạng ứng dụng di động trên nền tảng Web. Như vậy, bản chất việc xây dựng ứng dụng di động với IFML là việc chuyển đổi sử dụng mô hình M2T để sinh các mã JavaScript, HTML từ các mô hình IFML. Các công đoạn còn lại để tạo nên các ứng dụng di động được WebRatio sử dụng máy chủ của Adobe là PhoneGap - Cordova. Việc thể hiện ứng dụng dưới dạng "nhìn và cảm nhận" được dựa trên các file CSS3 cơ bản, chúng chứa các luật chung cho mẫu thiết kế tương tác ứng dụng di động. Tuy nhiên, WebRatio không khuyến khích các nhà phát triển thay đổi mã nguồn của ứng dụng được sinh ra mà thay vào đó là việc thay đổi các luật chuyển nhằm tạo một ứng dụng hoàn hảo hơn [18].

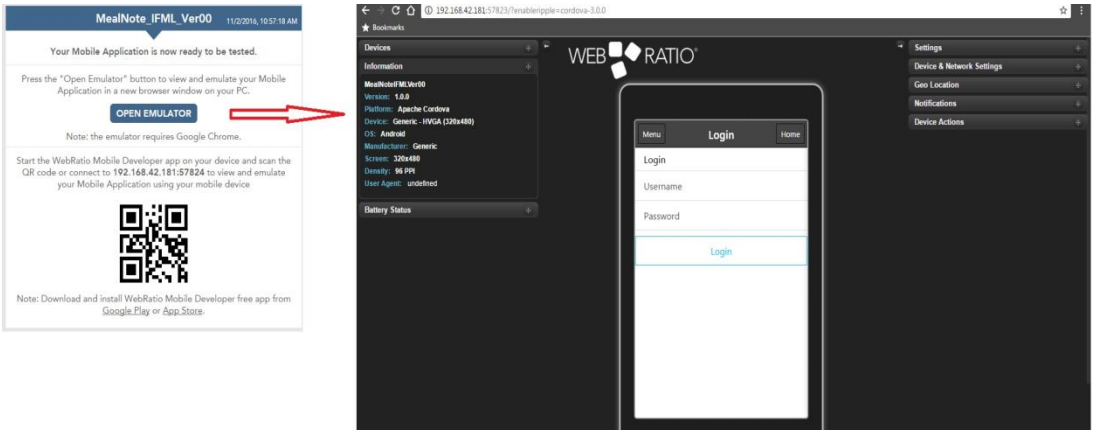
2.4.4. Sinh ứng dụng

WebRatio cho phép nhà phát triển sinh ứng dụng và giả lập trên máy chủ đám mây của họ hoặc sinh ứng dụng gốc trực tiếp dưới dạng file chạy (apk với Android hay ipa với iOS). Để thực hiện việc này, sử dụng tính năng Generate and Run cho việc chạy ứng dụng trên thiết bị giả lập hay Build để sinh trực tiếp ứng

dụng dưới dạng file chạy cho hệ điều hành di động tương ứng. Hình 2.22 mô tả màn hình sinh ứng dụng dưới dạng file chạy và Hình 2.23 thể hiện màn hình ứng dụng giả lập trên máy chủ đám mây. Một phương pháp sinh ứng dụng khác trực tiếp trên kho ứng dụng của hệ điều hành như Google Play hay Apple App Store là sử dụng phần mềm WebRatio Mobile Developer để chụp ảnh Bar Code trên màn hình giao diện máy chủ đám mây, cách sẽ đưa nhà phát triển đến ứng dụng của mình trên kho ứng dụng thể hiện trên minh họa Hình 2.24.



Hình 2.22: Sinh ứng dụng gốc với tính năng Build



Hình 2.23: Giả lập ứng dụng với máy chủ đám mây

Hình 2.24: Tích hợp ứng dụng thực tế qua kho ứng dụng sử dụng phần mềm WebRatio Mobile Developer.

2.4.5. Một số vấn đề đặt ra cho phương pháp mô hình hóa luồng tương tác

Với nhiều ưu điểm trong quá trình phát triển phần mềm, đặc biệt trong lĩnh vực phát triển ứng dụng di động, IFML là một kỹ thuật tốt cho việc cải thiện, nâng cao hiệu suất trong phát triển phần mềm. Tuy nhiên, còn nhiều câu hỏi về hiệu quả ứng dụng kỹ thuật IFML vào phát triển ứng dụng di động như:

(1) Có thể áp dụng kỹ thuật IFML để xây dựng ứng dụng di động trên tất cả các lĩnh vực không (trò chơi, ứng dụng tiện ích, ứng dụng giải trí...)?
(2) Chi phí cho quá trình phát triển phần mềm với IFML so với phương pháp truyền thống ra sao?
(3) Giao diện của các ứng dụng xây dựng với kỹ thuật IFML trên quan điểm người dùng và nhà phát triển có tốt không?
(4) Các ứng dụng được xây dựng với IFML có tương thích tốt với phần cứng và hệ điều hành hay không?
(5) Hiệu suất của ứng dụng được xây dựng với IFML có tốt không? Trải nghiệm người dùng thế nào?
(6) Tốc độ (thời gian) phát triển ứng dụng với IFML có nhanh không?
(7) Các công việc sau khi triển khai ứng dụng bao gồm: bảo trì, nâng cấp của ứng dụng có khả thi không? Hiệu quả ra sao?

(8) Trên quan điểm nhà phát triển, các công cụ, thư viện hỗ trợ để xây dựng ứng dụng với IFML có tốt không?

Với mục tiêu làm rõ những câu hỏi đặt ra cho quá trình xây dựng ứng dụng di động với kỹ thuật mô hình hóa luồng tương tác IFML. Luận văn sẽ đề cập chi tiết hơn đến việc đề xuất các tiêu chí và thực hiện đánh giá kỹ thuật này trong phần tiếp theo.

2.5. Các tiêu chí và phương pháp đánh giá kỹ thuật IFML trong phát triển ứng dụng di động

Bản chất của việc xây dựng ứng dụng di động sử dụng kỹ thuật IFML là việc sinh mã nguồn dạng JavaScript, HTML5 từ các mô hình IFML, đóng gói thành các ứng dụng di động bởi PhoneGap-Cordova và các nội dung ứng dụng được thể hiện dưới sự hỗ trợ của các file CSS3. Do đó, các ứng dụng di động được xây dựng bằng IFML (IFML mobile app) là các ứng dụng dạng lai giữa ứng dụng nền tảng web và ứng dụng gốc.

Ứng dụng lai (Hybrid) app là ứng đụng được chạy trên nền tảng Web của hệ điều hành di động và có khả năng sử dụng các tính năng gốc một cách hoàn hảo như: Bộ nhớ, chụp ảnh, bản đồ, cảm biến... Phân biệt với ứng dụng dạng Web được thể hiện hoàn toàn trên nền tảng Web và ứng dụng gốc được xây dựng để phục vụ duy nhất cho một nền tảng.

Điều này có nghĩa là IFML mobile app có thể thừa hưởng toàn bộ ưu điểm cũng như hạn chế của phương pháp xây dựng ứng dụng lai. Dựa theo các nghiên cứu, báo cáo khoa học của các tác giả: *Anmol Khandeparkar [3], Henning Heitkötter [12], Sanjeet Dhillon[19], Linus Oberg [15]* về khảo sát, đánh giá phương pháp phát triển ứng dụng di động đa nền tảng (

Bảng 2.3) và kết quả tìm hiểu của luận văn này trên ưu nhược điểm của kỹ thuật IFML, với mục tiêu đánh giá kỹ thuật mô hình hóa luồng tương tác IFML trong phát triển ứng dụng di động một cách cụ thể và chi tiết, tác giả đề xuất các tiêu chí đánh giá như sau:

- Khả năng xác định yêu cầu và tính khả thi của ứng dụng.
- Chi phí phát triển ứng dụng.
- Thiết kế và giao diện.
- Khả năng hỗ trợ tính năng phần cứng và hệ điều hành.

- Hiệu suất ứng dụng và trải nghiệm người dùng.
- Thời gian phát triển ứng dụng.
- Khả năng bảo trì, nâng cấp và bảo mật ứng dụng.
- Các tiêu chí khác: Các ứng dụng và thư viện hỗ trợ, Công cụ phát triển ứng dụng và sửa lỗi, Sự độc lập về nền tảng.

Ứng dụng gốc là tiêu chuẩn của mỗi khách hàng, nhà phát triển mong muốn xây dựng ứng dụng di động mới do chúng có khả năng tận dụng tối đa các hỗ trợ của thiết bị/hệ điều hành. Dựa vào phương pháp đánh giá của các báo cáo khoa học, nghiên cứu [3, 12, 19, 15] về phát triển ứng dụng di động nói chung và ứng dụng di động đa nền tảng nói riêng. Phương pháp đánh giá kỹ thuật IFML được đề xuất là so sánh chúng với kỹ thuật xây dựng ứng dụng gốc truyền thống. Luận văn này sẽ thể hiện các kết quả đánh giá bằng cách xây dựng hai ứng dụng di động độc lập trên nền tảng Android với nội dung giống nhau sử dụng hai kỹ thuật: Kỹ thuật mô hình hóa luồng tương tác IFML và phương pháp truyền thống là kỹ thuật xây dựng ứng dụng gốc. Thực hiện khảo nghiệm và so sánh hai ứng dụng trên cùng một thiết bị vật lý. Qua đó đưa ra kết luận dựa theo các tiêu chí đã nêu ra về kỹ thuật IFML.

Bảng 2.3: Tổng hợp các tiêu chí đánh giá ứng dụng đa nền tảng

Stt	Anmol Khandeparkar	Henning Heitkötter	Sanjeet Dhillon	Linus Oberg
1	Thiết kế của giao diện	Giấy phép và chi phí	Môi trường phát triển	Tài liệu và hỗ trợ
2	Chi phí	Nền tảng hỗ trợ	Trải nghiệm người dùng	Khả năng bảo trì
3	Thời gian phát triển	Truy cập tính năng gốc	Truy cập tính năng thiết bị	Tốc độ phát triển
4	Trải nghiệm người dùng và hiệu suất	Tính khả thi	Cảm biến	Công cụ gỡ lỗi
5	Khả năng bảo trì	Tốc độ ứng dụng	Bản đồ	Hiệu suất
6	Bảo mật	Phân phối tới người dùng	Thông báo	Giao diện, trải nghiệm người dùng
7	Sự độc lập về nền tảng		Bảo mật	Thời gian khởi động

8	Công cụ và gỡ lỗi			Phát triển ứng dụng

2.6. Tổng kết chương

Kỹ thuật mô hình hóa luồng tương tác ra đời nhằm đáp ứng sự phát triển về công nghệ đặc biệt trên lĩnh vực di động nhằm bổ sung cho các kỹ thuật truyền thống. Sự ra đời cho IFML không những giúp bổ sung, làm rõ các mặt còn thiếu mà còn mang tính ứng dụng đột phá khi cho phép phát triển ứng dụng di động hướng mô hình. Lĩnh vực này còn rất sơ khai cho nền tảng có tốc độ phát triển nhanh nhất hiện nay: nền tảng di động.

Việc ứng dụng IFML mang lại nhiều lợi ích, tuy nhiên cũng có không ít điểm hạn chế. Việc sử dụng framework thứ ba cùng ngôn ngữ chạy trên nền tảng Web làm giảm tính ứng dụng, mở rộng của IFML. Chương 3 sẽ đi vào khảo sát và đánh giá chi tiết về vấn đề này.

CHƯƠNG 3 : VẬN DỤNG VÀ THỰC NGHIỆM

Chương 3 đề cập đến quá trình thực nghiệm kỹ thuật IFML trong việc xây dựng ứng dụng di động trên Android: Ứng dụng MealNote. Xây dựng MealNote sử dụng phương pháp truyền thống với kỹ thuật xây dựng ứng dụng gốc và phương pháp MDD với kỹ thuật mô hình hóa luồng tương tác IFML. Nội dung chương 3 cũng trình bày kết quả đánh giá kỹ thuật mô hình hóa luồng tương tác IFML trong phát triển ứng dụng di động nói chung và trên nền tảng Android nói riêng.

3.1. Giới thiệu

Việc thực hiện khảo sát và đánh giá kỹ thuật IFML được thực hiện trên cơ sở so sánh với kỹ thuật xây dựng ứng dụng gốc. Như đã trình bày trong Chương 2, tác giả sẽ thực hiện xây dựng hai ứng dụng độc lập trên hệ điều hành Android có đặc tả yêu cầu giống nhau sử dụng hai phương pháp song song : Phương pháp MDD với kỹ thuật mô hình hóa luồng tương tác IFML và phương pháp truyền thống với kỹ thuật xây dựng ứng dụng gốc. Việc phát triển hai ứng dụng với hai kỹ thuật từ các bước cơ bản nhất nhằm đưa ra các đánh giá đứng trên quan điểm của nhà phát triển. Thêm vào đó, khảo sát hai ứng dụng trên cùng một thiết bị phần cứng chạy hệ điều hành Android (trong luận văn này tác giả sử dụng thiết bị Samsung Galaxy A7 2016) không những đưa ra đánh giá trên quan điểm nhà phát triển mà còn giúp đưa ra đánh giá trên quan điểm người dùng.

Các phần tiếp theo trình bày quá trình xây dựng ứng dụng gốc sử dụng phương pháp truyền thống với mã nguồn Java và công cụ Android Studio. Tiếp theo đó là việc vận dụng kết quả nghiên cứu về IFML để xây dựng ứng dụng một lần nữa sử dụng kỹ thuật IFML.

Phần cuối cùng của chương, luận văn đề cập đến các tiêu chí và kết quả đánh giá kỹ thuật IFML dựa theo kết quả thực nghiệm của các phần trước. Các tiêu chí đánh giá đã được đề cập trong mục *2.5*. Các tiêu chí và phương pháp đánh giá kỹ thuật IFML trong phát triển ứng dụng di động nhằm đưa ra nhận xét cụ thể và chi tiết nhất về kỹ thuật IFML trong phát triển ứng dụng di động.

3.2. Thực nghiệm xây dựng ứng dụng MealNote

Độ phức tạp của kỹ thuật xây dựng ứng dụng gốc là trở ngại chính trong quá trình sử dụng kỹ thuật này. Trái lại, phương pháp MDD với kỹ thuật IFML mang lại nhiều lợi thế về tốc độ phát triển ứng dụng. Chi tiết cụ thể sẽ được trình bày trong các phần sau đây

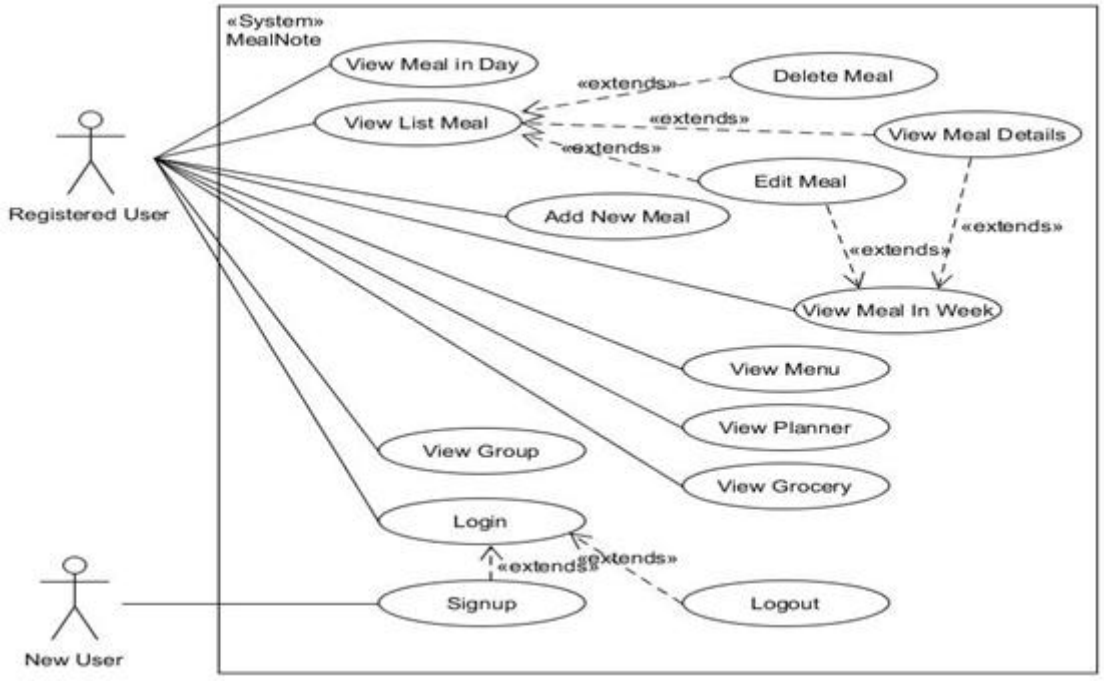
3.2.1. Ứng dụng MealNote

An toàn thực phẩm là một trong những vấn đề cấp thiết nhất hiện nay. Các vụ ngộ độc thực phẩm với tần suất ngày càng tăng là mối lo của toàn xã hội. Ngoài ra, một điều đáng lo ngại nữa đang xảy ra là tỉ lệ ung thư ở Việt Nam thuộc loại cao trên toàn thế giới, một trong những nguyên nhân chính dẫn đến ung thư đến từ thực phẩm không an toàn. Từ thực trạng trên, tác giả muốn xây dựng ứng dụng MealNote nhằm cung cấp riêng cho người Việt một công cụ giúp lên kế hoạch bữa ăn một cách khoa học, sắp xếp bữa ăn trong ngày một cách hợp lý cũng như lựa chọn, đánh dấu thực phẩm, món ăn an toàn hay trao đổi chế độ ăn uống với các người dùng khác.

Trong chương này, luận văn sẽ đề cập tới quá trình xây dựng ứng dụng MealNote với hai quá trình song song: phương pháp truyền thống và phương pháp MDD với kỹ thuật mô hình hóa luồng tương tác IFML. Việc xây dựng một ứng dụng hai lần sử dụng hai phương pháp khác nhau nhằm so sánh và đưa ra đánh giá một cách chi tiết nhất về IFML trên lĩnh vực phát triển phần mềm cho điện thoại di động thông minh.

