**ĐẠI HỌC QUỐC GIA HÀ NỘI**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ**

****

**NGUYỄN THỊ HOÀI**

**PHÁT TRIỂN ỨNG DỤNG MÔ PHỎNG NỀN WEB DỰA TRÊN TÁC TỬ SỬ DỤNG NỀN TẢNG GAMA**

**ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY**

**Ngành: Truyền thông và mạng máy tính**

**HÀ NỘI – 2020**

**ĐẠI HỌC QUỐC GIA HÀ NỘI**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ**

**NGUYỄN THỊ HOÀI**

**PHÁT TRIỂN ỨNG DỤNG MÔ PHỎNG NỀN WEB DỰA TRÊN TÁC TỬ SỬ DỤNG NỀN TẢNG GAMA**

**ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY**

**Ngành: Truyền thông và mạng máy tính**

**Cán bộ hướng dẫn: ThS. Đặng Văn Đô**

**Cán bộ đồng hướng dẫn: TS. Phạm Mạnh Linh**

**HÀ NỘI - 2020**

# LỜI CẢM ƠN

Kết thúc quá trình học tập bốn năm tại trường Đại học Quốc Gia Hà Nội, cá nhân tôi thật vinh dự khi có cơ hội được làm đồ án tốt nghiệp với mục đích nghiên cứu, thực hành và vận dụng các kiến thức khoa học và thực tiễn đã được tích luỹ suốt quá trình học tập tại Khoa Công Nghệ Thông Tin. Đồ án này, trước hết, phục vụ mục đích minh chứng và lưu trữ dự án học tập cá nhân của tôi, và cũng là tài liệu ghi chép lại quá trình thực hiện và kết quả của đồ án. Thứ hai, văn bản này cũng là tài liệu để các giảng viên đánh giá kết quả học tập và nghiên cứu cuối cùng của tôi trong quá trình học tập năm cuối ở Đại học Quốc Gia Hà Nội.

Tôi xin gửi lời cảm ơn chân thành nhất đến giảng viên hướng dẫn ThS. Đặng Văn Đô và TS. Phạm Mạnh Linh đã giúp đỡ tôi hoàn thành đồ án và cho tôi những chỉ dẫn bổ ích. Tiếp theo, tôi xin gửi lời cảm ơn Khoa Công Nghệ Thông Tin, trường Đại học Quốc Gia Hà Nội đã hướng dẫn tôi trong suốt 4 năm học tại trường và trao cho tôi niềm vinh dự được làm đồ án tốt nghiệp này. Trong hai tháng dành thời gian cho chủ đề “***Phát triển ứng dụng mô phỏng nền Web dựa trên tác tử sử dụng nền tảng GAMA***”, tôi đã học hỏi và tích lũy thêm rất nhiều kiến thức bổ ích và giá trị, ở cả hai góc độ lý thuyết và thực hành.

Tôi cũng xin cảm ơn những người bạn và các cựu sinh viên đã cho tôi nhiều lời khuyên và hướng dẫn để tôi hoàn thành đồ án với kết quả tốt nhất. Tôi cũng xin gửi lời cảm ơn và lòng biết ơn to lớn đến cha mẹ và gia đình của tôi, những người luôn ở phía sau ủng hộ và động viên tôi.

Đồ án này được tài trợ một phần bởi đề tài QG.20.55 của Đại học Quốc gia Hà Nội.

# TÓM TẮT

Trong những năm gần đây, mô hình hóa dựa trên tác tử (agent-based modeling) đã được sử dụng ngày càng rộng rãi để nghiên cứu các hệ thống phức tạp. Điều này xuất phát từ khả năng của tác tử cho phép thể hiện nhiều mức độ tương tác với những thể hiện môi trường chi tiết và phức tạp. Để đối phó với sự phức tạp gia tăng này, các công cụ mô hình hóa và mô phỏng mạnh mẽ đã được ra đời. Trong số các công cụ này, nền tảng mô hình hóa và mô phỏng đa tác tử trên nền hệ thống thông tin địa lý GAMA (GIS Agent-based Modeling Architecture) là một thí dụ tiêu biểu. GAMA đã được ứng dụng trong nhiều dự án nghiên cứu xung quanh các vấn đề nóng như dự báo, phòng chống thiên tai, thảm họa, dịch bệnh, v.v...

Tuy nhiên, tool mô phỏng GAMA thường cho phép chạy một mô phỏng chỉ trên một máy chủ. Phần mềm bắt buộc cần phải cài trên thiết bị máy tính thì mới sử dụng được. Do vậy, nếu người dùng muốn sử dụng ứng dụng, thì phải mang theo mình chiếc máy tính cá nhân để phục vụ cho công việc. Bên cạnh đó, việc phát triển một hệ thống mô phỏng trên nền desktop bị giới hạn về số lượng tác tử cần mô phỏng. Chính vì vậy để đồ án **“Phát triển ứng dụng mô phỏng nền Web dựa trên tác tử sử dụng nền tảng GAMA”** là một dự án phần mềm được hình thành với mục đích đưa một ứng dụng desktop đưa lên nền Web cho phép nhiều người cùng một lúc thực hiện các mô phỏng mà không cần cài đặt. Phần mềm cho phép người dùng thao tác tạo mới, chỉnh sửa file model, thay đổi các thông số để chạy mô phỏng các hệ thống đa tác tử dựa trên nền tảng GAMA.

# LỜI CAM ĐOAN

Tôi là Nguyễn Thị Hoài, mã số sinh viên 16022433, sinh viên lớp N, khoá 61. Giảng viên hướng dẫn là ThS. Đặng Văn Đô và TS. Phạm Mạnh Linh. Tôi xin cam đoan toàn bộ nội dung được trình bày trong đồ án **“*Phát triển ứng dụng mô phỏng nền Web dựa trên tác tử sử dụng nền tảng GAMA*”** là kết quả quá trình tìm hiểu và nghiên cứu của tôi. Các dữ liệu được nêu trong đồ án là hoàn toàn trung thực, phản ánh đúng kết quả thực tế. Mọi thông tin trích dẫn đều tuân thủ các quy định về sở hữu trí tuệ; các tài liệu tham khảo được liệt kê rõ ràng. Tôi xin chịu hoàn toàn trách nhiệm với những nội dung được viết trong đồ án này.

|  |  |
| --- | --- |
|  | *Hà Nội, ngày … tháng … năm 2020*  **NGƯỜI CAM ĐOAN**  **Nguyễn Thị Hoài** |

# MỤC LỤC

[PHẦN MỞ ĐẦU 1](#_Toc57909543)

[1. Lý do lựa chọn đề tài 1](#_Toc57909544)

[2. Đối tượng và phạm vi nghiên cứu 2](#_Toc57909545)

[3. Phương pháp nghiên cứu 2](#_Toc57909546)

[4. Cấu trúc của đồ án 2](#_Toc57909547)

[CHƯƠNG 1: TỔNG QUAN VỀ GAMA 3](#_Toc57909548)

[1.1. Mô hình tác tử và đa tác tử 3](#_Toc57909549)

[1.2. Tổng quan về GAMA 5](#_Toc57909550)

[1.3. Ngôn ngữ GAML 11](#_Toc57909551)

[1.4. Mô phỏng (Simulation) 12](#_Toc57909552)

[CHƯƠNG 2: TỔNG QUAN LÝ THUYẾT LẬP TRÌNH 14](#_Toc57909553)

[2.1. Spring Boot: Framework 14](#_Toc57909554)

[2.2.. Hibernate: Framework 14](#_Toc57909555)

[2.3. Sơ lược postgreSQL 16](#_Toc57909556)

[2.4. Angular: Framework 18](#_Toc57909557)

[2.5. Các phần mềm hỗ trợ 22](#_Toc57909558)

[CHƯƠNG 3: THIẾT KẾ HỆ THỐNG VÀ CƠ SỞ DỮ LIỆU 24](#_Toc57909559)

[3.1. Thiết kế kiến trúc 24](#_Toc57909560)

[3.2. Biểu đồ usecase 26](#_Toc57909561)

[3.3. Sequence diagram 27](#_Toc57909562)

[3.4. Thiết kế cơ sở dữ liệu 30](#_Toc57909563)

[CHƯƠNG 2: TRIỂN KHAI CHƯƠNG TRÌNH VÀ KẾT QUẢ 33](#_Toc57909564)

[3.1. Xác định hệ thống 33](#_Toc57909565)

[3.2. Giải pháp 35](#_Toc57909566)

[3.3. Quá trình cài đặt hệ thống 38](#_Toc57909567)

[3.4. Giao diện chương trình 39](#_Toc57909568)

[KẾT LUẬN 48](#_Toc57909569)

[TÀI LIỆU THAM KHẢO 50](#_Toc57909570)

# DANH MỤC CHỮ VIẾT TẮT

CNTT: Công nghệ thông tin

HTML: Hypertext Markup Language

CSS: Cascading Style Sheets

GUI: Graphical User Interface

API: Application Programming Interface

XML: eXtensible Markup Language

HTTP: HyperText Transfer Protocol

MVVM: Model-View-ViewModel

IDE: Integrated Development Environment

IBM: International Business Machines

MVC: Model – View – Controller

POJO: Plain Old Java 0bject

JPA: Java Persistence API

HQL: Hibernate Query Language

LGPL: Lesser General Public License

JDBC: Java Database Connectivity

GAMA: GIS & Agent-based Modeling Architecture

GAMAL: GAMA Modeling Language

# DANH MỤC HÌNH VẼ

[Hình 1.1 Biểu diễn một mô phỏng hướng tác tử 5](#_Toc57812836)

[Hình 1.2 Cấu trúc của một chương trình viết bằng GAMA 12](#_Toc57812836)

[Hình 3.1 Hoạt động ở phía client 24](#_Toc57812836)

[Hình 3.2 Hoạt động ở phía server 25](#_Toc57812837)

[Hình 3.3 Biểu đồ usecase 26](#_Toc57812838)

[Hình 3.4 Sequence diagram sign in/sign up 27](#_Toc57812839)

[Hình 3.5 Sequence diagram add/edit/delete project 28](#_Toc57812840)

[Hình 3.6 Sequence diagram run simulation model 29](#_Toc57812841)

[Hình 3.7 Lược đồ cơ sở dữ liệu 30](#_Toc57812842)

[Hình 4.1 Các chức năng quản lý project 34](file:///C:\Users\duong\Desktop\DATN-NTH%20(1).docx#_Toc57812843)

[Hình 4.2 Các chức năng trong quản lý file model 34](file:///C:\Users\duong\Desktop\DATN-NTH%20(1).docx#_Toc57812844)

[Hình 4.3 Các cách tạo file mô hình 35](#_Toc57812845)

[Hình 4.4 Giao diện chỉnh sửa file mô hình 36](#_Toc57812846)

[Hình 4.5 Giao diện thay đổi số bước mô phỏng 36](#_Toc57812847)

[Hình 4.6 Giao diện thay đổi các giá trị parameter 37](#_Toc57812848)

[Hình 4.7 Ví dụ về file có đuôi mở rộng .xml 37](#_Toc57812849)

[Hình 4.8 File cấu hình proxy.conf.json 39](#_Toc57812850)

[Hình 4.9 Màn hình Đăng nhập 39](file:///C:\Users\duong\Desktop\DATN-NTH%20(1).docx#_Toc57812851)

[Hình 4.10 Màn hình Đăng ký 40](#_Toc57812852)

[Hình 4.11 Màn hình Project (dạng folder) 40](file:///C:\Users\duong\Desktop\DATN-NTH%20(1).docx#_Toc57812853)

[Hình 4.12 Màn hình Project (dạng list) 41](file:///C:\Users\duong\Desktop\DATN-NTH%20(1).docx#_Toc57812854)

[Hình 4.13 Dialog thêm mới project 41](file:///C:\Users\duong\Desktop\DATN-NTH%20(1).docx#_Toc57812855)

[Hình 4.14 Dialog edit thông tin project 42](file:///C:\Users\duong\Desktop\DATN-NTH%20(1).docx#_Toc57812856)

[Hình 4.15 Search theo tên project(dạng folder) 42](file:///C:\Users\duong\Desktop\DATN-NTH%20(1).docx#_Toc57812856)

[Hình 4.16 Search theo tên project(dạng list)](file:///C:\Users\duong\Desktop\DATN-NTH%20(1).docx#_Toc57812856) 43

[Hình 4.17 Sort theo tên project(dạng folder)](file:///C:\Users\duong\Desktop\DATN-NTH%20(1).docx#_Toc57812856) 43

[Hình 4.18 Sort theo tên project(dạng list)](file:///C:\Users\duong\Desktop\DATN-NTH%20(1).docx#_Toc57812856) 44

[Hình 4.19 Giao diện thêm mới, edit file model 44](file:///C:\Users\duong\Desktop\DATN-NTH%20(1).docx#_Toc57812857)

[Hình 4.20 Chỉnh sửa giá trị của các parameter 45](#_Toc57812858)

[Hình 4.21 Giao diện chạy mô phỏng (1) 45](file:///C:\Users\duong\Desktop\DATN-NTH%20(1).docx#_Toc57812859)

[Hình 4.22 Giao diện mô phỏng dạng lưới (2) 46](#_Toc57812860)

[Hình 4.23 Giao diện mô phỏng dạng biểu đồ (3) 46](#_Toc57812861)

[Hình 4.24 Giao diện mô phỏng (4) 47](#_Toc57812862)

# PHẦN MỞ ĐẦU

## 1. Lý do lựa chọn đề tài

Để đáp ứng nhu cầu mô hình hóa dựa trên tác tử của nhiều hệ thống nghiên cứu các vấn đề nóng như dự báo, phòng chống thiên tai, thảm họa, dịch bệnh,... GAMA ra đời nhằm mục đích cung cấp cho các chuyên gia lĩnh vực, người lập mô hình và nhà khoa học máy tính một môi trường phát triển mô hình và mô phỏng hoàn chỉnh để xây dựng các mô phỏng đa tác tử rõ ràng trong không gian trên nền desktop.

Về desktop application, có thể nhận thấy rằng desktop app có tốc độ xử lý dữ liệu nhanh chóng, đảm bảo an toàn, bảo mật thông tin. Tiếp đến, hầu hết các desktop app không yêu cầu người dùng bắt buộc phải kết nối mạng internet. Ưu điểm này sẽ là một thế mạnh lớn khi người dùng đang cần sử dụng một chức năng nào đó mà không thể kết nối mạng, tránh rủi ro mất dữ liệu hoặc kết nối mạng không ổn định. Tuy nhiên, desktop app cũng có những bất lợi riêng. Phần mềm chạy trên nền tảng desktop bắt buộc cần phải cài trên thiết bị máy tính thì mới sử dụng được. Do vậy, nếu người dùng muốn sử dụng ứng dụng, thì phải mang theo mình chiếc máy tính cá nhân để phục vụ cho công việc. Ngoài ra, đồ họa trên desktop không cao nên giao diện phần mềm sẽ thiếu tính trực quan, hơi khó thao tác, không thân thiện với người dùng.

Chính vì vậy, đồ án này nhằm mục đích đưa ứng dụng GAMA chạy trên nền desktop lên nền web, đây là nền tảng được sử dụng phổ biến rộng rãi. Ưu điểm đầu tiên, nó có thể chạy trên tất cả các trình duyệt của thiết bị máy tính và di động, chỉ cần hỗ trợ phiên bản javascript và HTML. Web app không cần cài đặt trên thiết bị, người dùng có thể truy cập ứng dụng chỉ với một đường link. Tiếp đến, lập trình viên có thể thiết kế phần mềm web trên hầu hết các hệ điều hành vì không giới hạn môi trường lập trình. Khả năng chia sẻ ứng dụng cao hơn các nền tảng khác, vì người dùng chỉ cần sao chép đường link và chia sẻ cho người cần dùng là đã có thể truy cập được vào hệ thống. Giao diện web app luôn luôn được cập nhật với nhiều tiện ích đi kèm, mà không bị giới hạn bởi hệ thống. Dù có rất nhiều ưu điểm, cũng phải kể đến một số nhược điểm của nền tảng này. Web app có tốc độ xử lý dữ liệu không cao. Web app cũng phụ thuộc hoàn toàn vào đường truyền internet, lập trình viên khi xây dựng ứng dụng cũng luôn phải có kết nối mạng. Người dùng phải đảm bảo kết nối internet cũng như đường link phải chính xác thì mới truy cập được vào ứng dụng.

## 2. Đối tượng và phạm vi nghiên cứu

Đối tượng nghiên cứu: Hệ thống đa tác tử, hệ thống mô phỏng GAMA, Web application framework

Phạm vi nghiên cứu: Nghiên cứu xây dựng và phát triển phần mềm mô phỏng hệ thống đa tác tử nền Web dựa trên một desktop application có sẵn.