3.2.2. Đặc tả yêu cầu

3.2.2.1. Biểu đồ ca sử dụng (Use Case Diagram)



Hình 3.1: Biểu đồ ca sử dụng của ứng dụng MealNote

Bảng 3.1: Danh sách các tác nhân và mô tả

Tác nhân	Mô tả tác nhân	Ghi chú
Registered User	Là người sử dụng đã đăng ký tài khoản	
New User	Người sử dụng mới, chưa đăng ký tài khoản	

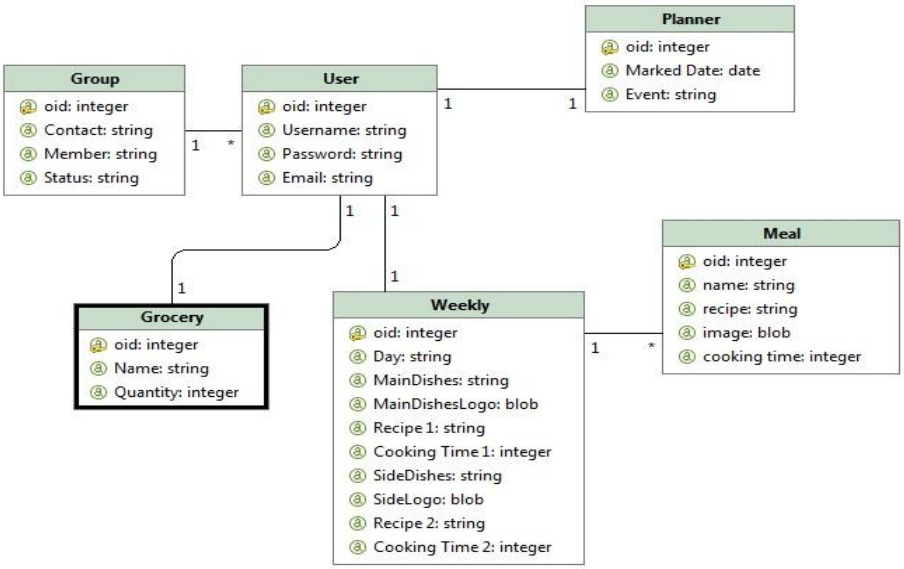
Bảng 3.2: Danh sách các ca sử dụng và mô tả

ID	Tên Use case	Mô tả ngắn gọn Use case	Biểu đồ hoạt động
UC_001	*Login*	*Người dùng đăng nhập hệ thống*	*Hình phụ lục B.*1
UC_002	*Signup*	*Người dùng đăng ký tài khoản mới*	*Hình phụ lục B.*2
UC_003	*View Meal In Week*	*Người dùng xem danh sách các món ăn trong tuần*	*Hình phụ lục B.*3
UC_004	*View Meal in Day*	*Người dùng xem các món ăn theo ngày*	*Hình phụ lục B.*5
UC_005	*View List Meal*	*Người dùng xem danh sách tất cả các món ăn hiện có*	*Hình phụ lục B.*4
UC_006	*View Meal Details*	*Người dùng xem chi tiết một món ăn*	*Hình phụ lục B.*6
UC_007	*Edit Meal*	*Người dùng chỉnh sửa một món ăn*	*Hình phụ lục B.***7**
UC_008	*Delete Meal*	*Người dùng xóa một món ăn*	*Hình phụ lục B.*8

UC_009	*Add New Meal*	*Người dùng thêm mới món ăn*	*Hình phụ lục B.*9
UC_010	*View Menu*	*Người dùng xem danh sách tính năng*	*Hình phụ lục B.*10
UC_011	*View Planner*	*Người dùng xem màn hình lập lịch lên kế hoạch theo ngày*	*Hình phụ lục B.*10
UC_012	*View Grocery*	*Người dùng xem danh sách đồ cần mua*	*Hình phụ lục B.*10
UC_013	*View Group*	*Người dùng truy cập tính năng nhóm, một nhóm bao gồm nhiều thành viên*	*Hình phụ lục B.*10
UC_014	*Logout*	*Người dùng đăng xuất tài khoản*	*Hình phụ lục B.*10

Hình 3.1 biểu diễn sơ đồ ca sử dụng tổng quá của ứng dụng MealNote. Diễn giải các thành phần tác nhân và ca sử dụng được thể hiện chi tiết trong Bảng 3.1 và Bảng 3.2.

3.2.2.2. Mô hình miền (Domain Model)



Hình 3.2: Mô hình miền của ứng dụng MealNote

Lớp User lưu trữ thông tin người dùng bao gồm Username, Password và Email. Lớp meal lưu trữ thông tin món ăn bao gồm tên món ăn (name), công thức món ăn (recipe), hình ảnh đại diện (image) và thời gian nấu. Lớp Weekly bao gồm các thông tin thể hiện món ăn ngày trong tuần bao gồm thứ trong tuần (Day), các trường cho món chính(MainDishes) và món phụ(SideDishes). Lớp Grocery bao gồm các trường mô tả món hàng còn thiếu như tên món hàng (Name) và số lượng cần mua (Quantity). Lớp Group để biểu diễn nhóm của người dùng bao gồm thông tin như thông tin liên hệ (Contact), thông tin tên tài khoản thành viên (Member) và trạng thái đăng nhập (Status).

Mỗi lớp người dùng chỉ có quan hệ 1 - 1 với lớp Weekly, lớp Planner và lớp Grocery. Trong khi đó, lớp Weekly có thể chứa nhiều lớp Meal, lớp Group có thể chứa nhiều User.

3.2.3. Xây dựng ứng dụng MealNote theo phương pháp truyền thống

Việc xây dựng ứng dụng trên Android tương đối phức tạp, cần rất nhiều thời gian để xử lý tỉ mỉ những chi tiết nhỏ, ngoài ra cần đảm bảo yêu cầu khắt khe về các biểu tượng, hình ảnh để có thể sử dụng trong ứng dụng.

Lỗi luôn xuất hiện trong từng bước xây dựng ứng dụng, để lập trình tốt ứng dụng trên Android đòi hỏi người phát triển phải có nền tảng lập trình, kỹ năng gỡ lỗi cũng như giải quyết vấn đề tương đối tốt.

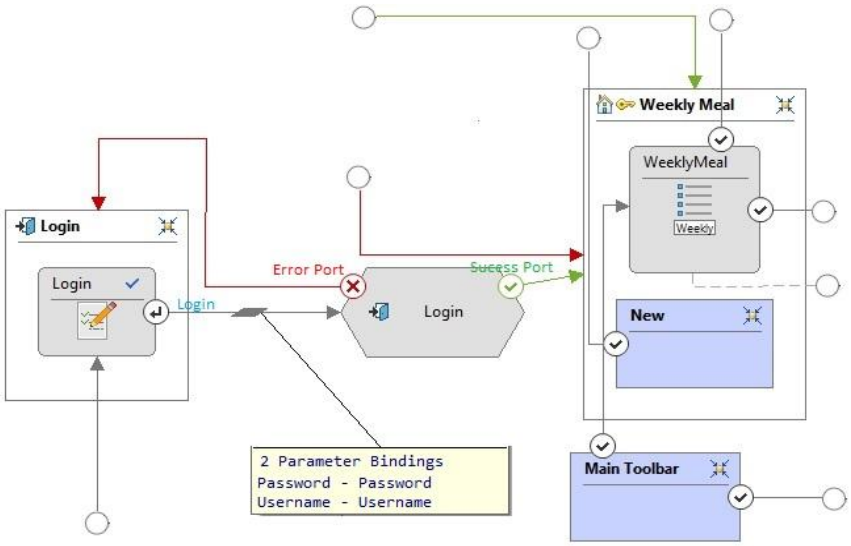
Trái lại, khả năng tùy chỉnh trong quá trình lập trình ứng dụng Android gốc rất cao, giúp nâng cao tối đa trải nghiệm người dùng.

Lập trình Android tốt nhất khi có một thiết bị kiểm thử thật chạy hệ điều hành Android tương ứng, việc sử dụng thiết bị kiểm thử mô phỏng sẽ không bao trùm được toàn bộ lỗi gặp phải

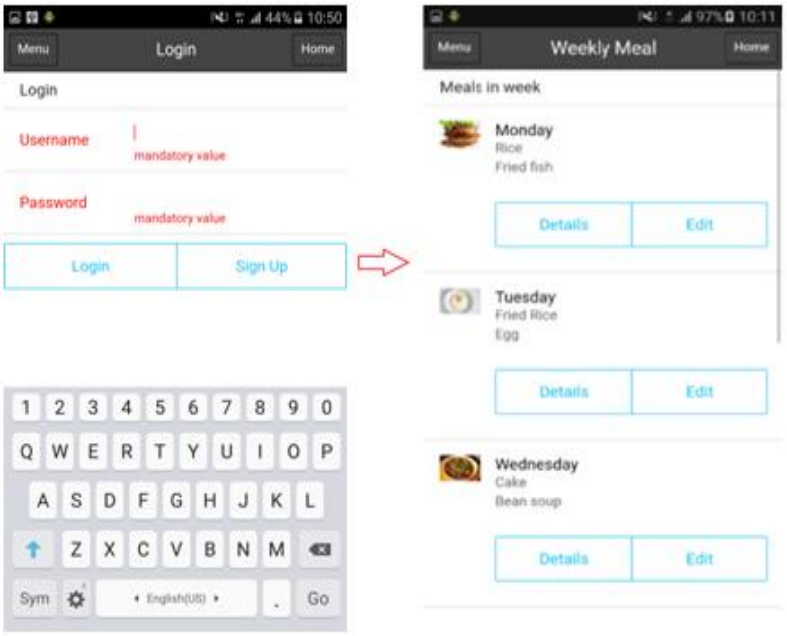
Quá trình xây dựng ứng dụng MealNote theo phương pháp truyền thống được đề cập chi tiết và cụ thể hơn trong *Phụ lục A: Xây dựng ứng dụng MealNote theo phương pháp truyền thống*.

3.2.4. Xây dựng ứng dụng MealNote sử dụng kỹ thuật IFML

3.2.4.1. Mô hình hóa luồng tương tác ca sử dụng Login

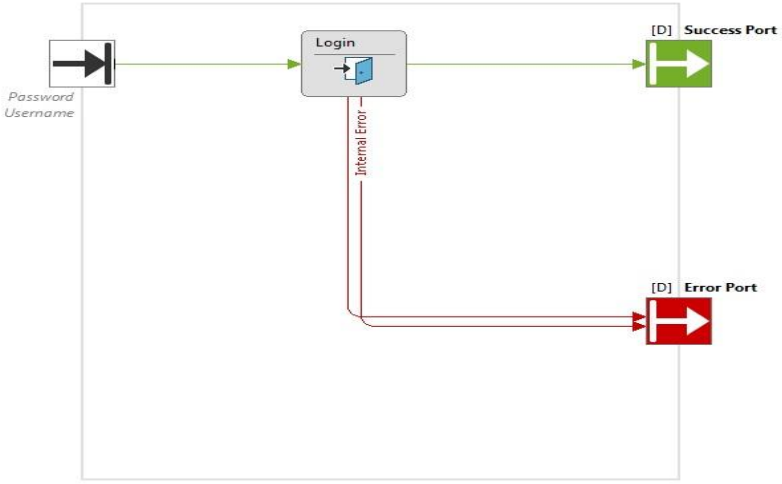


Hình 3.3: Mô hình hóa luồng tương tác ca sử dụng Login



Hình 3.4: Giao diện ca sử dụng Login

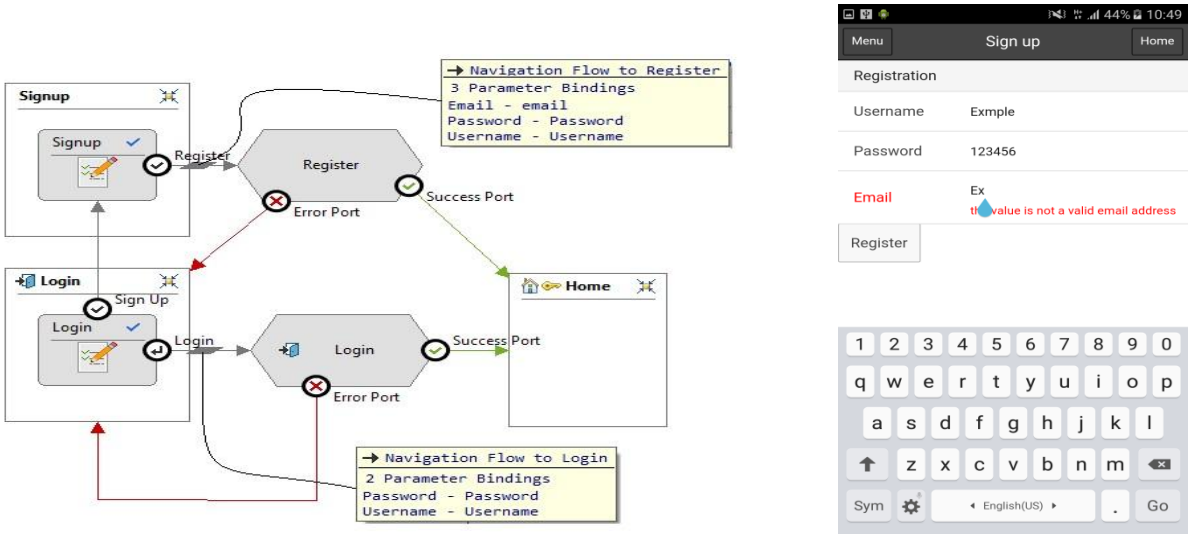
Trong ca sử dụng này, người dùng tương tác với hệ thống bằng sự kiện Login với hai tham số ràng buộc nằm trong mục Parameter Bindings là User name và Password. Các tham số ràng buộc này được đưa tới hành động Login bằng luồng điều hướng (Navigation Flow).



Hình 3.5: Định nghĩa hành động Login

Trong định nghĩa của hành động Login theo Hình 3.5, nếu việc kiểm tra tài khoản và mật khẩu xảy ra lỗi, sẽ trả về cổng lỗi (Error Port), nếu thành công sẽ trả về cổng thành công (Success Port). Từ Error Port sẽ trở về màn hình Login, từ Success Port sẽ điều hướng tới màn hình chính để người dùng có thể thực hiện các tương tác khác.

3.2.4.2. Mô hình hóa luồng tương tác ca sử dụng Signup:

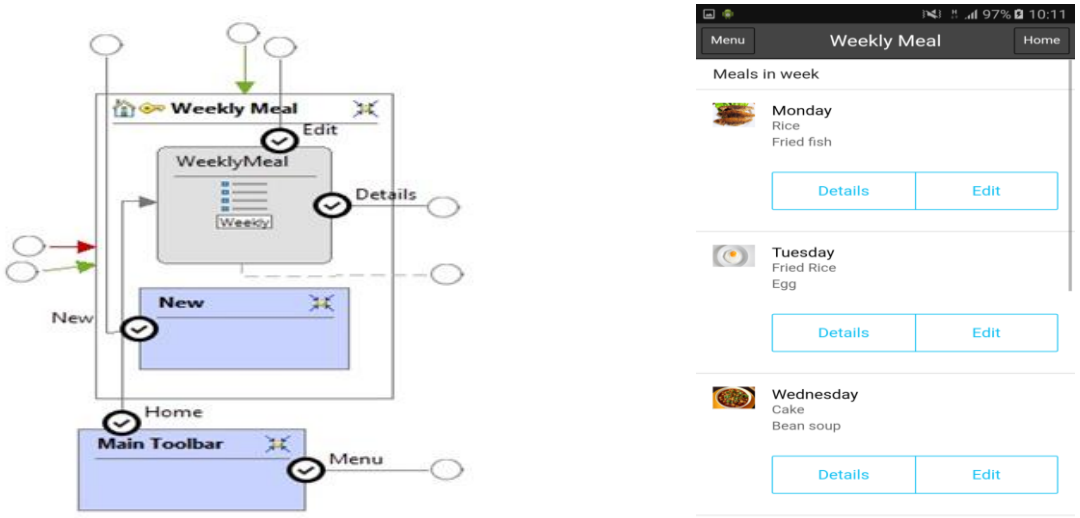


Hình 3.6: Mô hình hóa luồng tương tác ca sử dụng Signup

Ca sử dụng Signup được thể hiện song song với ca sử dụng Login và được kích hoạt bằng sự kiện Signup. Sau khi xác nhận các thông tin đăng ký, hành động Register được kích hoạt bằng tương tác người dùng trên nút Register. Màn hình Login được thể hiện trong trường hợp đăng ký thất bại, màn hình chính của ứng

dụng được thể hiện trong trường hợp đăng ký thành công. Tập tham số ràng buộc được yêu cầu cho quá trình đăng ký bao gồm : Email, Password, Username

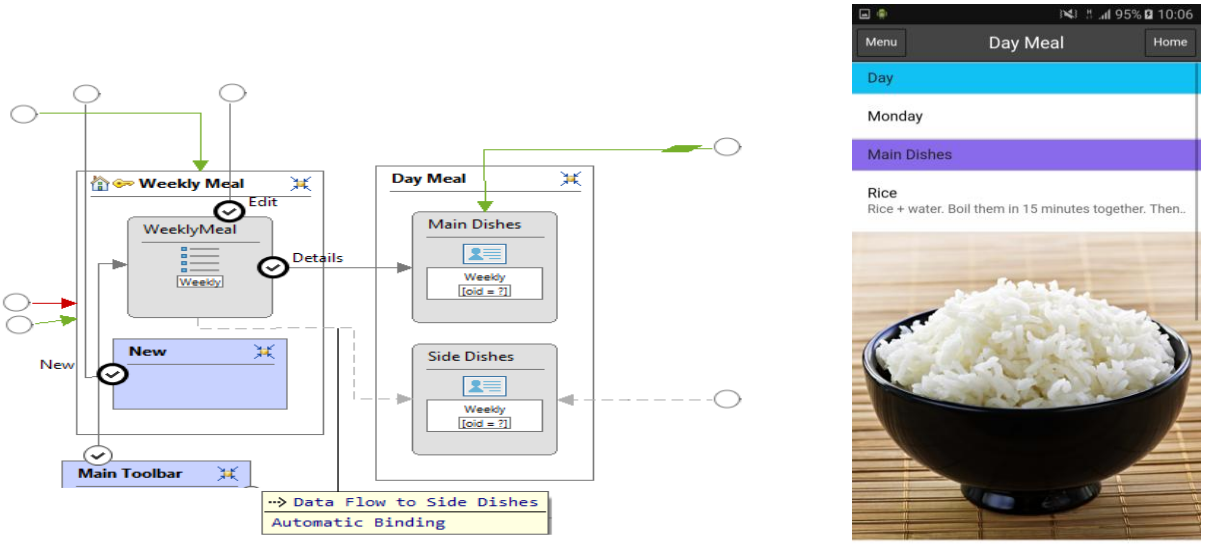
3.2.4.3. Mô hình hóa luồng tương tác ca sử dụng View Meal in Week



Hình 3.7: Mô hình hóa luồng tương tác ca sử dụng View Meal in Week

Ca sử dụng View Meal in Week và giao diện được thể hiện chi tiết trong Hình 3.7, danh sách các món ăn theo ngày trong tuần được thể hiện dưới một List View. Các sự kiện Edit cho phép thay đổi thông tin món ăn và Details cho phép xem chi tiết được kích hoạt bởi tương tác người dùng.

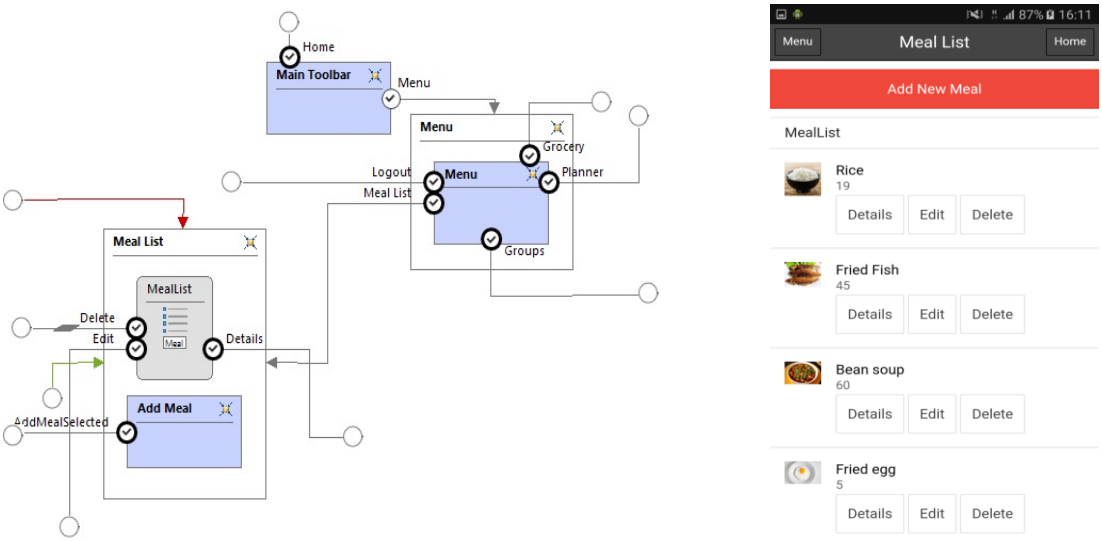
3.2.4.4. Mô hình hóa luồng tương tác ca sử dụng View Meal in Day



Hình 3.8: Mô hình hóa luồng tương tác ca sử dụng View Meal in Day

Người dùng kích hoạt sự kiện View Meal in Day bằng tương tác chạm nút Details. Luồng điều hướng giúp chuyển hướng sang màn hình chi tiết món ăn trong ngày sử dụng tham số ràng buộc mặc định khóa chính (oid). Màn hình Day Meal được thể hiện bởi hai View Component Main Dishes và Side Dishes.

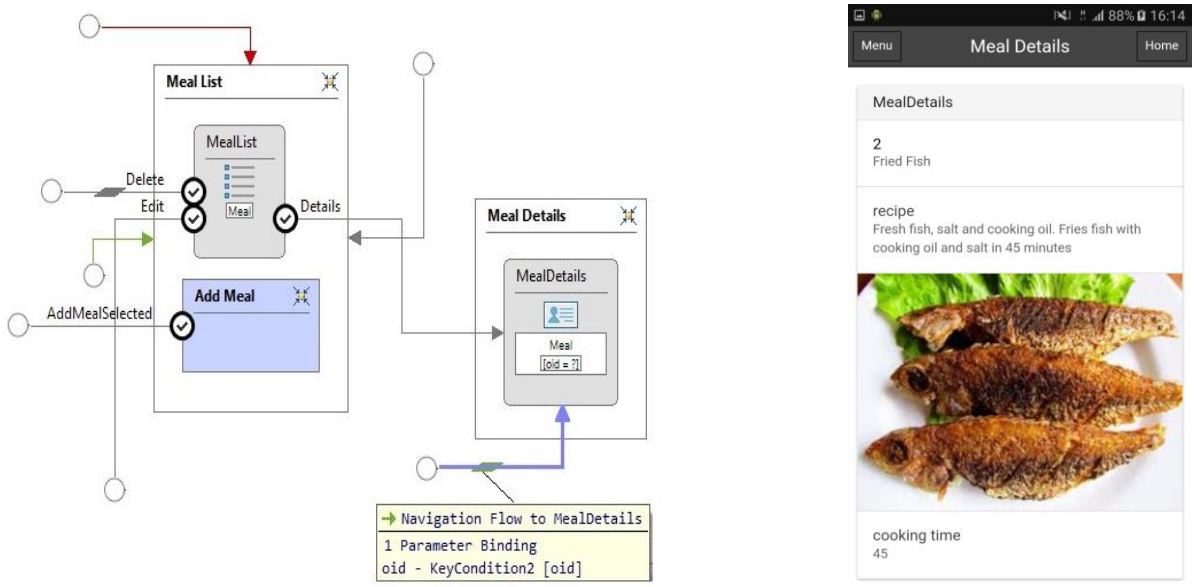
3.2.4.5. Mô hình hóa luồng tương tác ca sử dụng View List Meal



Hình 3.9: Mô hình hóa luồng tương tác ca sử dụng View List Meal

Ca sử dụng View List Meal được thể hiện bởi View Component MealList, ca sử dụng này cho phép kích hoạt các sự kiện : Details, Edit, Delete bằng các tương tác người dùng.

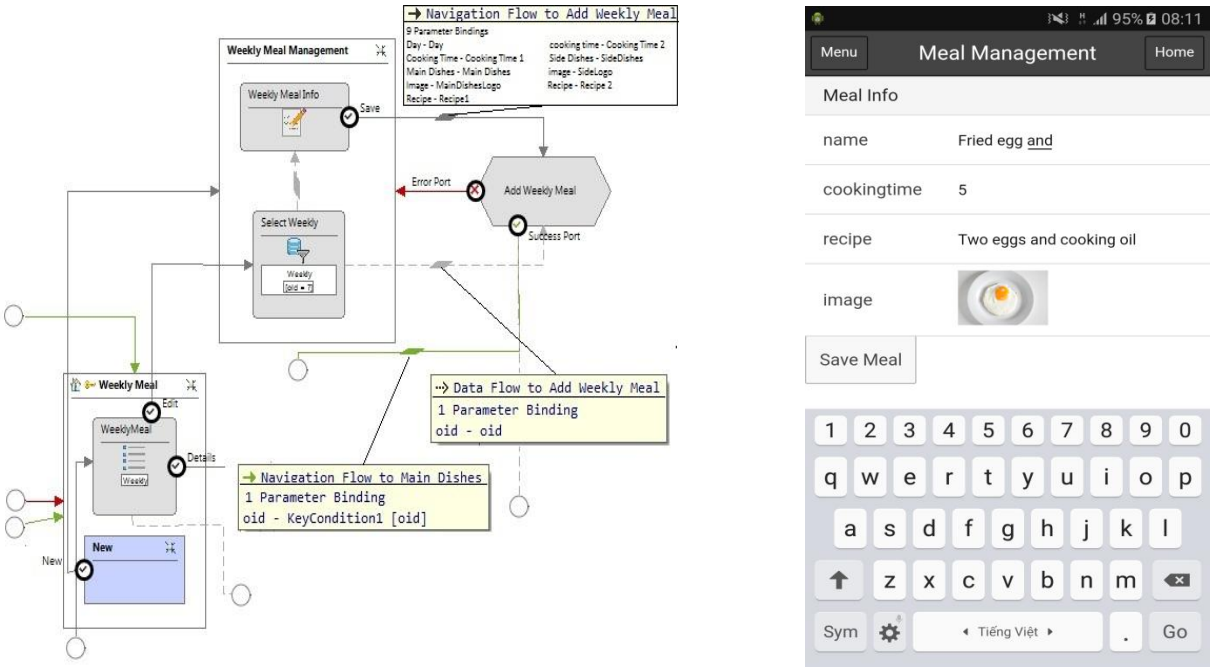
3.2.4.6. Mô hình hóa luồng tương tác ca sử dụng View Meal Details



Hình 3.10: Mô hình hóa luồng tương tác ca sử dụng View Meal Details

Ca sử dụng View Meal Details được thể hiện trên một View Component được kích hoạt bởi tương tác người dùng trên sự kiện Details. Tham số ràng buộc mặc định (oid) được sử dụng trong quá trình điều hướng từ màn hình Meal List.

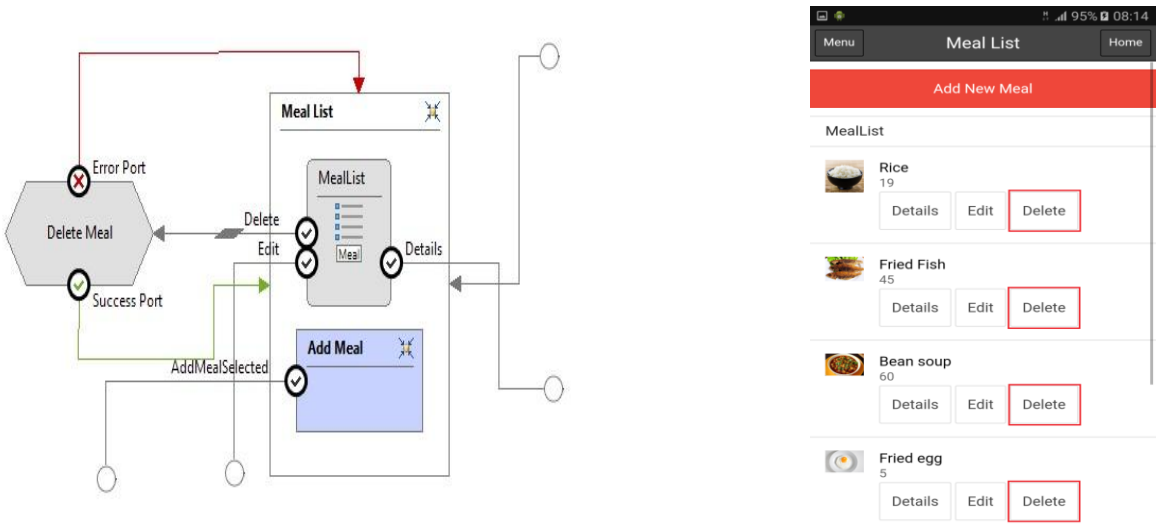
3.2.4.7. Mô hình hóa luồng tương tác ca sử dụng Edit Meal:



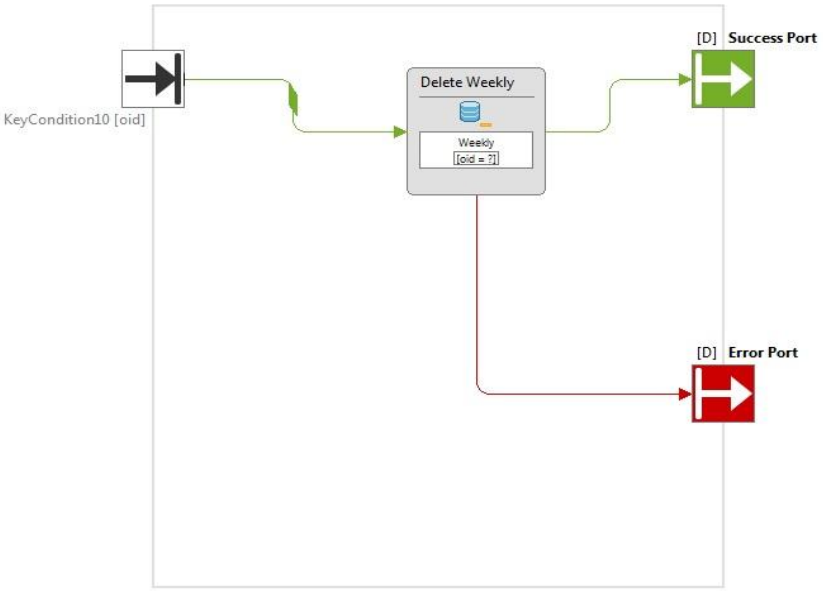
Hình 3.11: Mô hình hóa luồng tương tác ca sử dụng Edit Meal

Ca sử dụng Edit Meal được thể hiện trên màn hình Meal Management, bao gồm View Component cho phép thể hiện thông tin món ăn, Selector Select Weekly nhằm sử lý sự kiện cập nhật/thêm mới thông tin món ăn vào cơ sở dữ liệu. Các thông tin món ăn được người dùng xác nhận bằng cách tương tác nhằm kích hoạt sự kiện Save Meal.

3.2.4.8. Mô hình hóa luồng tương tác ca sử dụng Delete Meal



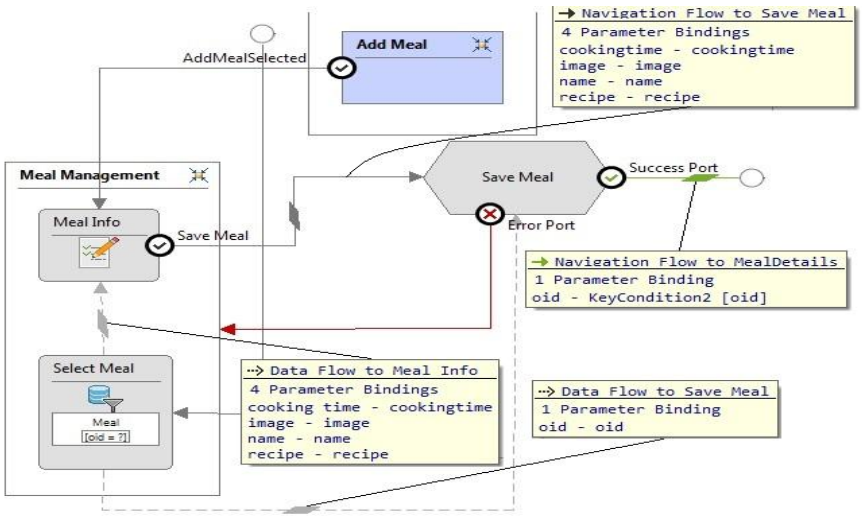
Hình 3.12: Mô hình hóa luồng tương tác ca sử dụng Delete Meal



Hình 3.13: Định nghĩa hành động Delete

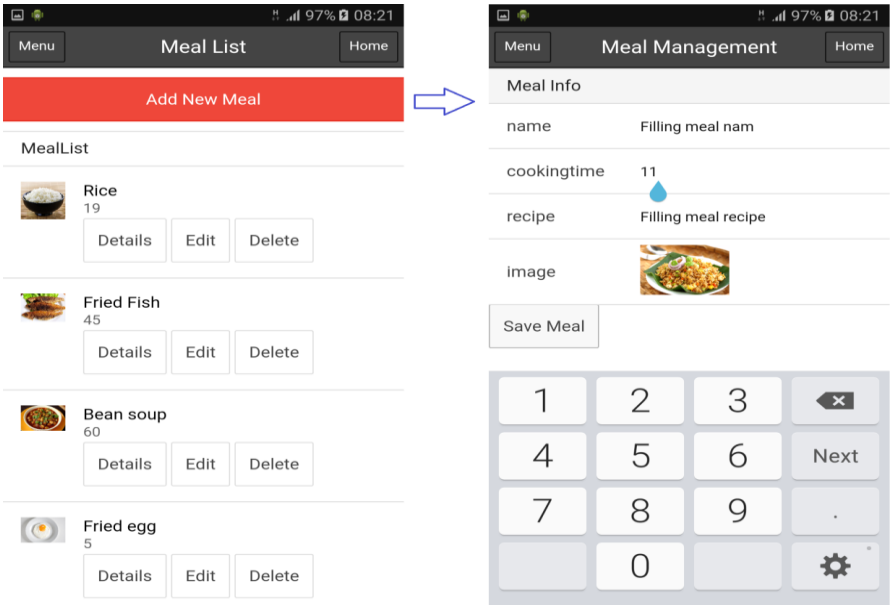
Ca sử dụng Delete Meal là một sự kiện được kích hoạt bởi người dùng trên màn hình Meal List (sự kiện Delete). Sự kiện Delete kích hoạt hành động Delete Meal được định nghĩa như Hình 3.13. Tại đây, tham số ràng buộc là oid. Hành động Delete trả về Success Port nếu xóa thành công thông tin món ăn và trả về Error Port nếu xóa thất bại.

3.2.4.9. Mô hình hóa luồng tương tác ca sử dụng Add New Meal

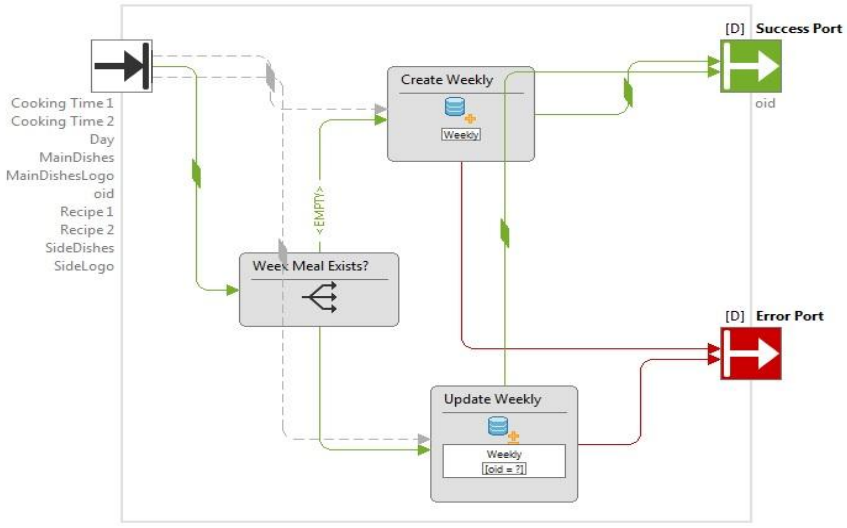


Hình 3.14: Mô hình hóa luồng tương tác ca sử dụng Add New Meal

Ca sử dụng Add New Meal (gọi từ màn hình Meal List) được thể hiện nhằm tái sử dụng màn hình Meal Management dưới dạng một nút có thể được kích hoạt bởi tương tác người dùng. Màn hình Meal Management được tùy chỉnh cho phép người dùng điền mới thông tin về món ăn. Kích hoạt sự kiện Save Meal sau khi hoàn tất khai báo thông tin. Giao diện ca sử dụng này được thể hiện trong Hình 3.15.



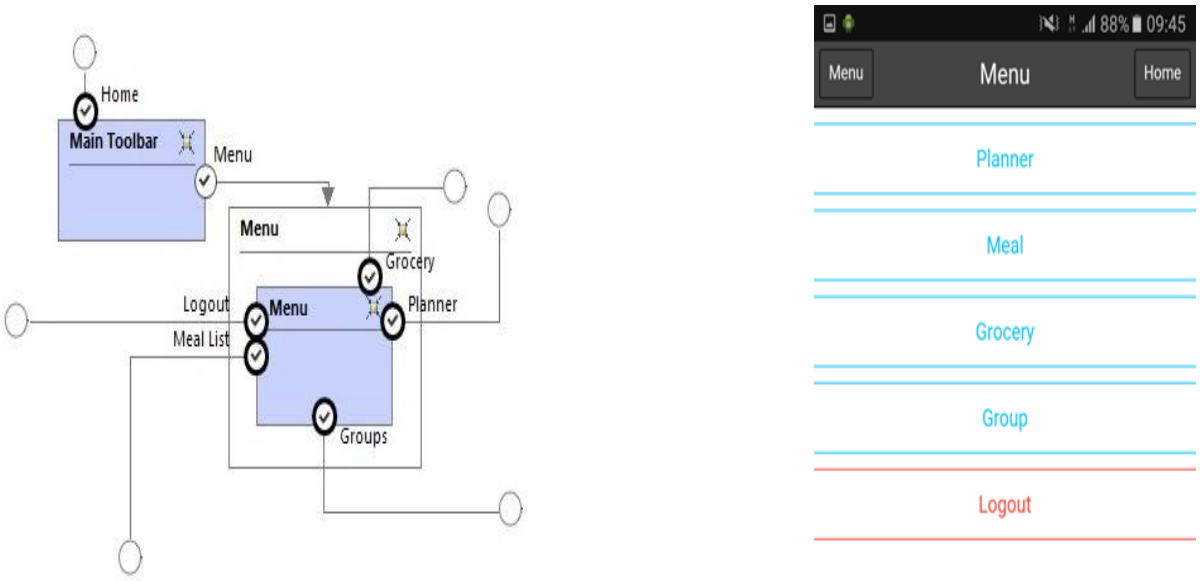
Hình 3.15: Giao diện ca sử dụng Add New Meal



Hình 3.16: Định nghĩa hành động Add New Meal

Hành động Save Meal được sử dụng cho màn hình Add New Meal bao gồm đầu vào là toàn bộ tham số ràng buộc liên quan đến món ăn được khai báo trong lớp Weekly. Hành động này bắt đầu bằng quá trình kiểm tra sự tồn tại của món ăn (Week Meal Exists?), quá trình thêm mới (Create Weekly) được gọi nếu món ăn chưa tồn tại và quá trình cập nhật thông tin (Update Weekly) được gọi nếu món ăn đã tồn tại. Hành động trả về Success Port nếu thành công và Error Port nếu thất bại.