## 3. Phương pháp nghiên cứu

- Phương pháp nghiên cứu lý thuyết.

+ Các tài liệu về cơ sở lý thuyết: ngôn ngữ GAML, lý thuyết mô hình hóa và mô phỏng đa tác tử

+ Các tài liệu mô tả công cụ GAMA

+ Các tài liệu liên quan đến một số nghiên cứu

- Phương pháp nghiên cứu thực tiễn.

- Phương pháp thực nghiệm.

+ Thực hiện tìm hiểu, thu thập số liệu

+ Phân tích yêu cầu và phạm vi của đồ án để từ đó xác định những gì phải làm.

+ Khảo sát mô hình thực tế, thu thập thông tin cho việc mô hình hóa

## 4. Cấu trúc của đồ án

Ngoài phần mở đầu, kết luận, danh mục tài liệu tham khảo, phụ lục, nội dung đồ án tốt nghiệp được kết cấu thành 04 chương:

**Chương 1: Tổng quan về GAMA**

**Chương 2: Tổng quan lý thuyết lập trình**

**Chương 3: Thiết kế cơ sở dữ liệu và thuật toán**

**Chương 4: Triển khai chương trình và kết quả**

# TỔNG QUAN VỀ GAMA

## 1.1. Mô hình tác tử và đa tác tử

Ở các thế hệ máy tính đầu tiên, số người có thể sử dụng máy tính rất hạn chế. Họ đều là chuyên gia về máy tính hoặc lập trình viên chuyên nghiệp, được trang bị kiến trúc đặc biệt để làm việc với máy tính. Ngược lại, yêu cầu với máy tính ngày nay là phục vụ người dùng càng tốt, thể hiện ở một loạt yêu cầu như giao diện thân thiện và trực giác, khả năng thich nghi với yêu cầu người dùng, cho phép cung cấp thông tin có tính cá nhân hóa với từng đối tượng sử dụng. Để xây dựng các hệ thống tính toán thỏa mãn các đặc điểm và yêu cầu nói trên một số hướng nghiên cứu và ứng dụng mới của máy tính đã ra đời, trong đó có tác tử và hệ đa tác tử đang trở thành công nghệ của tương lai để giải quyết các vấn đề nêu trên.

*1.1.1. Định nghĩa tác tử*

Tác tử (agent) là hệ thống tính toán hoạt động tự chủ trong một môi trường nào đó, có khả năng cảm nhận môi trường và tác động vào môi trường. Hệ thống tính toán có thể là phần cứng, phần mềm, hoặc cả phần cứng lẫn phần mềm. Bất cứ tác tử nào cũng tồn tại và hoạt động trong một môi trường nhất định. Tác tử nhận thông tin từ môi trường qua các cơ quan cảm nhận và tác động vào môi trường bằng các cơ quan tác động. Đối với các tác tử phần cứng, cơ quan cảm nhận có thể là các cảm biến, camera, cơ quan tác động có thể là các bộ phận cơ học, quang học hoặc âm thanh. Đối với các tác tử là chương trình phần mềm, môi trường hoạt động thông thường là các máy tính hoặc mạng máy tính. Việc cảm nhận môi trường và tác động được thực hiện thông qua các lời gọi hệ thống.

Nói chung, tác tử có thể được thiết kế để hoạt động trong nhiều dạng môi trường khác nhau. Một điểm cần chú ý là cảm nhận về môi trường của tác tử có thể không đầy đủ do môi trường quá phức tạp hoặc có chứa các yếu tố không xác định. Một yêu cầu quan trọng đối với tác tử là tính tự chủ. Ở đây tự chủ được hiểu như là khả năng các tác tử hành động không cần đến sự can thiệp trực tiếp của người hay các tác tử khác: tác tử hoàn toàn có khả năng kiểm soát trạng thái cũng như hành vi của mình trong một thời gian tương đối dài.

Một số ứng dụng của tác tử đó là:

+ Ứng dụng trong quản lý sản xuất.

+ Tác tử quản lý quá trình và luồng công việc (workflow).

+ Tác tử thu thập và quản lý thông tin.

+ Tác tử phục vụ thương mại điện tử.

+ Tác tử giao diện.

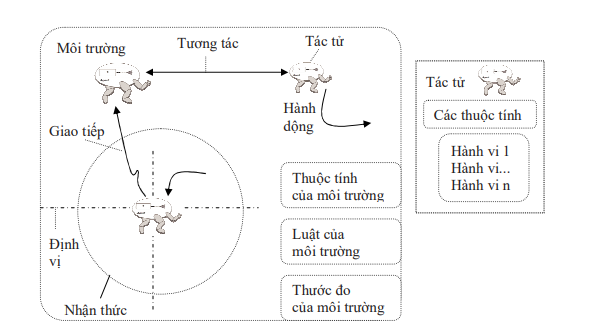
+ Trò chơi sử dụng tác tử.

*1.1.2. Hệ đa tác tử, mô phỏng hướng đa tác tử*

Mô hình hướng tác tử là một hệ thống bao gồm nhiều thực thể hay còn gọi là nhiều tác tử, những thực thể này phát triển trong cùng một môi trường, môi trường được thiết kế như một thực thể đặc biệt để các tác tử khác định vị trong đó. Những tác tử được gán những thuộc tính, các hành vi và những khả năng nhận thức và giao tiếp. Tập hợp những giá trị của các thuộc tính của một thể hiện của thực thể hình thành nên trạng thái của thực thể đó. Hợp của tập hợp các trạng thái của các thực thể hình thành nên trạng thái vi mô của hệ thống hay người ta còn gọi đơn giản là trạng thái của hệ thống.

Hành vi là những luật điều khiển sự thay đổi trạng thái thông qua việc can thiệp vào các trạng thái của các tác tử mang những hành vi này, cũng như những trạng thái của những tác tử khác xuất hiện trong các sự kiện, hành động, giao tiếp hay tương tác được mô tả trong các hành vi.

Hình 1.1 mô tả tổng quan về một mô hình hướng đa tác tử gồm đầy đủ các thành phần đã nêu: môi trường, tác tử, hành vi, sự giao tiếp,...



*Hình 1.1. Biểu diễn một mô phỏng hướng tác tử*

## 1.2. Tổng quan về GAMA

*1.2.1 Giới thiệu về GAMA*

GAMA là một nền tảng mô phỏng, nhằm mục đích cung cấp cho các chuyên gia lĩnh vực, người lập mô hình và nhà khoa học máy tính một môi trường phát triển mô hình và mô phỏng hoàn chỉnh để xây dựng các mô phỏng đa tác tử rõ ràng trong không gian. Nó đã được phát triển lần đầu tiên bởi nhóm nghiên cứu Việt – Pháp MSI (đặt tại IFI, Hà Nội, và một phần của Đơn vị Nghiên cứu Quốc tế IRD/SU UMMISCO) từ năm 2007 đến năm 2010, và hiện được phát triển bởi một nhóm các đối tác học thuật và công nghiệp của UMMISCO, trong đó có Đại học Rouen, Pháp, Đại học Toulouse 1, Pháp, Đại học Orsay, Pháp, Đại học Cần Thơ, Việt Nam, Đại học Quốc gia Hà Nội, EDF R&D, Pháp, và CEA LISC, Nước Pháp.

GAMA cho phép:

+ Thiết kế, tạo mẫu và lựa chọn công cụ mô hình đồ họa của nó.

+ GIS và các mô hình Data-Driven: Khởi tạo các tác tử (agent) từ bất kỳ bộ dữ liệu, bao gồm cả dữ liệu GIS, và thực hiện các mô phỏng quy mô lớn (lên đến hàng triệu các agent).

+ Xác định thí nghiệm phong phú về mẫu mã.

+ Thiết kế giao diện người dùng phong phú hỗ trợ kiểm tra sâu về các agent.

*1.2.2. Đặc điểm của GAMA*

+ Có thể sử dụng được cho nhiều ứng dụng khác nhau, ví dụ như: mô phỏng sự lan truyền dịch bệnh ở gia súc, mô phỏng giao thông đường bộ, mô hình gia tăng dân số, mô hình biến động số lượng giữa các loài động vật trong chuỗi thức ăn, ...

+ Phần mềm được phát triển theo giấy phép GPL/LGPL: miễn phí

+ Tích hợp ngôn ngữ mô hình hóa (GAML) và môi trường phát triển: giúp người sử dụng (ngay cả đối với người không phải kỹ sư tin học) có thể xây dựng được mô hình một cách dễ dàng và nhanh chóng.

+ Phát triển theo ngôn ngữ Java: dễ dàng mở rộng để đưa các dữ liệu mới vào

+ Công cụ tích hợp để phân tích mô hình: không gian tham số rộng để nghiên cứu và căn chỉnh mô hình.

*1.2.3. Phạm vi của GAMA*

+ Hỗ trợ phát triển các mô hình phức tạp

+ Tích hợp trực tiếp dữ liệu địa lý và hệ thống thông tin địa lý với mô hình dựa trên tác tử

+ Tích hợp cách tiếp cận phương pháp luận để xây dựng mô hình nhiều cấp độ

+ Tích hợp nhiều công cụ: công cụ hỗ trợ quá trình quyết định sử dụng nhiều tiêu chí, chức năng phân nhóm, công cụ thống kê…

+ Dễ mở rộng nhờ cấu trúc mở gắn với công nghệ Java: khung OSGI và chú thích trong Java

*1.2.4. Nền tảng*

GAMA bao gồm một ứng dụng duy nhất dựa trên kiến ​​trúc RCP do Eclipse cung cấp. Trong phần mềm ứng dụng duy nhất này, thường được gọi là platform, người dùng có thể thực hiện mà không cần thêm phần mềm của bên thứ ba, hầu hết các hoạt động liên quan đến mô hình hóa và mô phỏng, cụ thể là chỉnh sửa mô hình và mô phỏng, trực quan hóa và khám phá chúng bằng các công cụ chuyên dụng.

GAMA được thực hiện bằng cách trước tiên cung cấp một số nền tảng về các khái niệm quan trọng được tìm thấy trong toàn bộ platform, đặc biệt là các khái niệm về không gian làm việc và dự án, đồng thời giải thích cách tổ chức và điều hướng thông qua các mô hình. Sau đó, xem xét phiên bản của các mô hình và các công cụ và thành phần khác nhau của nó. Cuối cùng, chỉ ra cách chạy thử nghiệm trên các mô hình này và những gì hỗ trợ mà giao diện người dùng có thể cung cấp cho người dùng trong tác vụ này.

*1.2.5. Headless Mode*

Mục đích của tính năng này là có thể chạy một hoặc nhiều giao diện phiên bản của GAMA mà không cần bất kỳ giao diện người dùng nào, để các mô hình và thử nghiệm có thể được khởi chạy trên lưới hoặc theo cụm. Nếu không có GUI, dung lượng bộ nhớ cũng như tốc độ mô phỏng thường được cải thiện đáng kể. Trong chế độ này, GAMA chỉ có thể được sử dụng để chạy thử nghiệm và không thể chỉnh sửa hoặc quản lý các mô hình.

- Command: Có 2 cách để chạy thử nghiệm GAMA trong chế độ headless: sử dụng tập lệnh shell script chuyên dụng (khuyến nghị) hoặc chạy trực tiếp từ dòng lệnh. Các tập lệnh này đều có 2 đối số: tệp thử nghiệm và đầu ra.

+ Shell Script: Có thể được tìm thấy trong GAMA, tên là gamaHeadless.sh trên MacOSX và Linux hoặc gamaHeadless.bat trên Windows.



Trong đó:

* tệp thông số đầu vào $1: tệp xml xác định các thông số thử nghiệm và đầu ra
* đường dẫn thư mục đầu ra $2: thư mục chứa kết quả mô phỏng

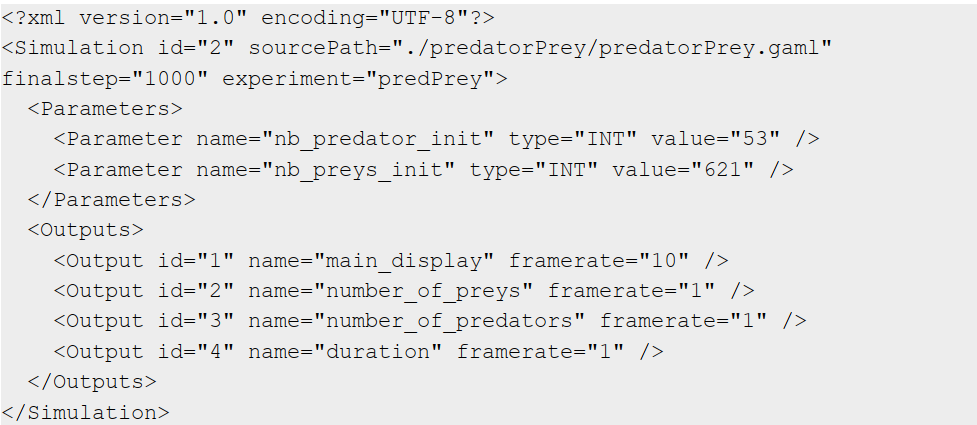
+ Java Command:



Trong đó:

* tệp thông số đầu vào $1: tệp xml xác định các thông số thử nghiệm và đầu ra
* đường dẫn thư mục đầu ra $2: thư mục chứa kết quả mô phỏng

- Experiment Input File (tệp đầu vào thử nghiệm): Tệp đầu vào xml có dạng như sau:



+ Heading:



Trong đó:

* id: cho phép tạo tiền tố cho các tệp đầu ra thử nghiệm với các mô phỏng lớn
* sourcePath: chứa đường dẫn tương đối hoặc tuyệt đối để đọc mô hình gaml
* finalstep: xác định số bước mô phỏng bạn muốn chạy. Là số bước để hoàn thành xong mô phỏng của hệ thống. Tùy vào giá trị của thuộc tính này mà ảnh hưởng đến quá trình mô phỏng.
* experiment: xác định thử nghiệm nào nên được chạy trên mô hình. Experiment phải tồn tại, nếu không chế độ headless sẽ không được thoát.

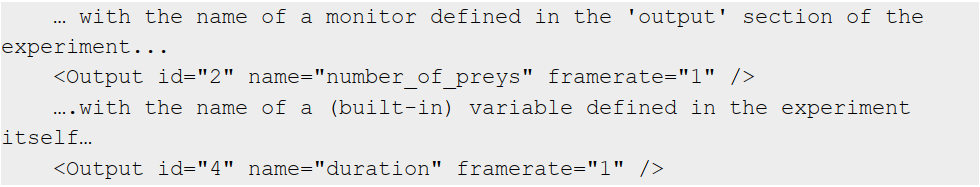
+ Parameters: Một dòng cho mỗi tham số bạn muốn chỉ định giá trị:



Trong đó:

* name: tên của tham số trong mô hình gaml
* type: loại tham số (INT, FLOAT, BOOLEAN, STRING)
* value: giá trị đã chọn

+ Outputs: Một dòng cho mỗi giá trị đầu ra mà bạn muốn truy xuất. Đầu ra có thể là tên của màn hình hoặc hiển thị đã được xác định trong phần output của thử nghiệm hoặc tên thuộc tính đã được xác định trong thử nghiệm hoặc chính mô hình (trong phần ‘global’)



Trong đó:

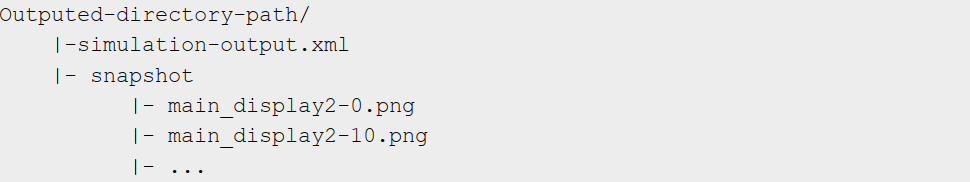
* name: tên của đầu ra trong phần “output/permanent” hoặc tên thuộc tính experiment/model để truy xuất
* framerate: tần suất giám sát (ví dụ: mỗi 1 bước, mỗi 2 bước, mỗi 100 bước,…)

Lưu ý:

* giá trị tốc độ khung hình (frame rate value) càng thấp thì thử nghiệm càng dài
* nếu đầu ra đã chọn là hiển thị, một tệp hình ảnh sẽ được tạo ra và tệp đầu ra chứa đường dẫn để truy cập hình ảnh này.