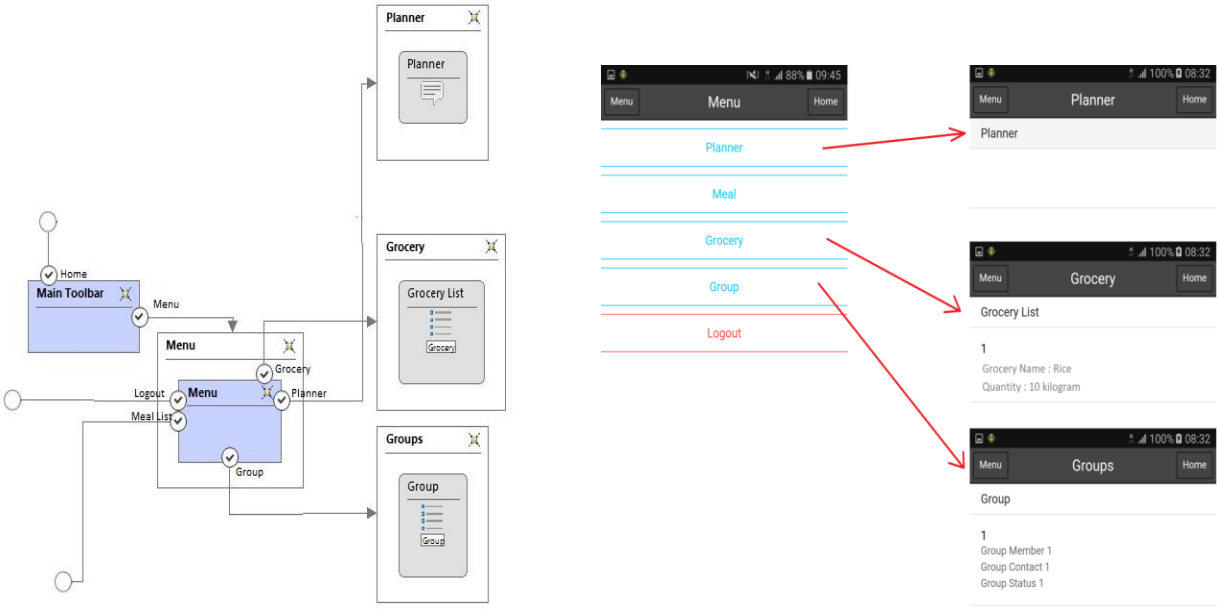
3.2.4.10. Mô hình hóa luồng tương tác ca sử dụng View Menu



Hình 3.17: Mô hình hóa luồng tương tác ca sử dụng View Menu

Ca sử dụng View Menu được gọi từ tất cả các màn hình trên thanh công cụ (Main Toolbar), Màn hình Menu bao gồm các sự kiện có thể kích hoạt bởi tương tác người dùng như Planner, Meal, Grocery, Group, Logout.

3.2.4.11. Mô hình hóa luồng tương tác ca sử dụng View Planner, View Grocery, View Group



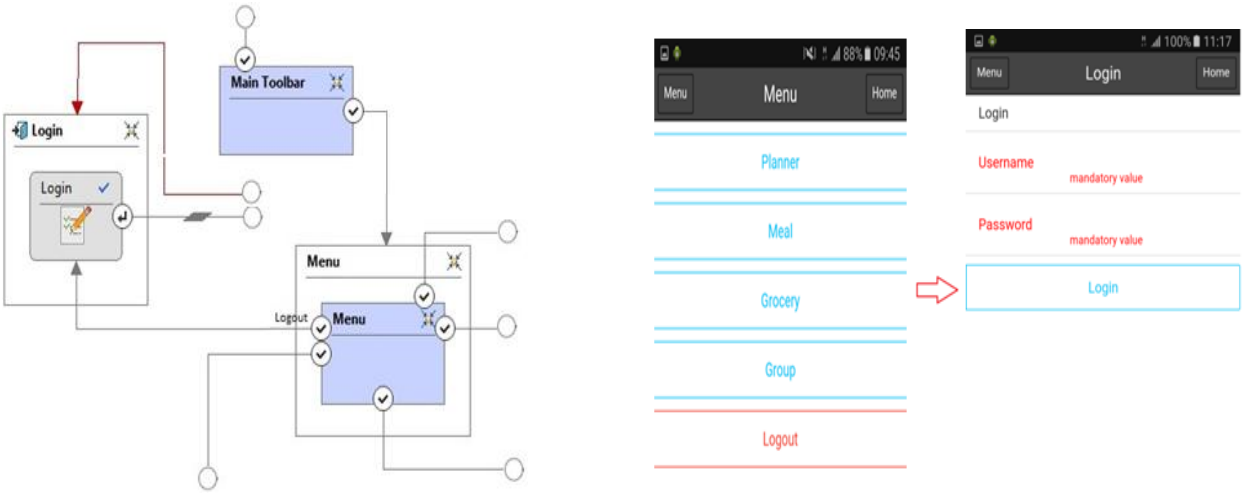
Hình 3.18: Mô hình hóa luồng tương tác ca sử dụng View Planner, View Grocery, View Group

Các màn hình tương ứng với sự kiện được kich hoạt bởi tương tác người dùng. Màn hình này xuất hiện sử dụng luồng điều hướng (Navigation Flow) từ màn hình Menu.

3.2.4.12. Mô hình hóa luồng tương tác ca sử dụng Logout

Ca sử dụng logout được gọi đến từ màn hình Menu, ca sử dụng này sẽ thực hiện đăng xuất khỏi tài khoản của người sử dụng và dùng luồng điều hướng tới màn hình Login cho ca đăng nhập tiếp theo.

Mô hình luồng tương tác ca sử dụng Logout và giao diện trên ứng dụng được thể hiện chi tiết trong Hình 3.19.



Hình 3.19: Mô hình hóa luồng tương tác ca sử dụng Logout

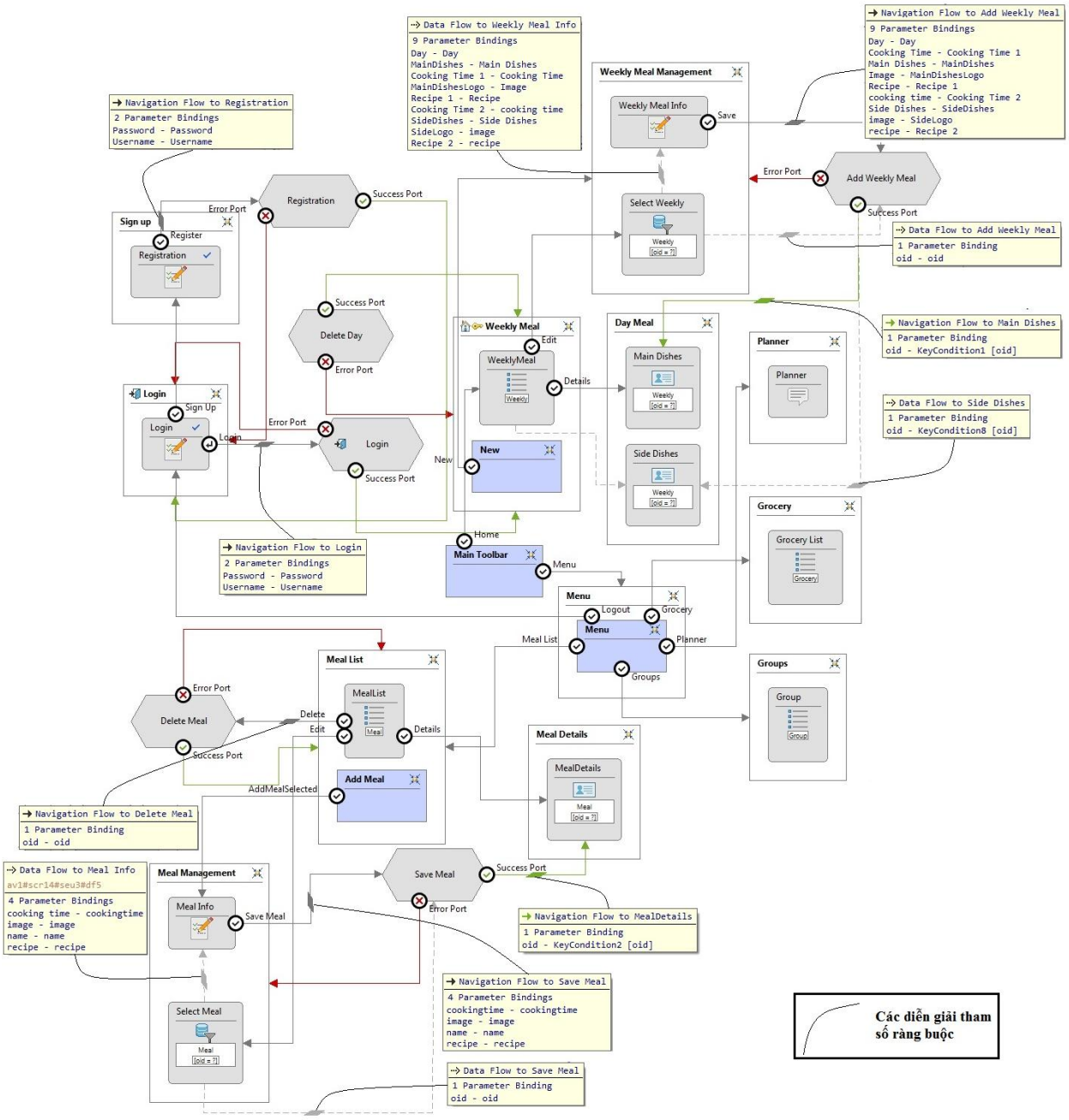
WMP hỗ trợ xây dựng ứng dụng với IFML bao gồm các ràng buộc cụ thể về thông tin được nhập bởi người dùng. Như giao diện Hình 3.19, các trường Username và Password là các trường bắt buộc phải điền để có thể chuyển đến ca sử dụng tiếp theo (mandatory value).

3.2.4.13. Nhận xét chung

Ứng dụng MealNote đã vận dụng toàn bộ các khái niệm chính của IFML và các tiện ích mở rộng của WebRatio dành cho phát triển ứng dụng trên nền tảng di động. Để có thể sử dụng IFML xây dựng ứng dụng di động trên công cụ Webratio Mobile Platform, ta cần hiểu và vận dụng thành thạo ngôn ngữ IFML. Ngoài ra, công cụ được xây dựng dựa trên nền tảng Eclipse nên những nhà phát triển có kinh nghiệm mô hình hóa trên EMF sẽ có bắt đầu thuận lợi hơn.

Quá trình xây dựng ứng dụng ít gặp phải lỗi, tuy nhiên các mẫu thiết kế/giao diện cung cấp sẵn tương đối ít, khả năng tùy chỉnh thấp. Nội dung luận văn sẽ trình bày phân tích chi tiết hơn trong phần tiếp theo.

Hình 3.20 thể hiện toàn bộ mô hình luồng tương tác của ứng dụng MealNote



Hình 3.20: Mô hình luồng tương tác của ứng dụng MealNote

3.3. Kết quả thực nghiệm và đánh giá

Qua quá trình thực nghiệm, tác giả đã xây dựng thành công ứng dụng MealNote sử dụng kỹ thuật IFML (được thể hiện trong Hình 3.20). Ứng dụng này được xây dựng và thử nghiệm trên hệ điều hành di động Android, tuy nhiên nó có thể được sử dụng cho cả nền tảng iOS mà không cần chỉnh sửa thêm.

Phát triển một ứng dụng di động trên hai nền tảng Android gốc và IFML đều có những khó khăn/thuận lợi riêng. Ở đây, tác giả sẽ đánh giá dựa trên các tiêu chí chính và quan trọng nhất dựa theo quan điểm người dùng và nhà phát triển. Với mục đích trình bày tối đa các lĩnh vực quan trọng trong quá trình phát triển ứng dụng di động. Các tiêu chí đã được đề xuất như sau :

- Khả năng xác định yêu cầu và tính khả thi của ứng dụng.
- Chi phí phát triển.
- Thiết kế và giao diện.
- Hỗ trợ tính năng phần cứng và hệ điều hành.
- Hiệu suất và trải nghiệm người dùng.
- Thời gian phát triển ứng dụng.
- Khả năng bảo trì, nâng cấp và bảo mật ứng dụng.
- Các tiêu chí khác : Ứng dụng và thư viện hỗ trợ, công cụ và sửa lỗi, sự độc lập về nền tảng.

3.3.1. Khả năng xác định yêu cầu và tính khả thi của ứng dụng

Ứng dụng gốc luôn là một tiêu chuẩn của các ngôn ngữ, công cụ ngoại lai hướng tới. Vì vậy, các yêu cầu của khách hàng về sản phẩm thường dựa theo các tính năng có thể thực hiện trên ngôn ngữ gốc. Nói cách khác, ngôn ngữ, công cụ cho phép phát triển ứng dụng gốc luôn hỗ trợ tối đa các tính năng của hệ điều hành. Điều này dẫn tới việc ta luôn cần xem xét kỹ khả năng phát triển ứng dụng đối với yêu cầu của khách hàng trước khi đưa ra quyết định có thể thực hiện nó hay không.

Với IFML và WMP, nhà phát triển không thể thực hiện tất cả các loại ứng dụng như ứng dụng Android gốc có thể làm. Bảng 3.3 là danh sách các loại ứng dụng hiện nay và khả năng hỗ trợ của IFML cho việc phát triển đó, so sánh với khả năng hỗ trợ của ứng dụng Android gốc.

Bảng 3.3: So sánh tính khả thi của kiểu ứng dụng giữa IFML và ứng dụng gốc

Loại ứng dụng	IFML	Native
Game		√
Bản đồ / GPS	√	√
Ứng dụng yêu cầu truy cập thiết bị phần cứng		√
Ứng dụng yêu cầu truy cập dữ liệu điện thoại	√	√
Ứng dụng yêu cầu truy cập danh bạ	√	√
Ứng dụng yêu cầu truy cập camera / album ảnh	√	√
Ứng dụng với hệ thống phức tạp, hiệu suất cao		√

Ứng dụng yêu cầu khả năng đồ họa		√
Ứng dụng đơn giản với view và dữ liệu	√	√
Ứng dụng yêu cầu truy cập tính năng lịch gốc	√	√
Ứng dụng sử dụng QR code - Barcode	Tốt, tích hợp sẵn	Tự xây dựng
Ứng dụng với các Dịch vụ dữ liệu (Data Service)	√	√
Ứng dụng yêu cầu bản địa hóa ngôn ngữ	√	√
Ứng dụng yêu cầu xử lý thuật toán phức tạp		√

Qua Bảng 3.3 có thể thấy IFML chỉ thích hợp với một số loại ứng dụng có độ phức tạp và hiệu suất trung bình, tuy nhiên một số tính năng gốc như Máy ảnh, Danh bạ, Lịch... được hỗ trợ rất tốt không thua kém ứng dụng gốc. Tuy nhiên, về tiêu chí này, IFML còn thua kém ứng dụng gốc do những yếu tố khách quan.

3.3.2 Chi phí phát triển

Thông thường, khi nhận được yêu cầu từ khách hàng, việc đầu tiên nhà phát triển cần cung cấp tới khách hàng là ước tính chi phí phát triển, đây là yếu tố quan trọng đối với khách hàng. Việc ước tính chi phí phát triển tuy không thể chính xác trong giai đoạn đầu của dự án, nhưng đó là cơ sở để đàm phán với khách hàng và quyết định sự sống còn của nhà phát triển.

Tiêu chí này được đánh giá trên các nghiên cứu tham khảo [3, 18, 17] và được xác nhận trong quá trình phát triển ứng dụng MealNote cho nền tảng Android sử dụng hai phương pháp: Xây dựng ứng dụng gốc và xây dựng ứng dụng sử dụng kỹ thuật mô hình hóa luồng tương tác IFML.

Sử dụng mô hình ước lượng Costructive Cost Model (COCOMO) theo kích cỡ phần mềm:

- Nỗ lực $E = a * L^b$
- Thời gian $T = c * E^d$
- Số người $N = E/T$

L: Số dòng lệnh (KLOC)
a, b, c, d: tham số theo Bảng 3.4

Bảng 3.4: Bảng giá trị tham số cho phương pháp COCOMO

	a	b	c	d
organic	3.2	1.05	2.5	0.38
semi-detached	3.0	1.12	2.5	0.35
embeded	2.8	1.2	2.5	0.32

- organic: là kiểu dự án đơn giản, không truy cập các thiết bị ngoại lai.

- semi-detached: Các dự án phức tạp, kinh nghiệm của các thành viên đến lĩnh vực liên quan là hạn chế.

- embeded: Các dự án phức tạp, yêu cầu truy cập thiết bị ngoại vi.

Theo bảng trên, dự án MealNote là dự án thuộc loại organic, từ đó ta có được tham số a = 3.2, b = 1.05, c = 2.5, d = 0.38.

Dựa theo phương pháp xây dựng ứng dụng Android gốc sử dụng Android Studio, ước tính để xây dựng tốt các chức năng của ứng dụng bao gồm cả thiết kế giao diện sẽ tương đương khoảng 1300 dòng lệnh ~ 1.3KLOC

Nỗ lực $E = a * L^b = 3.2 * 1.3^{1.05} = 4.215$ Man - month
Thời gian $T = c * E^d = 2.5 * 4.215^{0.38} = 4.32$ tháng
Số người N = E/T = 4.215 / 4.32 ~ 1 người

Vậy ta có thể thấy, ước tính công sức cho dự án MealNote sử dụng phương pháp xây dựng ứng dụng Android gốc tương đương khoảng 4 man - month.

Tuy nhiên, khi xây dựng ứng dụng này sử dụng kỹ thuật mô hình hóa luồng tương tác IFML, công sức phát triển thực tế đã sử dụng vào khoảng 21 man - day, tương đương ~1 man - month.

Bảng 3.5 thể hiện công sức phát triển thực tế dựa theo tính năng của phương pháp IFML.

Bảng 3.5: Công sức phát triển ứng dụng MealNote sử dụng IFML theo chức năng

Chức năng	Công sức phát triển (man-day)
Login/signup	1
View Weekly Meal	3.5
View Meal Details	1
Edit Meal	0.5
View List of Meal	1.5

View Menu	0.5
Logout	0.3
Add new Meal	2
Database connection	1.5
View Meal of Day	1.5
Delete Meal	1.5
Other View	3
Layout Design	3.5
Tổng	21.3

Điều này có thể giải thích một cách cơ bản do phát triển ứng dụng di động với IFML là phương pháp phát triển phần mềm hướng mô hình. Ngoài ra, IFML sử dụng PhoneGap như một bước chuyển trung gian để xây dựng ứng dụng di động, điều này có nghĩa là ứng dụng xây dựng bởi IFML là một dạng ứng dụng lai giữa nền tảng Web và nền tảng di động gốc (hybrid application). IFML sẽ kế thừa được toàn bộ những ưu điểm của phương pháp MDD bao gồm cả ưu điểm về chi phí phát triển thấp.

Có thể thấy lợi ích về chi phí phát triển ứng dụng sử dụng phương pháp IFML thấp hơn phương pháp xây dựng ứng dụng gốc tương đối nhiều. Đây cũng chính là một trong những điểm mạnh nhất của IFML.

3.3.3. Thiết kế và giao diện

So sánh với ứng dụng gốc, giao diện của ứng dụng xây dựng bằng IFML chỉ ở mức đơn giản. IFML - WMP không hỗ trợ về giao diện tốt như ứng dụng gốc, số tùy chỉnh màu sắc, hình ảnh hay hoạt cảnh bị giới hạn. Hiện tại, IFML-WMP chỉ hỗ trợ một số loại màu cơ bản bao gồm: đen (black), trắng (white), xám (gray), xanh biển (blue), xanh sáng (light blue), xanh lá (green), vàng (yellow), đỏ (red), tím (violet). Trong khi đó, số màu được hỗ trợ ở ứng dụng gốc là không giới hạn, có thể thay đổi bằng cách chọn mã màu tương ứng dạng số nguyên hoặc số hệ 16 (hexa).