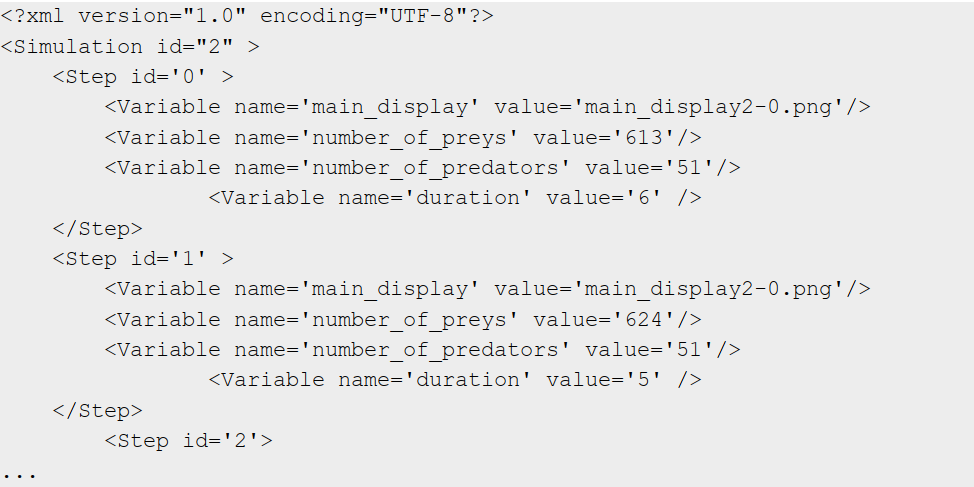
- Output Directory: Trong thử nghiệm headless, một thư mục sẽ được tạo ra có cấu trúc như sau:



Trong đó:

* simulation-output.xml: chứa các kết quả
* snapshot: chứa các ảnh chụp nhanh được tạo ra trong quá trình mô phỏng

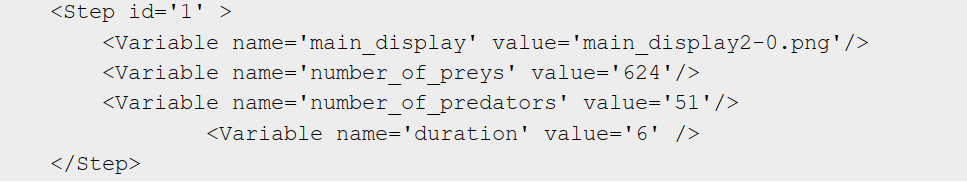
+ Simulation Output: Một tệp simulation-output.xml được tạo ra có nội dung như sau:



Trong đó:

* <Simulation id="2" >: khối chứa kết quả mô phỏng có id là 2 (id này được xác định trong Input Experiment File)
* <Step id='1' > ... </Step>: thực hiện một khối cho mỗi bước. id tương ứng với số bước.

+ Step:



có một khối Variable cho mỗi đầu ra được xác định trong Output Experiment File.

+ Variable:



Trong đó:

* name: tên đầu ra, biến mô hình
* value: giá trị hiện tại của biến mô hình

lưu ý rằng giá trị của đầu ra được lặp lại theo tốc độ khung hình được xác định trong Input Experiment File.

- Snapshot file: Thư mục này chứa các hình ảnh được tạo trong quá trình thử nghiệm. Có một hình ảnh trên mỗi đầu ra hiển thị trên mỗi bước (theo tốc độ khung hình). Tên tệp tuân theo quy ước đặt tên, ví dụ:



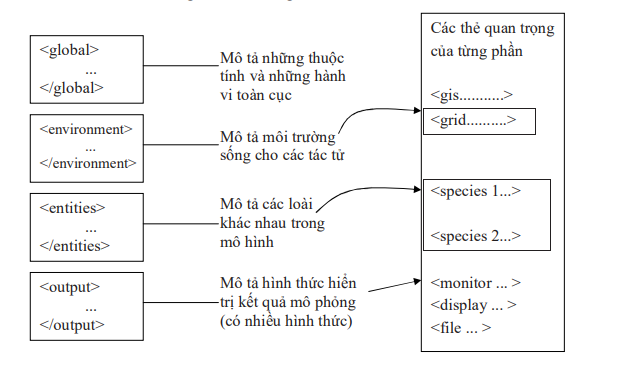
lưu ý rằng đầu ra sẽ được lưu với đuôi mở rộng là .png.

## 1.3. Ngôn ngữ GAML

GAML là ngôn ngữ được sử dụng trong hệ nền GAMA. GAML dựa trên ngôn ngữ XML. Đây là ngôn ngữ chung về mô hình hóa hệ thống đa tác tử, người ta sử dụng GAML để mô hình hóa những hành vi của tác tử. Nó cho phép các nhà mô hình hóa mô tả và thao tác trực tiếp đến hầu hết các thành phần trong meta-model của GAMA.

Trong GAML, các hành động (action) và những hành vi (behaviour) được viết thành những câu lệnh tuần tự. Những câu lệnh này sử dụng các biểu thức để định nghĩa các điều kiện, tính toán, chỉnh sửa dữ liệu,… biểu thức chứa những biến, từ khóa hoặc những phép toán. Mỗi biểu thức hay mỗi một biến thì thuộc về một kiểu (hay một loài mà chúng ta đã định nghĩa). Tập tin của GAML hiện nay có cấu trúc như tập tin XML.

Cấu trúc của một chương trình viết bằng GAMA:



*Hình 1.2. Cấu trúc của một chương trình viết bằng GAMA*

## 1.4. Mô phỏng (Simulation)

Cụm từ “mô phỏng” được sử dụng trong khoa học từ lâu trước khi thế hệ máy tính đầu tiên xuất hiện, với ý nghĩa là bắt chước một tiến trình bằng một tiến trình khác có sự điều khiển. Ngày nay, nhờ vào khả năng rất mạnh của hệ thống máy tính mà cụm từ “mô phỏng” có nghĩa là mô phỏng bằng máy tính.

Mô phỏng là một hoạt động theo một phương pháp và có một mục đích xác định, với sự trợ giúp của một phương pháp thực nghiệm (được gọi là bộ mô phỏng), người ta thực hiện một sự xáo trộn những dữ liệu đầu vào của một mô hình động, thực thi nó và nhận những dữ liệu đầu ra để hiểu được những chức năng, đặc tính của mô hình. Tất cả những mô hình được viết phải tôn trọng meta-model được liên kết với bộ mô phỏng và phải có ít nhất một tham số đầu vào để sẵn sàng thực hiện quá trình mô phỏng.

Mục đích của mô phỏng:

- Hợp thức hóa, đánh giá, kiểm chứng: Mô phỏng để kiểm tra một giả thuyết của mô hình hệ thống thực, kiểm chứng cũng như chứng minh lý thuyết đã dùng xây dựng mô hình.

- Mô phỏng cũng có mục đích chỉ ra sự giao tiếp, sự tập huấn, sự minh họa rõ ràng: Nghĩa là mô phỏng làm cho thấy rõ và chia sẻ mô hình động của hệ thống thực.

- Hiểu biết, khám phá: Mô phỏng cho phép hiểu chức năng của hệ thống thực bằng cách xem mô hình như một bản sao thu nhỏ mà chúng ta có thể nghiên cứu ñược một cách dễ dàng.

- Hỗ trợ quyết định hay là một sự điều khiển mà nó tác động đến trạng thái thực của hệ thống tham chiếu.

- Tiên đoán, dự báo những sự phát triển (thay đổi) có thể của hệ thống thực (ứng dụng trong các trò chơi).

Bộ mô phỏng (Simulator): là một chương trình tin học hay còn gọi là một hệ nền (platform) có khả năng thông dịch những mô hình động và được sử dụng để sinh ra những sự thay đổi (xáo trộn) mong muốn trên những mô hình này.

Mô hình thực thi (operational model): là mô hình động được mô tả trong một ngôn ngữ tuân theo meta-model kết hợp với bộ mô phỏng và ngôn ngữ này có thể được thông dịch trực tiếp hoặc được thực thi bởi bộ mô phỏng.

# CHƯƠNG 2: TỔNG QUAN LÝ THUYẾT LẬP TRÌNH

## 2.1. Spring Boot: Framework

Spring Boot là một dự án nổi bật trong hệ sinh thái Spring Framework. Nếu như trước đây, công đoạn khởi tạo một dự án Spring khá vất vả từ việc khai báo các dependency trong file pom.xml cho đến cấu hình bằng XML hoặc annotation phức tạp, thì giờ đây với Spring Boot, chúng ta có thể tạo các ứng dụng Spring một cách nhanh chóng và cấu hình cũng đơn giản hơn.

Dưới đây là một số ưu điểm nổi bật của Spring Boot:

+ Đóng gói ứng dụng Spring dưới dạng một file JAR (stand-alone application). Chúng ta có thể dễ dàng start ứng dụng Spring chỉ với câu lệnh quen thuộc java -jar.

+ Tối ưu công đoạn cấu hình cho ứng dụng Spring, không sinh code cấu hình và không yêu cầu phải cấu hình bằng XML.

+ Cung cấp sẵn các thư viện ‘starter’ để đơn giản hóa cấu hình Maven/Gradle

+ Cung cấp một loạt các tính năng phi chức năng phổ biến cho các dự án lớn như nhúng trực tiếp web server như Tomcat, Jetty, ... vào ứng dụng, bảo mật, health check...

+ Các ứng dụng Spring Boot là các ứng dụng web trên nền tảng Spring. Vì vậy, nó mặc định bảo mật với xác thực cơ bản trên tất cả các thiết bị endpoint HTTP. Có một tập hợp các endpoint được cung cấp sẵn để phát triển một ứng dụng Spring Boot an toàn.

## 2.2.. Hibernate: Framework

*2.2.1. Giới thiệu về Hibernate*

Hibernate là một trong những ORM Framework. Hibernate framework là một framework cho persistence layer. Hibernate giúp lập trình viên viết ứng dụng Java có thể map các object (POJO) với hệ quản trị cơ sở dữ liệu quan hệ (database), và hỗ trợ thực hiện các khái niệm lập trình hướng đối tượng với cơ sở dữ liệu quan hệ.

Hibernate giúp lưu trữ và truy vấn dữ liệu quan hệ mạnh mẽ và nhanh. Hibernate cho phép bạn truy vẫn dữ liệu thông qua Java Persistence API (JPA) hoặc bằng ngôn ngữ SQL mở rộng của Hibernate (HQL) hoặc bằng SQL thuần (Native SQL).

Hibernate vốn là một thư viện sinh ra để làm việc với mọi loại database, nó không phụ thuộc vào bạn chọn loại database nào. Nếu Javalà “Viết 1 lần chạy mọi nơi” thì Hibernate là “Viết 1 lần chạy trên mọi loại database”.

*2.2.2. Lợi ích của Hibernate Framework*

+ Mã nguồn mở và nhẹ: Hibernate Framework là mã nguồn mở có giấy phép LGPL và nhẹ.

+ Hiệu suất nhanh: Hiệu suất của Hibernate Framework là nhanh bởi vì bộ nhớ cache được sử dụng trong nội bộ Hibernate Framework.

+ Truy vấn cơ sở dữ liệu độc lập: HQL (Hibernate Query Language) là phiên bản hướng đối tượng của SQL. Nó tạo ra các truy vấn cơ sở dữ liệu độc lập. Vì vậy, bạn không cần phải viết các truy vấn cơ sở dữ liệu cụ thể.

+ Tạo bảng tự động: Hibernate framework cung cấp phương tiện để tạo ra các bảng cơ sở dữ liệu một cách tự động. Vì vậy, không cần phải tốn công sức tạo ra các bảng trong cơ sở dữ liệu thủ công.

+ Đơn giản lệnh join phức tạp: Có thể lấy dữ liệu từ nhiều bảng một cách dễ dàng với Hibernate framework.

+ Cung cấp thống kê truy vấn và trạng thái cơ sở dữ liệu: Hibernate hỗ trợ bộ nhớ cache truy vấn và cung cấp số liệu thống kê về truy vấn và trạng thái cơ sở dữ liệu.

*2.2.3. Vì sao nên dùng Hibernate thay vì JDBC*

+ Với [JDBC](https://gpcoder.com/6230-su-dung-jdbc-api-thuc-thi-cau-lenh-truy-van-du-lieu/) ta phải map các trường trong bảng với các thuộc tính của Java object một cách “thủ công”. Với Hibernate sẽ hỗ trợ map một cách “tự động” thông qua các file cấu hình map XML hay sử dụng các annotation.

+ Hibernate cung cấp các câu lệnh truy vấn tương tự SQL, HQL của Hibernate hỗ trợ đầy đủ các truy vấn đa hình như, HQL “hiểu” các khái niệm như kế thừa (inheritance), đa hình (polymorphism), và liên kết (association).

+ Code sử dụng Hibernate là độc lập với hệ quản trị cơ sở dữ liệu, nghĩa là ta không cần thay đổi câu lệnh HQL khi ta chuyển từ hệ quản trị CSDL MySQL sang Oracle, hay các hệ quản trị CSDL khác… Do đó rất dễ để ta thay đổi CSDL quan hệ, đơn giản bằng cách thay đổi thông tin cấu hình hệ quản trị CSDL trong file cấu hình.

+ Khi ta thay đổi (thêm) cột vào bảng, với JDBC ta phải thay đổi những gì:

- Thêm thuộc tính vào POJO class.

- Thay đổi method chứa câu truy vấn “select”, “insert”, “update” để bổ sung cột mới.

- Có thể có rất nhiều method, nhiều class chứa các câu truy vấn như trên.

+ Với Hibernate ta chỉ cần:

- Thêm thuộc tính vào Entity class

- Cập nhật Hibernate Annotation để map column – property.

## 2.3. Sơ lược postgreSQL

*2.3.1. Giới thiệu về PostgreSQL*

PostgreSQL là một hệ thống quản trị cơ sở dữ liệu quan hệ miễn phí và nguồn mở (RDBMS) tập trung vào khả năng mở rộng và tuân thủ các tiêu chuẩn kỹ thuật. Nó được thiết kế để xử lý một loạt các khối lượng công việc lớn, từ các máy tính cá nhân đến kho dữ liệu hoặc dịch vụ Web có nhiều người dùng đồng thời.

PostgreSQL bắt đầu từ năm 1986 như một phần của dự án POSTGRES tại Đại học California tại Berkeley. Đây là cơ sở dữ liệu mặc định cho macOS Server, và cũng có các bản phân phối cho Linux, FreeBSD, OpenBSD và Windows.

*2.3.2. Vì sao sử dụng PostgreSQL?*

- PostgreSQL sở hữu một hệ tính năng đa dạng giúp hỗ trợ các nhà phát triển xây dựng app, các nhà quản trị bảo vệ toàn vẹn dữ liệu, và tạo ra một môi trường chịu lỗi fault-tolerant giúp bạn quản lý dữ liệu bất kể tập dữ liệu lớn hay nhỏ. Bên cạnh hệ thống nguồn mở và miễn phí, PostgreSQL cũng có khả năng mở rộng tuyệt vời. Ví dụ, chúng ta có thể định nghĩa các kiểu dữ liệu riêng, xây dựng các hàm tùy chỉnh, hay viết mã từ các ngôn ngữ lập trình khác nhau mà không cần biên dịch lại cơ sở dữ liệu.

- PostgreSQL tuân theo tiêu chuẩn SQL nhưng không mâu thuẫn với các tính năng truyền thống hay có thể dẫn đến các quyết định kiến trúc gây hại. Nhiều tính năng theo tiêu chuẩn SQL được hỗ trợ, tuy nhiên đôi khi có thể có cú pháp hoặc hàm hơi khác một chút.

- PostgreSQL cung cấp cho người dùng nhiều tính năng hiện đại hỗ trợ các nhà phát triển xây dựng ứng dụng như:

+ Kiểu dữ liệu: nguyên thuỷ (các số nguyên, boolean, số, chuỗi); cấu trúc (UUID, phạm vi, mảng, ngày/giờ); Hình học (điểm, đường thẳng, đường tròn, đa giác); Tùy chỉnh (loại hỗn hợp, loại tuỳ chỉnh); Tài liệu (JSON/JSONB, XML, Key-value).

+ Toàn vẹn dữ liệu: Ràng buộc loại từ, Primary Keys, Foreign Keys, UNIQUE, NOT NULL, Khóa khuyến nghị/ Advisory Locks, Khóa hàm số/ Explicit Locks,…

+ Hiệu suất, đồng quy: Tính năng lập danh mục, lập danh mục nâng cao, trình lập kế hoạch, trình tối ưu hóa truy cập phức tạp, thống kê số liệu trên nhiều cột, quét index – only, giao tác – giao tác dạng test, điều khiển đồng thời nhiều phiên bản (MVCC), phân vùng bảng, truy vấn đọc song song, độ tin cậy, disaster recovery, nhật ký ghi trước (Write-ahead Logging – WAL), replication, khôi phục điểm – thời gian, bảng

+ Chức năng bảo mật: Bảo mật, xác thực (SCRAM-SHA-256, SSPI, LDAP, GSSAPI, Certificate và các hình thức khác), hệ thống kiểm soát truy cập mạnh mẽ, bảo mật cấp độ cột – hàng.