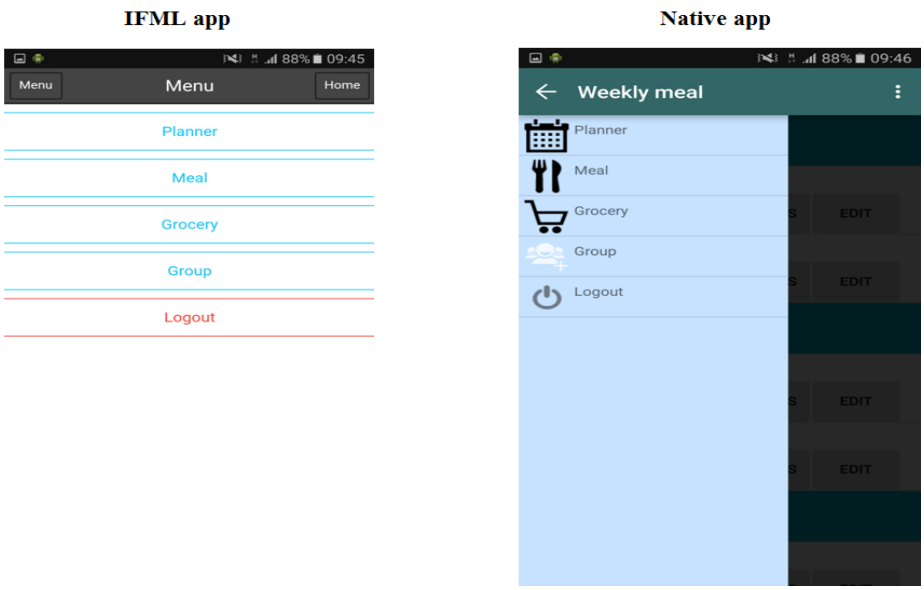
Việc lựa chọn màu sắc, hình ảnh với IFML-WMP rất đơn giản và nhanh chóng. Hình ảnh có thể chọn trực tiếp từ bộ nhớ thiết bị mà không yêu cầu qua bất kỳ công đoạn chỉnh sửa nào, tuy nhiên khi chọn màu sắc, giao diện không thể hiện trực quan cho đến khi chạy ứng dụng. Trong khi đó, với ứng dụng gốc, ta cần phải chọn định dạng và kích cỡ ảnh nhất định để có thể sử dụng chúng cho ứng dụng.

Ngoài ra, các hình ảnh cần phải được tích hợp vào thư mục nhất định trong thư mục của dự án, ở đây là - thư mục app\resource\drawable.

Hoạt cảnh (amimation) khi chuyển View là tối thiểu khi chỉ được hỗ trợ một kiểu trượt duy nhất. Với ứng dụng gốc có thể hỗ trợ: trượt, xoay, cuộn, lật...hoặc tùy chỉnh bởi lập trình viên.

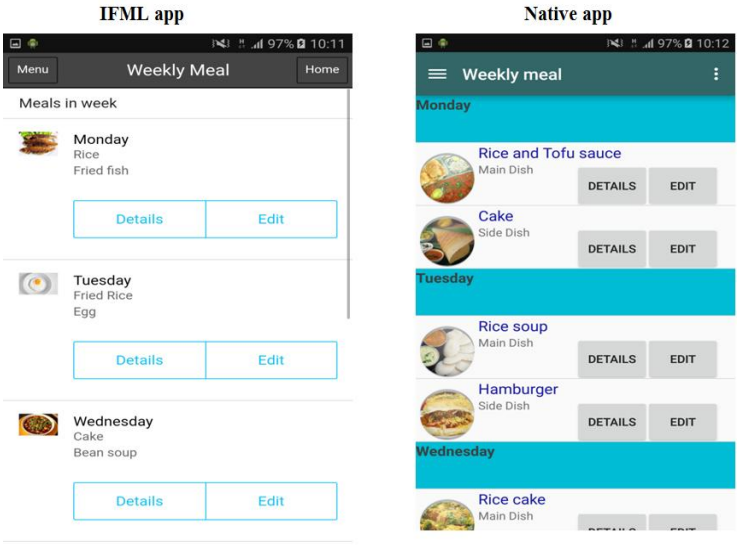
Khó khăn trong việc tùy chỉnh kích thước, vị trí của các thành phần View bao gồm Button, Text View, List View... Ví dụ về kích thước của Button bị giới hạn trong các kiểu như: Small, Normal, Large, Block, Full. Tùy chọn về kiểu hiển thị là đầy đủ (icon & text) so sánh với ứng dụng gốc.

Về thiết kế giao diện, IFML-WMP luôn hỗ trợ kiểu linear layout, là kiểu thiết kế cho phép phù hợp với mọi thiết bị có kích cỡ màn hình/độ phân giải khác nhau. Hình 3.21 là một số hình ảnh so sánh trực quan giữa màn hình ứng dụng được xây dựng bởi phương pháp IFML và ứng dụng gốc:



Hình 3.21: So sánh thiết kế giao diện chức năng Menu

Qua hình so sánh trực tiếp trên, có thể thấy khả năng tùy chỉnh giao diện của ứng dụng gốc là vượt trội so với ứng dụng sử dụng IFML. Tiếp theo là hình ảnh so sánh của màn hình chức năng View Weekly Meal – Hình 3.22



Hình 3.22: So sánh thiết kế giao diện chức năng View Weekly Meal

Trong màn hình này, tác giả đã xây dựng MealNote_IFML sao cho giống ứng dụng gốc nhất, tuy nhiên do hạn chế về khả năng tùy chỉnh, giao diện của MealNote_IFML chỉ có thể dừng lại ở mức như trên.

Khả năng tùy chỉnh về giao diện không phải là điểm mạnh của IFML-WMP so sánh với ứng dụng gốc khi sự hỗ trợ về giao diện còn hạn chế (màu sắc, hoạt cảnh...). Việc thiết kế giao diện cũng gặp nhiều khó khăn hơn so với kỹ thuật gốc do sự hạn chế về hoạt cảnh, kích cỡ và loại thành phần hiển thị.

Tuy nhiên, IFML cũng hỗ trợ các tính nhằm rút ngắn thời gian thiết kế giao diện người dùng, vốn là một giai đoạn tốn kém rất nhiều thời gian và chi phí như: trực quan trong việc tùy chỉnh màu sắc, luôn hỗ trợ linear layout cho phép hoạt động tốt với các loại thiết bị có kích cỡ khác nhau.

3.3.4. Khả năng hỗ trợ tính năng phần cứng và hệ điều hành

Như ta đã biết, Android được phát triển và công bố bởi Google, Android Studio là công cụ lập trình ứng dụng Android gốc chính thức của Google. Tương tự với hệ điều hành iOS và công cụ lập trình ứng dụng iOS gốc X-Code. Điều này đồng nghĩa với việc mỗi khi có phiên bản hệ điều hành mới, hay công nghệ mới cho thiết bị phần cứng, các công cụ phát triển ứng dụng gốc sẽ được nâng cấp ngay lập tức, thậm chí còn sớm hơn rất nhiều thời điểm ra mắt hệ điều hành/công nghệ mới đó. Đây là một lợi thế không thể phủ nhận của công cụ phát triển ứng dụng gốc. Điều này có nghĩa là các công cụ được phát triển bởi bên thứ 3 như PhoneGap hay WebRatio luôn chậm chân trong việc cập nhật hỗ trợ tính năng/hệ điều hành mới.

Dựa trên cơ sở hệ điều hành và tính năng thiết bị hiện tại, Bảng 3.6 là bảng các tính năng được hỗ trợ trên IFML-WMP:

Bảng 3.6: So sánh tính năng gốc và khả năng hỗ trợ giữa IFML và ứng dụng gốc

Tính năng	IFML	Native
Camera	√	√
Danh bạ	√	√
NFC		√
Dữ liệu cảm biến gia tốc		√
Kết nối mạng	√	√
Bar code / QR code	√	√
Lịch	√	√
Bộ nhớ	√	√
Xoay màn hình	√	√
Thông báo	√	√
Bản đồ / GPS	√	√
La bàn		√
Truy cập file trên thiết bị	√	√
Bản địa hóa	√	√

IFML hỗ trợ phần lớn các tính năng hiện tại của thiết bị, tuy chưa thể so sánh tương xứng với ứng dụng gốc. Dựa trên cơ sở những loại ứng dụng có thể phát triển với phương pháp IFML với WMP thì sự hỗ trợ như trên là tương đối đầy đủ.

3.3.5. Hiệu suất ứng dụng và trải nghiệm người dùng

3.3.5.1 Hiệu suất ứng dụng

Tất cả các bước so sánh trong phần này được thực nghiệm trên thiết bị Galaxy A7 - 2016 có mã sản phẩm A710F - 2016. Dưới đây là một số thông số kỹ thuật của thiết bị:

- Hệ điều hành: Android Marshmallow 6.0.1

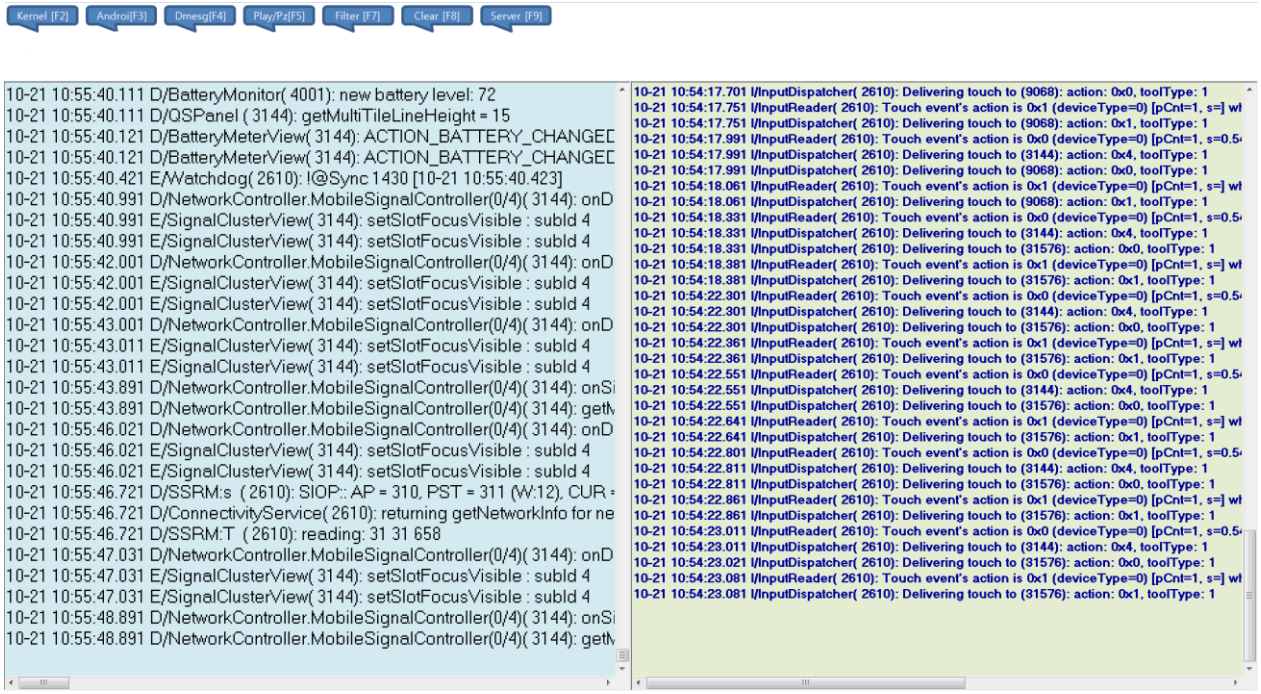
- Màn hình: 5.5 inches, độ phân giải 1080x1920

- Bộ nhớ trong: 3GB RAM

- Dung lượng pin: 3300 mAh

- Vi xử lý: Octa-core (4x1.2 GHz Cortex-A53 & 4x1.5 GHz Cortex-A53) Octa-core 1.6 GHz Cortex-A53

So sánh trực tiếp giữa hai ứng dụng MealNote_IFML và MealNote_Android bằng các sử dụng các công cụ kiểm tra thông tin gỡ lỗi (Debug messages). Ở đây sử dụng công cụ cho phép kiểm tra các Debug messages thời gian thực. Cơ chế hoạt động của công cụ này là truy cập và lấy thông tin trực tiếp từ bộ nhớ RAM và in ra màn hình công cụ ngay lập tức, đảm bảo tính trực quan và chính xác khi giúp người sử dụng có thể kiểm tra tất cả các thông điệp bao gồm thông điệp ứng dụng và thông điệp của hệ điều hành Android. Hình 3.23 thể hiện giao diện của công cụ này:



Hình 3.23: Màn hình công cụ kiểm tra thông điệp ứng dụng thời gian thực

Thực hiện đo tốc độ khởi động ứng dụng, Android_IFML khởi động chậm hơn ứng dụng gốc tương đối nhiều. Với ứng dụng gốc chỉ cần chưa đến một giây để khởi động, tuy nhiên ứng dụng IFML cần đến khoảng 6 giây. Thời gian trễ này nằm trong giới hạn kỹ thuật cho việc đồng bộ dữ liệu của IFML được xác nhận bởi Webratio [21]. Các thông điệp được kiểm tra trong quá trình khởi động ứng dụng:

- Ứng dụng gốc

Thực hiện khởi động ứng dụng

```
10-21 11:11:35.661 I/InputReader( 2610): Touch event's
action is 0x0 (deviceType=0) [pCnt=1, s=0.5468 ]
when=43875125855000
```

Ứng dụng hoàn tất khởi động

```
10-21 11:11:36.471 V/MealNote App( 2592): booting finished
```

Cần chính xác 0.81 giây để khởi động ứng dụng gốc MealNote_Android.

- Kiểm tra việc khởi động ứng dụng với kỹ thuật IFML:

Thực hiện khởi động ứng dụng

```
10-21 12:35:41.481 I/InputReader( 2610): Touch event's
action is 0x0 (deviceType=0) [pCnt=1, s=0.5500 ]
when=48920940355000
```

Ứng dụng hoàn tất khởi động

```
10-21 12:35:47.941 I/chromium( 6835): [INFO:CONSOLE(117)]
"12:35:47.749 DEBUG [wrm.core.PanelService] [scr1] Updated
view of ListService:pwu1 -
```

Cần tới 6.46 giây để hoàn tất quá trình khởi động ứng dụng MealNote_IFML.

Thực hiện đo tốc độ truy xuất dữ liệu trên cả 2 ứng dụng ta có kết quả hiệu suất truy cập cơ sở dữ liệu của MealNote_IFML tương đối kém so với ứng dụng gốc. MealNote_IFML cần tới 0.241 giây để truy cập cơ sở dữ liệu trong khi đó với ứng dụng gốc chỉ cần 0.1 giây. Ngoài ra, theo bản thông tin về giới hạn kỹ thuật của WebRatio [21]. Các ứng dụng di động được xây dựng bằng IFML cần tới 30 giây để đồng bộ dữ liệu 3000 đối tượng và đây là giới hạn của ứng dụng xây dựng bằng kỹ thuật mô hình hóa luồng tương tác. Ứng dụng sẽ không thể hoạt động bình thường với bộ dữ liệu lớn hơn 3000 đối tượng.

- Ứng dụng gốc

Thời gian bắt đầu truy xuất dữ liệu:

```
10-17 13:55:11.481 I/InputReader( 2589): Touch event's
action is 0x0 (deviceType=0) [pCnt=1, s=0.7248 ]
when=97311522328000
```

Truy xuất dữ liệu thành công:

```
10-17 13:55:11.721 I/chromium(19258):    "data": {
....
10-17 13:55:11.721 I/chromium(19258):    }
```

Cần 0.241 giây để truy xuất dữ liệu

- Ứng dụng MealNote_IFML

Thời gian bắt đầu truy xuất dữ liệu

```
10-17 14:19:57.461 V/myApp   (20521): View Meal Details
started
10-17 14:19:57.461 I/Timeline(20521): Timeline:
Activity_launch_request id:com.example.kuldeep.project
time:98797507
```

Thời gian kết thúc truy xuất dữ liệu

```
10-17 14:19:57.561 V/myApp   (20521): finished load meal
details
10-17 14:19:57.571 D/ActivityManager( 2589): post active
user change for 0 fullscreen true isFloatingActivity() false
isHomeActivity() false
```

Cần 0.1 giây để truy xuất dữ liệu.

Thực hiện đo tốc độ thực thi với thử kiệm chuyển màn hình trên cả hai ứng dụng ta được kết quả tốc độ thực thi hàm với phép đo thời gian chuyển màn hình mang lại kết quả không tốt cho IFML, ứng dụng xây dựng bằng IFML cần tới 0.24 giây để chuyển giữa hai màn hình trong khi ứng dụng gốc chỉ cần 0.1 giây, điều này làm giảm hiệu suất của ứng dụng tương đối lớn.

- Ứng dụng gốc

Thực hiện chuyển màn hình

```
10-17 17:50:05.251 V/MainView(28253): Start to switch view
```

Kết thúc chuyển màn hình

```
10-17 17:50:05.351 V/myApp   (28253): View finished
```

Cần 0.1 giây để thực hiện chuyển giữa hai màn hình khác nhau.

Ứng dụng MealNote_IFML

Thực hiện chuyển màn hình

```
10-17 17:26:35.881 I/InputReader( 2589): Touch event's action is
0x0 (deviceType=0) [pCnt=1, s=0.8261 ] when=109821387964000
```

Kết thúc chuyển màn hình với thông điệp hệ thống từ chromium

```
10-17 17:26:36.121 I/chromium(19258): [INFO:CONSOLE(117)]
"17:26:36.102 DEBUG [wr.mvc.BlobController] Released internal URL
for BLOB 245", source:
file:///android_asset/www/app/lib/angular.js (117)
```

Cần 0.24 giây để chuyển màn hình trên ứng dụng MealNote_IFML.

Đo tốc độ cuộn trên màn hình list view đã được kết nối dữ liệu nhằm làm rõ hơn hiệu suất của ứng dụng. Thao tác cuộn là rất phổ biến cho trải nghiệm người dùng do hầu hết các ứng dụng hiện nay đều sử dụng kiểu hiển thị này. Kiểu hiển thị này cho phép ứng dụng thể hiện nhiều nội dung trên một màn hình duy nhất. Cần 0.16 giây để cuộn một màn hình thể hiện danh sách 20 phần tử món ăn từ cơ sở dữ liệu trên ứng dụng MealNote_IFML.

- Ứng dụng MealNote Android gốc

Thực hiện để cuộn màn hình

```
10-17 17:52:57.431 D/InputReader( 2589): Input event(7): value=1
when=111402936871000
```

Kết thúc cuộn màn hình

```
10-17 17:52:57.511 I/InputDispatcher( 2589): Delivering touch to
(28253): action: 0x1, toolType: 1
```

Cần 0.02 giây để thực hiện cuộn màn hình thể hiện danh sách 20 phần tử món ăn từ cơ sở dữ liệu trên ứng dụng gốc.