+ Khả năng mở rộng: phương pháp lưu trữ, ngôn ngữ thủ tục (PL / PGSQL, Python, Perl, và nhiều ngôn ngữ khác), PostGIS, tính năng kết nối cơ sở dữ liệu hoặc luồng khác với giao diện SQL chuẩn, cùng nhiều tính năng mở rộng khác.

+ Tìm kiếm văn bản: Tìm kiếm văn bản đầy đủ, hệ thống các bộ ký tự quốc tế (thông qua ICU collations).

+ Tính năng khác: Khả năng quản lý số lượng người dùng đang thao tác cùng lúc, phù hợp với môi trường sản xuất quản lý nhiều terabyte và petabyte.

*2.3.3. Vai trò của PostgreSQL*

PostgreSQL là một hệ thống quản trị dữ liệu mở dành cho các doanh nghiệp. Hệ thống quản lý này tương thích với nhiều nền tảng khác nhau, sử dụng được đa dạng ngôn ngữ và phần mềm trung gian phổ biến hiện nay. Bởi vậy, PostgreSQL được áp dụng nhiều trong các ngành dữ liệu GIS của chính phủ, tài chính ngân hàng, sản xuất – kinh doanh, công nghệ web và các công việc thu thập dữ liệu khoa học.

## 2.4. Angular: Framework

*2.4.1. Giới thiệu về Angular*

Angular là một javascript framework do google phát triển để xây dựng các Single Page Application (SPA) bằng JavaScript, HTML và TypeScript. Angular cung cấp các tính năng tích hợp cho animation, http service và có các tính năng như auto-complete, navigation, toolbar, menus,… Code được viết bằng TypeScript, biên dịch thành JavaScript và hiển thị tương tự trong trình duyệt.

*2.4.2. Ưu điểm của Angular*

Một số ưu điểm của Angular là:

- Cung cấp khả năng tạo ra các Single Page Application một cách dễ dàng

- Dễ dàng Unit test

- Cung cấp khả năng Data binding tới HTML do đó giúp người dùng cảm giác linh hoạt, thân thiện

- Cung cấp khả năng tái sử dụng các component

- Lập trình viên sẽ viết ít code hơn, với nhiều chức năng hơn.

*2.4.3. Vì sao nên chọn Angular*

+ **Angular giúp nâng cao năng suất của lập trình viên:**

Việc phát triển Web đã có bước thay đổi đáng kể trong vài năm qua. Với phiên bản ECMAScript (ES) 2015 – chúng ta quen thuộc với cái tên ES6, với những class hay arrow function. Angular 2+ ứng dụng những tính năng mới này giúp việc code với Angular trở nên rõ ràng và dễ học hơn rất nhiều.

Thêm vào đó, với việc ứng dụng Typescript – một ngôn ngữ – hay là một bản nâng cấp đáng giá của Javascript, Angular kết hợp với Typescript, chúng ta có một công cụ tuyệt vời giúp xử lý các vấn đề hạn chế của JS như kiểm tra kiểu dữ liệu, refactor code an toàn hơn,… từ đó cũng hỗ trợ tốt hơn cho việc Debug cũng như giúp các Dev thực sự hiểu rõ mã nguồn của họ hơn.

+ **Cấu trúc phát triển rõ ràng:**

Điều quan trọng của một Frameworks đối với lập trình viên đó là cấu trúc phát triển ứng dụng của nó, và Angular mang đến một kiến trúc rất rõ ràng, dựa trên ba yếu tố chính: class, các dependency được thêm vào và mô hình MVVM (model-view-view / model).

Angular sử dụng class trong ES6 với một loạt các thuộc tính để xây dựng toàn bộ các cấu trúc chủ chốt, giả sử bạn muốn tạo một Angular component – Tạo một class và thêm vào các thuộc tính cần thiết. Hay bạn muốn tạo một Angular module – Hãy tạo một class và thêm vào đó các thuộc tính cần thiết. Về cơ bản sẽ là như vậy, Angular cung cấp một cấu trúc rõ ràng để xây dựng từng tính năng cho ứng dụng của bạn.

Các dependency mạnh mẽ được sử dụng trong ứng dụng khi cần thiết, và khi cần tích hợp bất kì dependency nào, như HTTP hay Router, chúng ta chỉ cần thêm nó vào bên trong constructor của class.

Mô hình MVVM cũng giúp Angular chiếm lợi thế trong xây dựng ứng dụng client-side, thường ta sẽ có 3 điều cần quan tâm chính: đó là giao diện người dùng, mã nguồn điều khiển giao diện và mô hình dữ liệu (data) cho giao diện. Angular với MVVM phân biệt hoàn toàn rõ ràng các yếu tố trên nhờ mô hình MVVM:

Phần giao diện (view) được định nghĩa trong một template bao hàm HTML dành cho một component nhất định. Template có thể là toàn bộ Layout hoặc bất cứ mảnh ghép nào trong Layout đó.

Model được định nghĩa như là các thuộc tính của component class. Có thể hiểu là dữ liệu, dựa vào đó để phần View sử dụng để thực thi.

View/model là class quản lý cả view cũng như model. Là phần code sẽ xử lý việc truy xuất dữ liệu, đồng thời thực thi các tương tác của người dùng trên view.

Với việc ứng dụng các điểm tích cực của các thành phần trên, Angular khiến việc phát triển ứng dụng trở nên dễ dàng và hiệu quả hơn.

+ **Extensive binding:**

Rất nhiều ứng dụng Web làm việc với dữ liệu (data). App sẽ truy xuất dữ liệu từ Server và hiển thị dữ liệu đó tới người dùng trên view, sử dụng template. Và các tương tác của người dùng sẽ được khiến dữ liệu thay đổi, được view ghi nhận và lưu lại trên server. Data Binding trong Angular giúp bạn thực thi tiến trình trên rất dễ dàng. Đơn thuần từ việc ràng buộc thành phần HTML trong template với các thuộc tính trong class và dữ liệu sẽ tự động xuất hiện trên màn hình. Với các tương tác của người dùng đòi hỏi thay đổi dữ liệu, Angular sử dụng phương pháp two-way binding. Bất kì thay đổi dữ liệu đến từ view sẽ tự động cập nhật thuộc tính “model” bên trong class.

Thêm vào đó, Angular cũng hỗ trợ property binding – cho phép chúng ta điều khiển DOM bằng cách ràng buộc thuộc tính HTML với thuộc tính của component class, data sẽ tự động xuất hiện bên trong view. Ví dụ, chúng ta ràng buộc thuộc tính hidden đối với một thẻ img với thuộc tính hideImg bên trong class. Khi thuộc tính hideImg nhận giá trị true, img sẽ tự động hidden và ngược lại khi hideImg nhận giá trị false, thẻ img sẽ tự động hiển thị tới người dùng.

Cuối cùng, Angular hỗ trợ event binding, có nghĩa là chúng ta có thể xử lí bất kì event nào từ phía view, như HTML event. Về cơ bản chúng ta sẽ gắt event với một method bên trong class. Mỗi khi event xuất hiện, method tương ứng sẽ được thực thi.

Extensive binding giúp quá trình hiển thị dữ liệu, điều khiển DOM, thực thi các event một cách trơn tru và dễ dàng.

+ **Hỗ trợ đầy đủ tính năng điều hướng (routing):**

Đa số các ứng dụng Web không chỉ có 1 view hay một page duy nhất, mà sẽ cung cấp nhiều view khác nhau tương ứng với với các chức năng chính. Ví dụ như một trang web với các trang giới thiệu, trang nội dung, trang chi tiết, trang đăng nhập, đăng ký,… Chúng ta sẽ cần hiển thị đúng view vào đúng thời điểm. Đó là mục đích của điều hướng (routing). Và Angular cung cấp đầy đủ tính năng cho việc này, chúng ta định nghĩa các đường dẫn (route) cho mỗi page view của ứng dụng. Và chúng ta sẽ kích hoạt route dựa trên tương tác của người dùng (user). Chúng ta có thể truyền thêm dữ liệu vào các route, giúp view hiển thị nội dung một cách dynamic, có thể bảo vệ route để người dùng chỉ có thể truy cập sau khi đã đăng nhập hoặc có quyền truy cập, có thể ngăn chặn việc người dùng ngay lập tức rời một trang khi các thao tác còn dang dở cho đến khi họ thực sự xác nhận việc rời đi hoặc lưu lại tiến trình sử dụng,…Angular đồng thời cũng hỗ trợ child-route cho việc điều hướng bên trong một route. Việc điều hướng giữa các view bên trong ứng dụng Angular thực sự rất linh hoạt và mạnh mẽ.