- Ứng dụng MealNote_IFML

Thực hiện cuộn màn hình

```
10-17 17:55:17.491 D/InputReader( 2589): Input event(7): value=1
when=111543008176000
```

Kết thúc cuộn màn hình

```
10-17 17:55:17.651 I/InputReader( 2589): Touch event's action is
0x1 (deviceType=0) [pCnt=1, s=] when=111543159911000
10-17 17:55:17.651 I/InputDispatcher( 2589): Delivering touch to
(19258): action: 0x1, toolType: 1
10-17 17:55:17.651 D/ViewRootImpl(19258): ViewPostImeInputStage
processPointer 1
```

Qua các phép đo những tính năng cơ bản ảnh hưởng tới trải nghiệm người dùng nhất ở trên, ta có thể thấy được hiệu suất là điểm yếu lớn của phương pháp lập trình ứng dụng di động sử dụng kỹ thuật IFML. Các phép đo hiệu suất của ứng dụng xây dựng với kỹ thuật IFML đều thua kém ứng dụng gốc. Từ điều này có thể đưa ra một kết luận, những ứng dụng yêu cầu hiệu suất cao không phù hợp để xây dựng với kỹ thuật IFML. Luận văn trình bày các đánh giá về trải nghiệm người dùng ở phần tiếp theo để có thể thấy hiệu suất ứng dụng ảnh hưởng trực tiếp tới người dùng như thế nào.

3.3.5.2 Trải nghiệm người dùng - UX

Với người dùng, tiêu chí đầu tiên để đánh giá một ứng dụng là về hình thức ứng dụng, tiếp theo là tiêu chí quan trọng khác: trải nghiệm người dùng. Trải nghiệm người dùng là một phần không nhỏ lý do khiến người dùng quyết định tiếp tục hay ngừng sử dụng ứng dụng.

So sánh ứng dụng MealNote_IFML được xây dựng bằng kỹ thuật mô hình hóa luồng tương tác và ứng dụng gốc MealNote_Android, việc đầu tiên nhận thấy là ứng dụng sử dụng kỹ thuật IFML cung cấp một trải nghiệm không tốt do đáp ứng chậm với tương tác người dùng.

Thời gian khởi động ứng dụng chậm dẫn đến người dùng phải dành nhiều thời gian hơn để có thể sử dụng ứng dụng, nguyên nhân chính được thể hiện trên phần đo hiệu suất khi MealNote_IFML cần đến khoảng 6 giây để khởi động hoàn tất, trong khi ứng dụng gốc chỉ cần chưa đầy 1 giây.

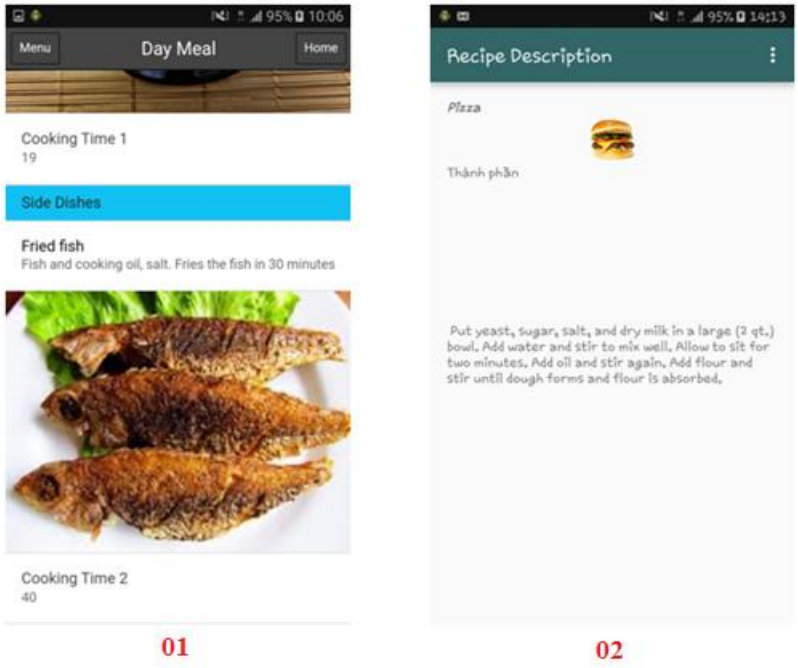
Trải nghiệm người dùng khi chuyển màn hình ứng dụng trên MealNote_IFML tuy không tốt bằng ứng dụng gốc, nhưng do thời gian chuyển không lớn (chỉ khoảng 0.2) giây kèm theo hiệu ứng trượt, điều này dẫn đến người dùng không cảm thấy sự khác biệt giữa hai nền tảng. Có thể coi trải nghiệm người dùng về mặt này tốt tương đương ứng dụng gốc.

Trái lại, trải nghiệm người dùng khi sử dụng thao tác cuộn là không tốt khi so sánh với ứng dụng gốc. Cụ thể với một danh sách 20 phần tử món ăn, ứng dụng gốc cho trải nghiệm mượt mà và chỉ cần một thao tác khi cuộn từ đầu đến cuối danh sách. Trong khi đó, ứng dụng MealNote_IFML đáp ứng nhanh thao tác thao tác ban đầu, nhưng xảy ra hiện tượng giật từ 2 đến 3 lần cho danh sách có độ dài tương tự. Điều này gây ấn tượng xấu tới trải nghiệm người dùng.

Số hiệu ứng khi thao tác với ứng dụng MealNote_IFML khá hạn hẹp. Chỉ hỗ trợ hiệu ứng trượt đơn điệu, trong khi đó ứng dụng gốc hỗ trợ tốt các hiệu ứng khác nhau như cuộn, xoay, trượt...làm phong phú hơn trải nghiệm người dùng.

Ngoài ra, ứng dụng xây dựng bằng kỹ thuật IFML không hỗ trợ tốt đối với tùy chỉnh của người dùng trên hệ điều hành. Trong trường hợp người dùng thay đổi font chữ hệ thống, tất cả các ứng dụng gốc sẽ được thể hiện dưới dạng font chữ mới.

Tuy nhiên ứng dụng MealNote_IFML vẫn sử dụng font chữ ban đầu. Có thể nói, kỹ thuật IFML không cho phép người dùng cá nhân hóa ứng dụng. Một thử nghiệm nhỏ dưới đây sẽ thể hiện rõ điểm khác biệt này. Trong minh họa Hình 3.24, font chữ hệ thống được đổi sang font Choco Cooky, ứng dụng gốc MealNote_Android thể hiện ngay lập tức sự thay đổi, nhưng không có sự thay đổi nào trên ứng dụng MealNote_IFML.



Hình 3.24: So sánh sự thay đổi phông chữ ứng dụng theo phông chữ hệ thống

Như vậy, trải nghiệm người dùng của ứng dụng được xây dựng với kỹ thuật IFML là không tốt khi so sánh với ứng dụng gốc. Tuy nhiên, các trải nghiệm này vẫn đáp ứng được nhu cầu cơ bản của người dùng, đặc biệt là với các ứng dụng không yêu cầu hiệu suất lớn

3.3.5.3. Nhận xét chung

Hiệu suất là điểm yếu của ứng dụng được xây dựng với kỹ thuật IFML, các ứng dụng yêu cầu hiệu suất cao là không thích hợp với kỹ thuật này. IFML tỏ ra thích hợp hơn với các ứng dụng nhẹ về mặt hiệu suất.

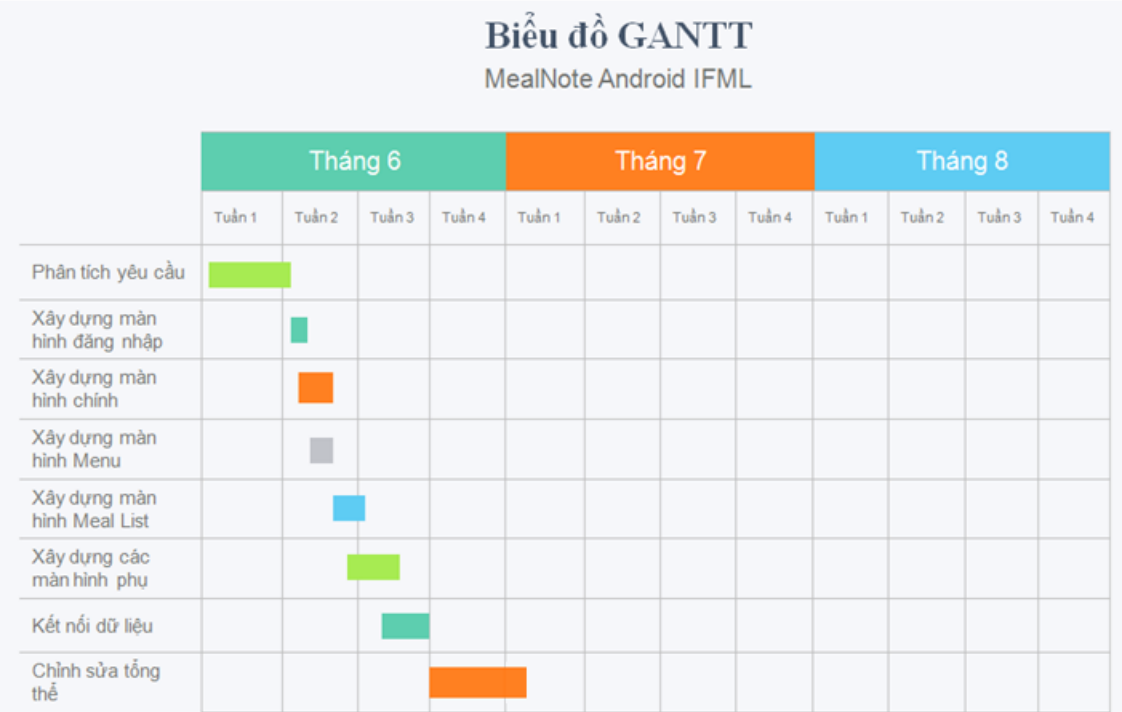
Hiệu suất thấp hơn dẫn đến trải nghiệm người dùng không tốt trên ứng dụng xây dựng bằng kỹ thuật IFML so sánh với ứng dụng gốc. Tuy nhiên, ứng dụng với IFML vẫn đáp ứng các yêu cầu cơ bản về trải nghiệm người dùng.

3.3.6. Thời gian phát triển ứng dụng

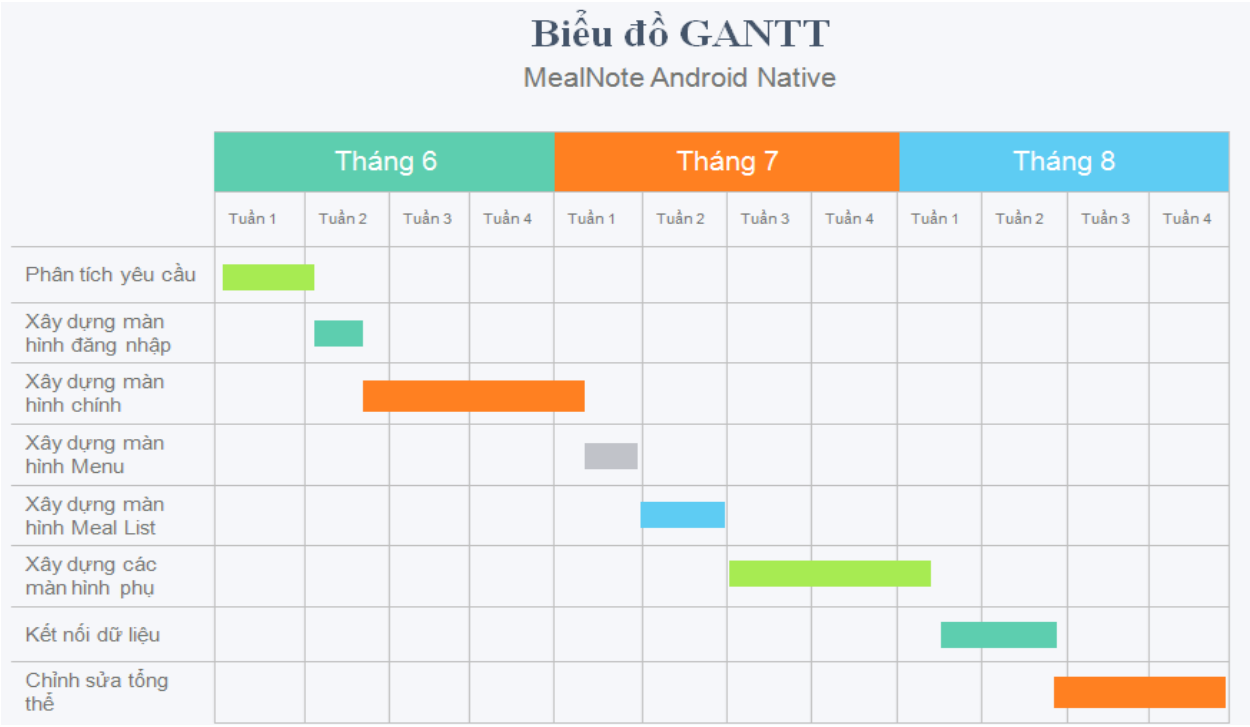
Một ứng dụng được yêu cầu phát triển thường dựa trên những tiêu chuẩn của ứng dụng gốc, do vậy việc xác định yêu cầu và tính khả thi của ứng dụng trong việc phát triển bằng kỹ thuật IFML phải được xem xét cẩn thận. Công đoạn này thường chiếm một phần lớn thời gian của quá trình phát triển ứng dụng, nhưng về tổng thể chung so với phương pháp xây dựng ứng dụng gốc, quá trình phân tích xác định yêu cầu là tương đương nhau.

Sau khi xác định rõ yêu cầu và tính khả thi của ứng dụng cần xây dựng, việc phát triển và xây dựng ứng dụng sẽ được triển khai nhanh chóng với kỹ thuật IFML. Nhưng đối với những tính năng khó, hay những tính năng chưa được xác định rõ ràng sẽ là trở ngại lớn trong quá trình phát triển. Khó khăn này chủ yếu đến từ những tính năng yêu cầu tương tác với các công nghệ gốc của hệ điều hành, hay của thiết bị như : Dữ liệu có sẵn, các cảm biến, thời gian phát triển những tính năng này sẽ tương đối dài so với tổng thể toàn bộ thời gian phát triển ứng dụng. Trái lại, với phương pháp xây dựng ứng dụng gốc, tất cả yêu cầu đều có thể được thực hiện, thời gian phát triển mỗi tính năng không quá dài so với tổng thời gian phát triển ứng dụng.

Trong trường hợp cần một ứng dụng mẫu để trình bày với khách hàng, phương pháp IFML có thể cũng cấp ứng dụng mẫu một cách nhanh chóng. Trong khi đó, thời gian phát triển ứng dụng mẫu sử dụng phương pháp phát triển ứng dụng gốc tương đối dài, do khối lượng công việc lớn. Vòng đời phát triển ứng dụng là một lợi thế rất lớn của phương pháp IFML. Hình 3.25 và Hình 3.26 là biểu đồ GANTT về thời gian phát triển ứng dụng của cả hai phương pháp.



Hình 3.25: Biểu đồ thời gian phát triển ứng dụng MealNote_IFML



Hình 3.26: Biểu đồ thời gian phát triển ứng dụng gốc MealNote_Android

Qua hai biểu đồ có thể thấy được lợi thế của phát triển ứng dụng di động sử dụng kỹ thuật IFML. Điều này là đồng nghĩa với công sức phát triển giữa hai sản

phẩm với hai phương pháp. Theo ước tính ban đầu, cần khoảng 4 man-month để phát triển ứng dụng gốc hoàn chỉnh, nhưng trên thực tế cần khoảng 3 tháng để xây dựng hoàn thiện các tính năng chính. Trong khi đó, chỉ cần khoảng 1 man-month cho toàn bộ quá trình phát triển sản phẩm sử dụng kỹ thuật IFML.

Về thời gian phát triển ứng dụng, phương pháp xây dựng phần mềm sử dụng kỹ thuật mô hình hóa luồng tương tác tỏ ra vượt trội hơn kỹ thuật gốc nhiều lần.

3.3.7. Khả năng bảo trì, nâng cấp và bảo mật ứng dụng

Bảo trì, nâng cấp ứng dụng gốc là rất tốn kém và khó khăn, do tính chất phức tạp của cấu trúc dự án, nhà phát triển được giao nhiệm vụ bảo trì/nâng cấp cần hiểu được toàn bộ câu lệnh, luồng xử lý của ứng dụng ban đầu mới có thể tiến hành bảo trì/nâng cấp ứng dụng gốc. Ngoài ra còn một số rủi ro cao trong việc gây ra lỗi không mong muốn trong quá trình nâng cấp, bảo trì. Những lỗi này khó phát hiện trong quá trình sửa đổi mã nguồn và thường được báo cáo bởi người dùng.

Với ứng dụng sử dụng kỹ thuật IFML, các thành phần, luồng tương tác bao gồm cả tương tác với dữ liệu đều được mô hình hóa, nhà phát triển có thể tương tác một các trực quan. Do vậy, khả năng bảo trì/nâng cấp ứng dụng dạng này dễ dàng và tốn ít chi phí.

Một thách thức không nhỏ khác với các ứng dụng kiểu truyền thống là việc chuyển giao ứng dụng giữa các nhà phát triển khác nhau. Trong trường hợp ứng dụng được chuyển giao cho nhà phát triển khác, việc tìm hiểu để có thể tiếp tục bảo trì/nâng cấp ứng dụng sẽ tiêu tốn một thời gian dài. Điều này sẽ không xảy ra với ứng dụng xây dựng bằng kỹ thuật mô hình hóa luồng tương tác.

Về khả năng bảo mật, lợi thế thuộc về ứng dụng gốc, theo ý kiến khách quan, ứng dụng gốc đã có một cộng đồng phát triển lớn mạnh, có thể dễ dàng tìm được các thư viện, tính năng cho phép thực thi các phương pháp mã hóa, bảo mật dữ liệu giúp bảo mật dữ liệu người dùng thêm một tầng trong trường hợp bảo mật thiết bị bị vượt qua. IFML-WMP chưa hỗ trợ khả năng này. Xét về mặt bằng chung các ứng dụng hiện nay, hầu hết các ứng dụng đều xử dụng bảo mật của thiết bị hay hệ điều hành mà không có thêm bất kỳ biện pháp tăng cường bảo mật nào khác, do vậy, tuy chưa tốt như ứng dụng gốc nhưng IFML đáp ứng được hầu hết các yêu cầu về bảo mật của ứng dụng thông thường.

3.3.8. Các tiêu chí khác

Ứng dụng và thư viện hỗ trợ: Có rất ít tài nguyên và lựa chọn khi sử dụng các thư viện bên ngoài cho người phát triển ứng dụng di động sử dụng kỹ thuật IFML. Nguyên nhân chính do cộng đồng lập trình viên chưa lớn mạnh, hầu hết các công cụ và thư viện hỗ trợ hiện nay được cung cấp bởi WebRatio

Công cụ phát triển và sửa lỗi: Các công cụ phát triển ứng dụng gốc được cung cấp đầy đủ và chi tiết về khả năng tìm và gỡ lỗi, điều này cho phép nhà phát triển dễ dàng giải quyết các lỗi phát sinh trong quá trình xây dựng ứng dụng. Để có thể sửa một lỗi trong khi xây dựng ứng dụng với phương pháp IFML sẽ cần nhiều thời gian hơn kỹ thuật xây dựng ứng dụng gốc. Tuy nhiên, có rất ít lỗi xảy ra trong quá trình xây dựng ứng dụng di động sử dụng kỹ thuật IFML

Hiện nay, chỉ có một công cụ duy nhất cho phép phát triển ứng dụng di động sử dụng kỹ thuật IFML là WebRatio Mobile Platform. Trong khi đó, để phát triển ứng dụng Android với phương pháp xây dựng ứng dụng gốc, nhà phát triển có thể sử dụng Eclipse ADT hoặc Android Studio là hai công cụ phổ biến nhất hiện nay.

Sự độc lập về nền tảng: Nền tảng ứng dụng là một vấn đề lớn hiện nay, do có rất nhiều nền tảng di dộng khác nhau, các nhà phát triển lớn thường phải duy trì mỗi đội ngũ lập trình viên để phát triển một nền tảng. Có thể nhận thấy sự tốn kém chi phí trong việc này là không nhỏ cụ thể với hai nền tảng lớn hiện nay là Android và iOS. Tuy nhiên, với IFML nhà phát triển có thể xây dựng ứng dụng mà không cần quan tâm đến nền tảng nào. Một đội ngũ duy nhất chịu trách nhiệm phát triển ứng dụng cho cả hai nền tảng lớn, và chỉ phát triển một lần duy nhất. Đây là lợi ích lớn nhất của kỹ thuật mô hình hóa luồng tương tác bên cạnh việc giảm thiểu chi phí và thời gian phát triển ứng dụng.

3.4. Tổng kết chương

Bên cạnh các lợi ích không thể phủ nhận về tốc độ phát triển, chi phí phát triển và sự độc lập về nền tảng, kỹ thuật mô hình hóa luồng tương tác IFML còn tồn tại không ít hạn chế. Kỹ thuật IFML trong phát triển ứng dụng di động có thể được biểu diễn bằng cụm từ “Xây dựng một lần, chạy mọi nơi”, nó cũng đã biến một việc dường như bất khả thi thành hiện thực : Xây dựng ứng dụng di động mà không cần viết bất kỳ dòng lệnh nào.

Việc xây dựng một ứng dụng di động hoàn chỉnh sử dụng kỹ thuật mô hình hóa luồng tương tác IFML là hoàn toàn khả thi, tuy nhiên nhà phát triển cần cân

nhắc kỹ loại ứng dụng mình sẽ phát triển. Bên cạnh các điểm lợi ích không thể phủ nhận, kỹ thuật IFML vẫn còn tồn tại nhiều hạn chế cần khắc phục. Một số điểm hạn chế lớn nhất của IFML liên quan đến hiệu suất ứng dụng và trải nghiệm người dùng bắt nguồn từ sự yếu kém trong hỗ trợ tính năng gốc. Đây cũng là lý do chính khiến IFML chưa thể trở thành một kỹ thuật có thể thay thế hoàn toàn kỹ thuật xây dựng ứng dụng gốc.

Bảng 3.7 mô tả kết luận về lợi ích và hạn chế của kỹ thuật IFML và kỹ thuật xây dựng ứng dụng gốc dựa trên cơ sở các tiêu chí và kết quả đánh giá của các phần trước :

Bảng 3.7: Các tiêu chí và lựa chọn phương pháp thích hợp

Tiêu chí	Ứng dụng gốc	Ứng dụng sử dụng kỹ thuật IFML
Sự độc lập về nền tảng		√
Chi phí phát triển		√
Chuyển giao công nghệ		√
Thời gian phát triển ứng dụng nhanh		√
Khả năng bảo trì và nâng cấp		√
Hỗ trợ tính năng thiết bị và hệ điều hành	√	
Hiệu suất và trải nghiệm người dùng	√	
Bảo mật	√	
Thiết kế và giao diện	√	
Các ứng dụng và thư viện hỗ trợ	√	
Công cụ phát triển và sửa lỗi	√	
Các loại ứng dụng có thể xây dựng	√	

Dựa theo bảng trên, tác giả đề xuất các trường hợp phát triển ứng dụng và phương pháp lựa chọn thích hợp :

- Các trường hợp nên phát triển ứng dụng với kỹ thuật IFML:
  - Bị giới hạn về chi phí, IFML - WMP là sự lựa chọn tốt hơn
  - Cần phát triển ứng dụng nhanh chóng.
  - Cần phát triển ứng dụng đơn giản, có ít hiệu ứng, chỉ yêu cầu các tính năng gốc cơ bản, chỉ yêu cầu đáp ứng trải nghiệm người dùng cơ bản.
  - Ứng dụng đơn giản cần phát triển cho nhiều nền tảng.
- Các trường hợp nên phát triển ứng dụng sử dụng kỹ thuật phát triển ứng dụng gốc:

- Ứng dụng yêu cầu tối đa về trải nghiệm người dùng, hiệu suất sử dụng và sử dụng nhiều tính năng gốc.
- Các ứng dụng có nguồn kinh phí dồi dào, là các ứng dụng lớn và không giới hạn về công nghệ, tính năng mới.

Bản kết luận này nên được tham khảo và cân nhắc kỹ trước từng tình huống cụ thể nhằm lựa chọn phương án phù hợp nhất.

KẾT LUẬN

Sau khi nghiên cứu cơ sở lý thuyết và thực hiện luận văn này, tôi đã thu được nhiều kiến thức mới và có hướng nhìn rõ ràng và cụ thể hơn về phương pháp phát triển hướng mô hình, đặc biệt với kỹ thuật mô hình hóa luồng tương tác trong phát triển phần mềm nói chung và trong lập trình ứng dụng di động nói riêng.

Kỹ thuật mô hình hóa luồng tương tác là một cách tiếp cận mới cho việc phát triển ứng dụng di động hướng mô hình, đặc biệt trong lĩnh vực di động. Lợi ích thiết thực và to lớn của kỹ thuật này giúp đơn giản hóa quá trình phát triển ứng dụng di động cho nhiều nền tảng hiện nay. Ngoài việc dễ dàng nắm bắt và sử dụng, kỹ thuật này giúp giảm thiểu rất nhiều thời gian cũng như công sức phát triển ứng dụng trên nền tảng di động.

Bên cạnh những lợi ích to lớn, kỹ thuật này còn tồn tại không ít hạn chế về những mặt quan trọng như thiết kế, tùy chỉnh ứng dụng và đặc biệt về hiệu suất, trải nghiệm người dùng so với ứng dụng gốc. Từ các mặt này, nhà phát triển cần cân nhắc kỹ trước khi đưa ra quyết định chọn phát triển ứng dụng theo phương pháp nào.

Qua quá trình nghiên cứu, thực nghiệm, luận văn đạt được các kết quả chính như:

(1) Giới thiệu về phát triển ứng dụng hướng mô hình và lập trình ứng dụng di động. Trong đó tập trung tìm hiểu về kỹ thuật phát triển ứng dụng hướng mô hình nhằm áp dụng trong lập trình ứng dụng di động. Tìm hiểu về xu hướng mới trong lập trình đi động hiện nay: Lập trình di động đa nền tảng, các công cụ và ưu nhược điểm của chúng.

(2) Tìm hiểu và trình bày về kỹ thuật mô hình hóa luồng tương tác. Tìm hiểu sâu hơn về phát triển ứng dụng hướng mô hình với kỹ thuật mô hình hóa luồng tương tác cũng như cách tiếp cận cho kỹ thuật sinh mã tự động. Phạm vi ứng dụng nói chung và khả năng ứng dụng cho nền tảng di động nói riêng.

(3) Vận dụng và thực nghiệm kỹ thuật mô hình hóa luồng tương tác IFML trong quá trình phát triển ứng dụng di động MealNote. Xây dựng một phiên bản khác của ứng dụng MealNote sử dụng kỹ thuật xây dựng ứng dụng gốc nhằm so sánh, đánh giá kỹ thuật IFML. Đề xuất các tiêu chí đánh giá kỹ thuật IFML dựa theo các báo cáo khoa học, nghiên cứu liên quan.

(4) Đưa ra đánh giá về kỹ thuật mô hình hóa luồng tương tác một cách chi tiết dựa trên các tiêu chí chính cho quá trình phát triển/sử dụng ứng dụng di động. Các ưu điểm và các điểm hạn chế của kỹ thuật IFML, các loại ứng dụng có thể áp dụng IFML cũng như các ưu/nhược điểm của IFML trong quá trình phát triển ứng dụng di động.

Trong quá trình làm luận văn này, tôi nhận thấy một số hướng đi mới cũng như tính ứng dụng cao của sản phẩm thực nghiệm trong thực tế. Do đó, tôi đưa ra một số đề xuất và định hướng nghiên cứu tiếp theo như sau:

(1) Hoàn thiện và phát hành ứng dụng thực nghiệm MealNote. Do quỹ thời gian có hạn, việc phát triển ứng dụng MealNote chưa thật sự hoàn hảo với cả hai phiên bản kỹ thuật. Sau luận văn này, tôi sẽ tiếp tục hoàn thành và đưa ứng dụng MealNote tới người dùng hoàn toàn miễn phí trên cả hai phiên bản sử dụng kỹ thuật IFML và kỹ thuật xây dựng ứng dụng gốc, nhằm góp phần giảm thiểu nguy cơ từ tình trạng bất ổn về an toàn thực phẩm hiện nay. Thêm vào đó, qua sự đánh giá khách quan của người dùng trên hai phiên bản sẽ giúp bổ sung cho các nghiên cứu tiếp theo.

(2) Nghiên cứu và xây dựng phương pháp chuyển mô hình giữa UML và IFML. Phạm vi của luận văn này chỉ tập trung vào kỹ thuật mô hình hóa luồng tương tác IFML mà chưa đề cập đến khả năng chuyển đổi giữa UML và IFML. Tôi nhận thấy hướng đi mới này trong quá trình thực hiện luận văn, tuy nhiên mới chỉ bước đầu tìm hiểu. Định hướng nghiên cứu tiếp theo, tôi sẽ tập trung tìm hiểu và xây dựng công cụ, bộ thư viện nhằm mục đích chuyển đổi giữa các ngôn ngữ mô hình hóa khác sang IFML. Rút ngắn thêm một bước trong quá trình phát triển ứng dụng di động.

TÀI LIỆU THAM KHẢO

Tiếng Việt

1. Trần Đình Diễn, Huỳnh Quyết Thắng (2015), "Phát triển ứng dụng Web hướng mô hình dựa trên kỹ thuật web UWE", *Kỷ yếu Hội nghị Quốc gia lần thứ VIII về nghiên cứ cơ bản và ứng dụng Công Nghệ thông tin (FAIR),* Hà Nội.

Tiếng Anh

2. A. Pleuss(2005). "Mml: A language for modeling interactive multimedia applications". *In ISM '05: Proceedings of the Seventh IEEE International Symposium on Multimedia*, Washington, DC, USA. IEEE Computer Society, pp. 46, 54, 73.

3. Anmol Khandeparkal, Rashmi Gupta, B.Sindhya (2015), "An Introduction to Hybrid Platform Mobile Application Development", *International journal of Computer Applications*, Volume 118, Mumbai, India.

4. AnneKe K., Jos W., Wim B. (2003), *MDA Explained: The Model Driven Architecture: Practice and Promise*, Addison Wesley, United States.

5. Avinash Shrivas, Anandkumar Pardeshi (2013), "To Study and Design a Cross-Platform Mobile Application for Student Information System using PhoneGap Framework", *International Journal of Emerging Technology and Advanced Engineering*, Volume 3, Mumbai, India.

6. David Ameller (2009), *Considering Non-Functional Requirements in Model-Driven Engineering*, Master thesis, Polytechnic University of Catalonia, pp.13.

7. Davide Di Ruscio (2007), *Specification of Model Transformation and Weaving on Model Driven Engineering*, Ph.D Thesis, University of L'Aquila, Italy.

8. Eric Umuhoza (2015), *Domain-Specific Modeling and Code Generation for Cross-Platform Multi-Device mobile Apps*, Polytechnic University of Milan, Italy, pp. 6.

9. Florence T. Balagtas-Fernandez (2008), *Model-Driven Development of Mobile Applications*, Muniversity of Munich, Germany.

10. Frank Truyen (2006), "The Basics of Model Driven Architectture (MDA)", *The fast guide to Model Driven Architecture*, Cephas Consulting Corp, United States.

11. G. Bartolomeo, N. B. Melazzi, G. Cortese, A. Friday, G. Prezerakos, and R. W. S. Salsano (2006), *Sms: Simplifying mobile services for users and service*

providers. In Proceedings of the Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services, IEEE Computer Society.

12. Henning Heitkötter, Sebastian Hanschke and Tim A. Majchrzak (2012), "Comparing Cross-Platform Development approaches for Mobile Applications", *8th International Conference on Web information Systems and Technologies*, pp. 5,6.

13. J. Munoz, V. Pelechano, and J. Fons(2004), *Model driven development of pervasive systems*, European Research Consortium for Informatics and Mathematics (ERCIM) News, volume 58, pp. 50, 51.

14. J.D. Meier, Alex Homer, David Hill, Jason Taylor, Prashant Bansode, Lonnie Wall, Rob Boucher Jr, Akshay Bogawat (2008), *Mobile Application Architecture Guide*, Microsoft, United States, pp. 9.

15. Linus Oberg (2015), *Evaluation of Cross-Platform Mobile Development Tools*, Master Thesis, University in Umeå, Sweden.

16. Marco Brambilla (2013), *Interaction Flow Modeling Language - Model Driven Development of Software Front Ends*, Polytechic University Of Milan, Italy, pp. 85.

17. Marco Brambilla, Piero Fraternali (2014), *Interaction Flow Modeling Language*, Elsevier Publisher Inc, pp. 7,9,17.

18. Marco Brambilla, Andrea Mauri, Eric Umuhoza (2014), *Extending the Interaction Flow Modeling Language (IFML) for Model Driven Development of Mobile Applications Front End*, Polytechic University Of Milan, Italy.

19. Sanjeet Dhillon (2012), *An Evaluation Framework for Cross-Platform Mobile Application Development Tools*, Master Thesis, The University of Guelph, Canada, pp. 11.

20. Thomas Stahl, Markus Volter, Jorn Bettin, Arno Haase, Simon Helsen (2006), *Model-Driven Software Development*, Wiley and Sons Publisher Ltd, United State, pp. 34.

21. WebRatio (2016), *Mobile Platform 8.9 Release Notes*, pp. 25.

22. WebRatio (2016), *Sota, Architectural, Functional and Business requirements - Automatic Code Generation from models for cross-platform Mobile Application*, pp. 40, 41, 42, 57.

Một số nguồn tham khảo khác

23. http://www.omg.org/ifml/.

24. http://phonegap.com/.

25. https://developer.android.com/.

26.https://www.statista.com/ .

27.http://www.idc.com/ .

PHỤ LỤC

Phụ lục A: Xây dựng ứng dụng MealNote theo phương pháp truyền thống

1. Giới thiệu và cài đặt

Android Studio là một môi trường phát triển tích hợp (IDE) được phát triển bởi Google dựa trên IntelliJ IDEA nhằm hỗ trợ đặc biệt cho lập trình ứng dụng trên nền tảng Android. Được giới thiệu lần đầu vào ngày 16 tháng 5 năm 2013 tại sự kiện Google I/O - San Francisco, Mỹ như là công cụ lập trình ứng dụng Android chính thức của Google, Android Studio đã dần thay thế công cụ truyền thống vốn được sử dụng rộng rãi trước đó là Eclipse ADT nhanh chóng.

Chuẩn bị môi trường cài đặt như sau:

- Hệ điều hành Microsoft® Windows® 8/7/Vista (32- or 64-bit)
- Tối thiểu 2 GB RAM, tiêu chuẩn 4 GB RAM
- 400 MB dung lượng ổ đĩa cứng trống
- Ít nhất 1 GB cho Android Software Development Kit (SDK)
- Độ phân giải màn hình tối thiểu 1280 x 800
- Java Development Kit (JDK) 7
- Tùy chọn cho thiết bị giả lập: bộ vi sử lý Intel® hỗ trợ virtual technology Intel® VT-x.

Tải về và cài đặt Java Develipment Kit tại địa chỉ : *http://www.oracle.com/*

Tải về và cài đặt Android Studio tại địa chỉ :*https://developer.android.com*

Do hạn chế về mặt thời gian phát triển cũng như mục đích xây dựng ứng dụng Android gốc như một tiêu chuẩn để thực hiện so sánh, đánh giá kỹ thuật IFML và nhằm dành nhiều nội dung tập trung vào kỹ thuật IFML. Luận văn tập trung trình bày những ca sử dụng chính trên ứng dụng Android gốc.

2. Xây dựng ca sử dụng Login

Tạo lớp LoginSignupActiviy được mở rộng từ lớp ActionBarActivity với các thuộc tính như sau:

```
    public class LoginSignupActivity extends ActionBarActivity {
// Declare Variables
Button loginbutton;
Button signup;
String usernametxt;
```

```
String passwordtxt;
EditText password;
EditText username;
....
}
```

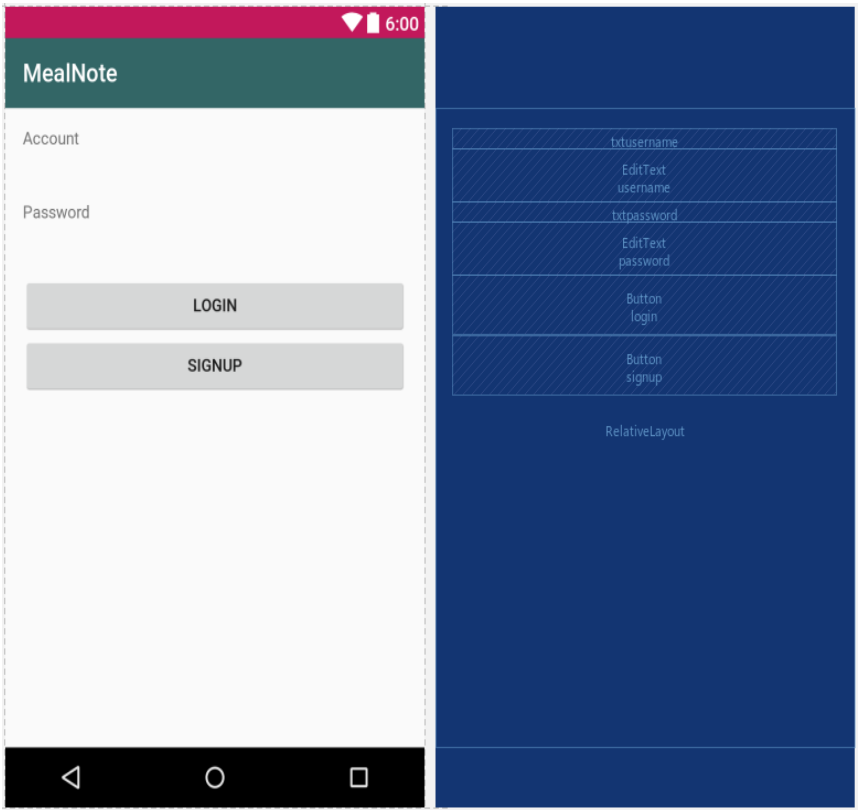
Xây dựng hai trường thuộc tính EditText nhằm cho phép người dùng điền tài khoản và mật khẩu:

```
username = (EditText) findViewById(R.id.username);
password = (EditText) findViewById(R.id.password);
```

Xây dựng nút (button) Login và xử lý sự kiện đăng nhập:

```
loginbutton.setOnClickListener(new View.OnClickListener() {
 @Override
  public void onClick(View v) {
            ...
        usernametxt = username.getText().toString();
        passwordtxt = password.getText().toString();
        ...
         }
 });
```

Thiết kế màn hình đăng nhập tại file *acitivty_login_signup.xml*. Thể hiện đồ họa trong Hình phụ lục A.1.



Hình phụ lục A.1: Thiết kế giao diện màn hình Login

3. Xây dựng ca sử dụng Signup

Tạo lớp LoginSignupActiviy được mở rộng từ lớp ActionBarActivity với các thuộc tính như sau:

```
    public class LoginSignupActivity extends ActionBarActivity {
...
Button signup;
String usernametxt;
String passwordtxt;
EditText password;
EditText username;
....
}
```

Xử lý sự kiện cho chức năng Signup là một hàm nằm trong lớp LoginSignupActivity tương tự chức năng Login:

```
signup.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
    ....
        usernametxt = username.getText().toString();
        passwordtxt = password.getText().toString();
        if(username.equals("") && password.equals("")){
            Toast.makeText(getApplicationContext(),
            "Please fill all field",
            Toast.LENGTH_LONG).show();
        }
        else{
        ParseUser user = new ParseUser();
        user.setUsername(usernametxt);
        user.setPassword(passwordtxt);

       }
    ....
    }
});
```

Do chức năng Login và Signup được xây dựng trên cùng một màn hình nên thiết kế giao diện đều nằm trên file *acitivty_login_signup.xml*. Dưới đây là thiết kế dạng thẻ xml của nút Signup:

```
    <Button
            android:id="@+id/signup"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_below="@+id/login"
            android:text="@string/SignupBtn" />
```

4. Xây dựng ca sử dụng View Meal In Week

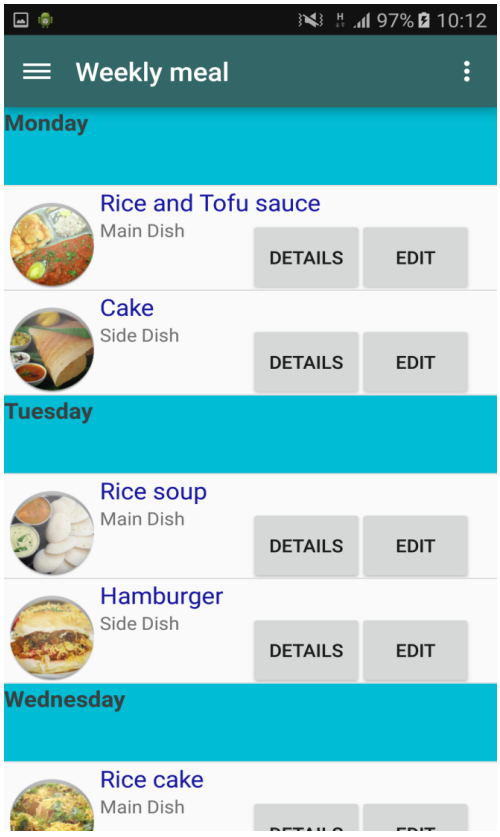
Ca sử dụng View Meal In Week được xây dựng với lớp *FrontPage* với các thuộc tính được khai báo trong lớp này :

```
        public class FrontPage extends ActionBarActivity implements
AdapterView.OnItemClickListener {

            Toolbar toolbar;
            private DrawerLayout drawerLayout;
            private ListView listview;
            private ExpandListAdapter ExpAdapter;
            private ArrayList<ExpandListGroup> ExpListItems;
            private ExpandableListView ExpandList;
            private String[] mDrawerTitles;
            private DrawerLayout mDrawerLayout;
        ...
```

Giao diện cho ca sử dụng này được thiết kế với file *activity_front_page.xml* , được liên kết với lớp FrontPage như sau:

```
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_front_page);
            ....
        }
```

File giao diện sau khi xây dựng được nhìn thấy như Hình phụ lục A.2:



Hình phụ lục A.2: Giao diện ca sử dụng View Meal In Week

5. Xây dựng ca sử dụng View List Meal

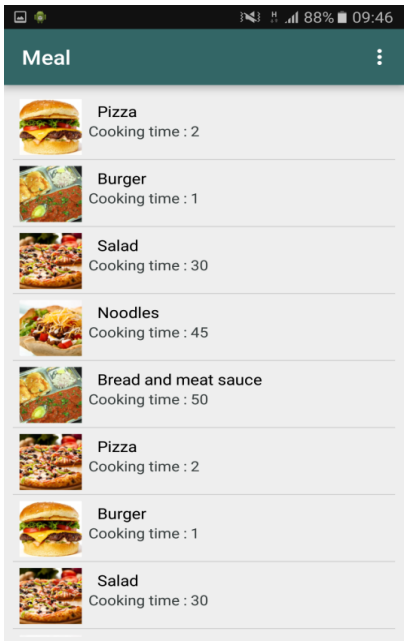
Ca sử dụng View List Meal được xây dựng với lớp tương ứng là Recipepage kế thừa từ ActionBarActivity như sau:

```
public class Recipepage extends ActionBarActivity {

   DbAdapter dbAdapter;
   private ListView lister;
   private List<recipedata> myRecipes = new ArrayList<recipedata>();
   private int itemClickedPosition;
   private boolean debugger = false;
   private static final String MyTag = "myApp";
   ...
}
```

Thực hiện lấy dữ liệu các món ăn từ cơ sở dữ liệu:

```
lister = (ListView) findViewById(R.id.listView);

Cursor cursor = dbAdapter.getAllRecipes();
List<recipedata> array = new ArrayList<recipedata>();

while (cursor.moveToNext()) {
// Truy vấn cơ sở dữ liệu
}

dbAdapter.close();
```

Giao diện ca sử dụng này được thiết kế với file *row_recipe.xml*. Giao diện ca sử dụng được thể hiện như Hình phụ lục A.3:



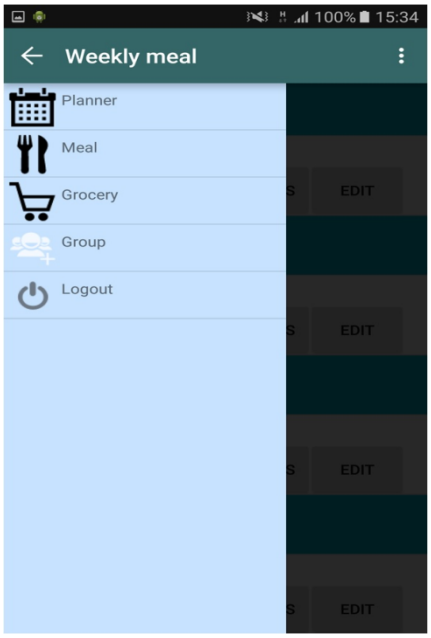
Hình phụ lục A.3: Màn hình ca sử dụng View List Meal

6. Xây dựng ca sử dụng View Menu

Màn hình View Menu được xây dựng bên trong màn hình chính ở lớp FrontPage dưới dạng một list view như sau:

```
private ListView mDrawerList;
mDrawerList.setOnItemClickListener(new DrawerItemClickListener());
...
private class DrawerItemClickListener implements ListView.OnItemClickListener {
    @Override
    public void onItemClick(AdapterView parent, View view, int position, long id)
{
        selectItem(position);
    }

}
```

Các sự kiện được xử lý trong hàm *selectItem(int position)* với tham số truyền vào là vị trí tương ứng của các chức năng. Hình phụ lục A.4 mô tả giao diện ca sử dụng View Menu:



Hình phụ lục A.4: Giao diện ca sử dụng View Menu

7. Các xử lý khác

Để có thể lưu trữ cũng như cung cấp dữ liệu, ta cần một bộ cơ sở dữ liệu, trong ứng dụng này tác giả sử dụng SQLite database là bộ cơ sở dữ liệu phổ biến cho hệ điều hành di động bao gồm Android/iOS.

Các thao tác với cơ sở dữ liệu được xử lý tại lớp DbAdapter với các thuộc tính được khai báo như sau:

```
public class DbAdapter {

    public static final String DATABASE_NAME = "meal_planer";
    public static final int DATABASE_VERSION = '9';
    public static final String TAG = "DBAdapter";
    ...
}
```

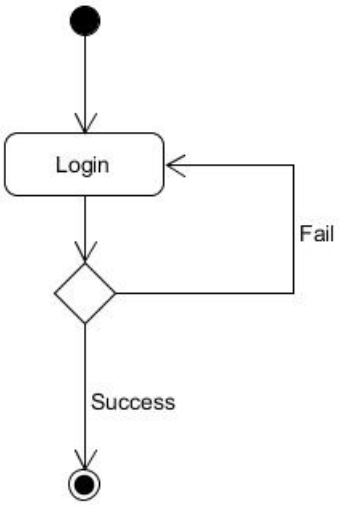
Các tuy vấn cơ sở dữ liệu cũng được thực hiện trong lớp này, một số ví dụ được thể hiện như bên dưới:

```
private static class DatabaseHelper extends SQLiteOpenHelper {

    DatabaseHelper(Context context) {

       super(context, DATABASE_NAME, null, DATABASE_VERSION);

    }

    private DatabaseHelper(Context context, String name,
SQLiteDatabase.CursorFactory factory, int version) {
        super(context, name, factory, version);
    }


    @Override
    public void onCreate(SQLiteDatabase db) {
        try {
                db.execSQL(CREATE_RECIPE_TABLE);
                db.execSQL(CREATE_INGREDIENT_TABLE);
                db.execSQL(CREATE_RECIPE_INGREDIENT_TABLE);

                ...
        }
    }
 }
```

Ngoài ra, các dữ liệu hình ảnh cần được tích hợp sẵn trong ứng dụng với độ phân giải khác nhau nhằm phù hợp với các loại kích thước màn hình và được đặt trong folder drawable với tên riêng biệt cho từng hình ảnh, điều này giúp ta dễ dàng sử dụng bằng cách gọi tên hình ảnh để truy xuất đến hình ảnh trực tiếp. Các biểu tượng tương ứng cũng cần được tích hợp trong folder mipmap.

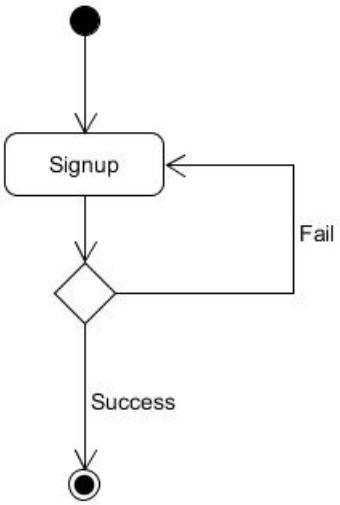
Phụ lục B: Biểu đồ hoạt động đặc tả các ca sử dụng của ứng dụng MealNote

1. Biểu đồ hoạt động ca sử dụng UC_001 : Login



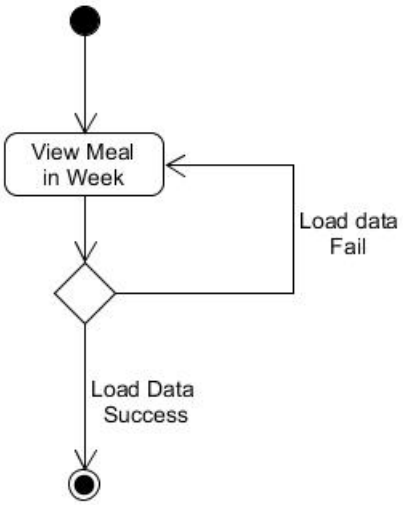
Hình phụ lục B.1: Biểu đồ hoạt động ca sử dụng Login

2. Biểu đồ hoạt động ca sử dụng UC_002 : Signup



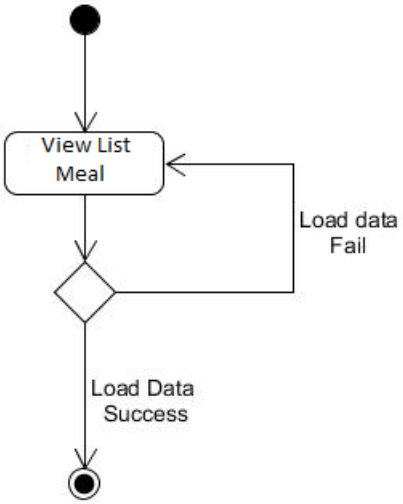
Hình phụ lục B.2: Biểu đồ hoạt động ca sử dụng Signup

3. Biểu đồ hoạt động ca sử dụng UC_003 : View Meal In Week



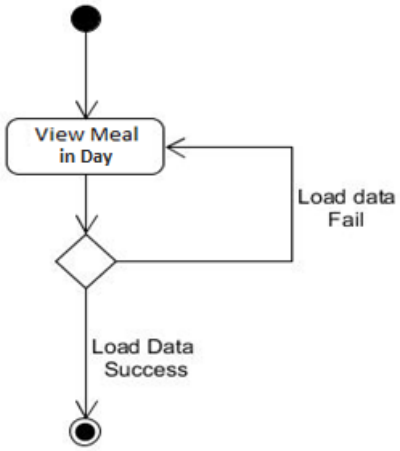
Hình phụ lục B.3: Biểu đồ hoạt động ca sử dụng View Meal in Week

4. Biểu đồ hoạt động ca sử dụng UC_004 : View List Meal



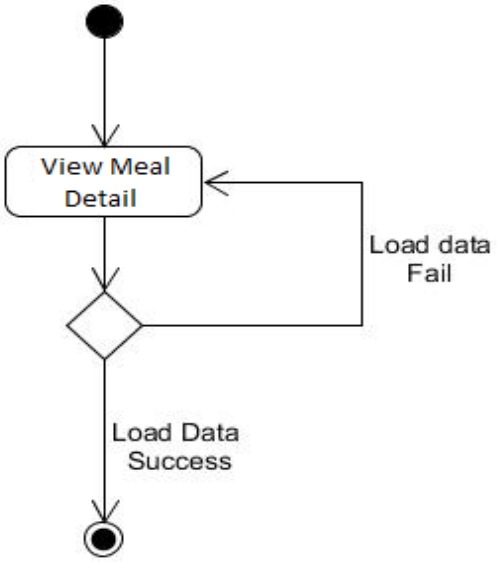
Hình phụ lục B.4: Biểu đồ hoạt động ca sử dụng View List Meal

5. Biểu đồ hoạt động ca sử dụng UC_005 : View Meal in Day



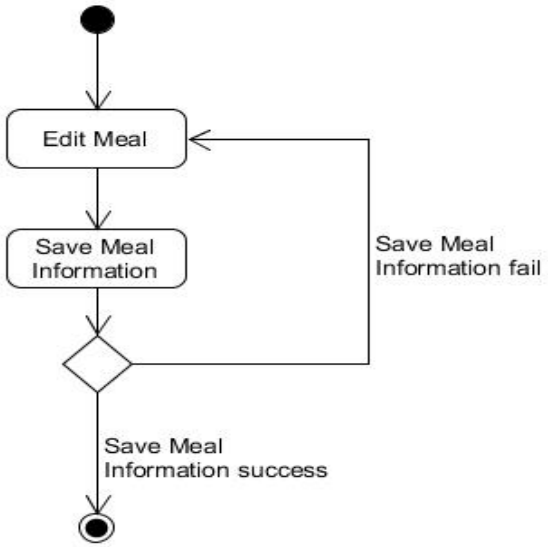
Hình phụ lục B.5: Biểu đồ hoạt động ca sử dụng View Meal in Day

6. Biểu đồ hoạt động ca sử dụng UC_006 : View Meal Details



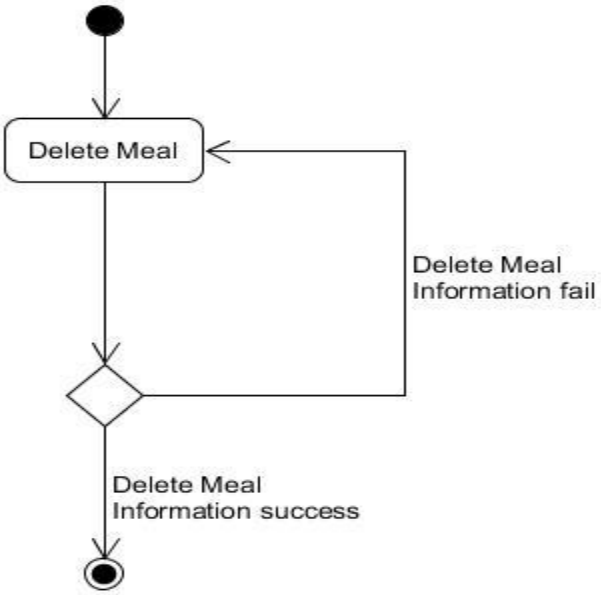
Hình phụ lục B.6: Biểu đồ hoạt động ca sử dụng View Meal Details

7. Biểu đồ hoạt động ca sử dụng UC_007 : Edit Meal



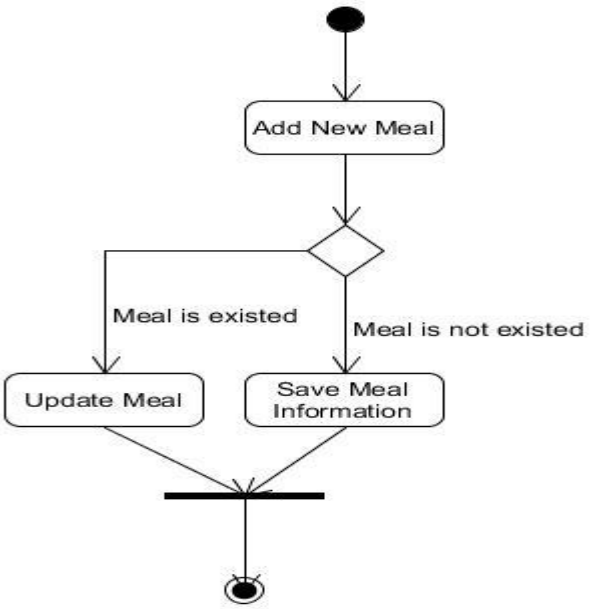
Hình phụ lục B.7: Biểu đồ hoạt động ca sử dụng Edit Meal

8. Biểu đồ hoạt động ca sử dụng UC_008 : Delete Meal



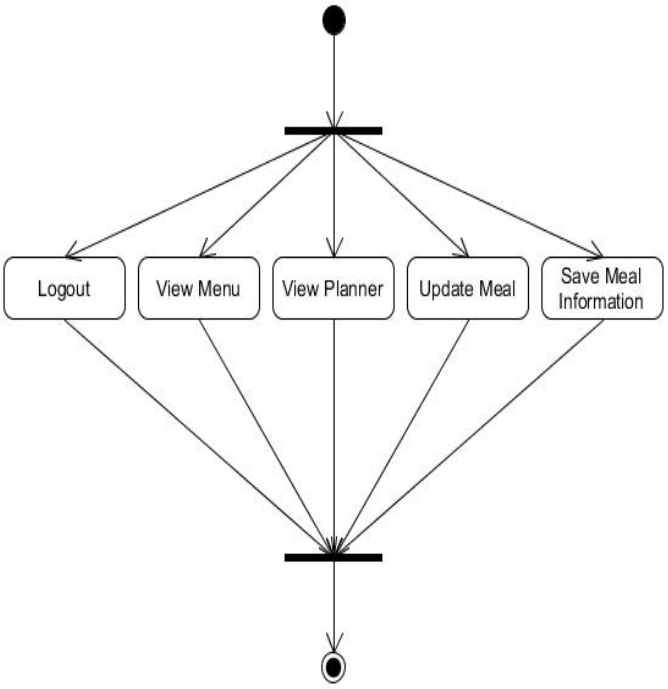
Hình phụ lục B.8: Biểu đồ hoạt động ca sử dụng Delete Meal

9. Biểu đồ hoạt động ca sử dụng UC_009 : Add New Meal



Hình phụ lục B.9: Biểu đồ hoạt động ca sử dụng Add New Meal

10. Biểu đồ hoạt động ca sử dụng UC_0010 – UC_012:



Hình phụ lục B.10: Biểu đồ hoạt động ca sử dụng UC_0010 – UC0014