**+ Angular giúp giảm tối đa kích thước và tăng tối đa hiệu suất của ứng dụng:**

Kích thước và hiệu năng có mối liên quan mật thiết khi chúng ta làm việc trên nền tảng Web. Một component nhỏ hơn sẽ giúp nâng cao hiệu suất khởi động – giảm cả thời gian download cũng như thời gian cũng như thời gian compile trên trình duyệt. Giảm kích thước component và giúp tăng hiệu suất là một ưu điểm cũng như mục tiêu mà Angular mong muốn mang đến cho các lập trình viên.

Giảm kích thước ứng dụng có thể thực hiện bằng nhiều cách. Đầu tiên chúng ta có thể giảm tối đa kích thước của từng component tới mức tối thiểu có thể. Tiếp theo các component sẽ được sắp xếp bên trong Angular Module bằng 1 cách đề cho các nhóm logic có liên quan đến nhau sẽ được download cùng với nhau. Và bước thứ ba, lazy loading bên trong các route sẽ chỉ download những module cần thiết cho việc hiển thị nội dung cần thiết tới người dùng, và sẽ không bao giờ download những nội dung không cần thiết.

Chúng ta có một trình biên dịch tên là AOT, trình biên dịch này sẽ chạy một lần trong thời gian build ứng dụng. Trình duyệt sau đó sẽ download phiên bản chưa được biên dịch của ứng dụng và render ứng dụng tới người dùng ngay lập tức mà không cần biên dịch nó lần đầu trong trình duyệt. Thêm nữa là sẽ không cần download trình biên dịch Angular, giúp làm giảm đáng kể kích thước (size) của ứng dụng cần tải về.

+ **Document và cộng đồng (community):**

Document cho Angular 2+ – angular.io – rất đầy đủ và chi tiết, bao hàm giới thiệu cơ bản giúp bạn làm quen nhanh chóng với Angular, giới thiệu chi tiết, từ cơ bản đến nâng cao các API của Angular, cũng như có hẳn một Tutorial Basic được xây dựng nên bởi Angular team, cung cấp cho bạn nhanh chóng nắm bắt các thuộc tính cơ bản của Framework.

## 2.5. Các phần mềm hỗ trợ

*2.5.1. Github*

GitHub là dịch vụ server quản lý nhiều phiên bản code, giúp lưu trữ chúng và cả nội dung dự án của hai hay nhiều tài khoản của lập trình viên.

Với khả năng lưu trữ và độ bảo mật cao của mình, GitHub được xem là nguồn lưu dữ liệu mở phổ biến nhất hiện nay. GitHub là sự kết hợp giữa 2 từ:

-         Git – hệ thống quản lý dự án và phiên bản code

-         Hub – nơi biến những dòng lệnh trên Git thành mạng xã hội cho lập trình viên

GitHub được sử dụng chủ yếu cho dự án có nhiều người cùng hợp tác và cần giám sát toàn bộ thay đổi của dự án. Bên cạnh đó, GitHub còn có khả năng khôi phục code khi cần thiết. Khi sử dụng GitHub, ngoài các công việc chính như tạo Branch, tạo Pull Request và Fork một Repository, bạn có thể theo dõi, tương tác với người khác như một mạng xã hội thông thường.

*2.5.2. pgAdmin 4*

PgAdmin là một công cụ quản lý PostgreSQL Database cung cấp nhiều tính năng như kết nối tới PostgreSQL, thư thi SQL Query, backup và khôi phục Database.

*2.5.3. Eclipse*

Eclipse là công cụ hỗ trợ lập trình mã nguồn mở được phát triển bởi IBM. Eclipse như một môi trường phát triển Java tích hợp (IDE), với Eclipse chúng ta có thể mở rộng hơn mã nguồn bằng cách chèn thêm các plugins cho project. Eclipse còn hỗ trợ cho lập trình viên code theo các mô hình phát triển như MVC, tạo thêm các libraryhỗ trợ phát triển phần mềm

# CHƯƠNG 3: THIẾT KẾ HỆ THỐNG VÀ CƠ SỞ DỮ LIỆU

## 3.1. Thiết kế kiến trúc

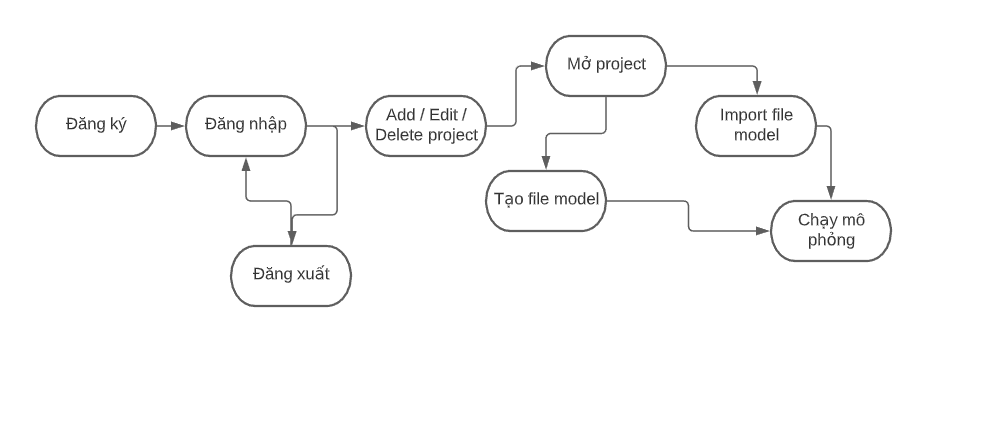
Về kiến trúc của hệ thống, chúng tôi sử dựng mô hình Client – Server để thiết kế hệ thống.

Trong mô hình client server thì máy khách là máy tính của người dùng, máy khách client gửi yêu cầu đến máy chủ server. Máy chủ server tiếp nhận yêu cầu, xử lý các yêu cầu đó và trả về kết quả.

Các thông tin sẽ được server lưu vào trong database.

Phần mềm cho phép người dùng quản lý các hệ thống mô phỏng bằng cách thao tác chỉnh sửa trên giao diện.

***Client:***



Hình 3. Hoạt động ở phía client

Ở phía client:

- Trường hợp người dùng chưa có tài khoản, có thể đăng ký thông tin tài khoản trên màn hình sign up

- Khi người dùng đã có tài khoản, tiến hành đăng nhập với tài khoản cá nhân để vào hệ thống.

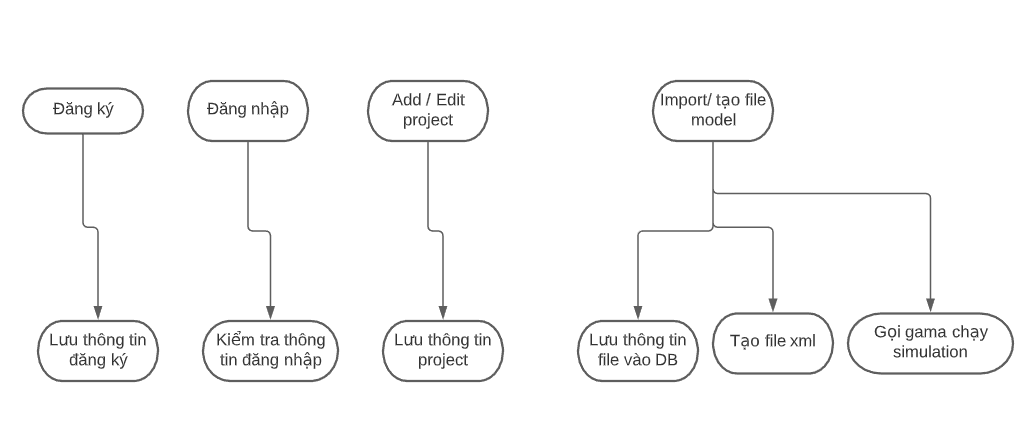
- Ở màn hình trang chủ, người dùng có thể tạo hoặc chỉnh sửa project bằng cách nhập thông tin project trên dialog. Click Save để lưu, hoặc Cancel để hủy thao tác.

- Người dùng cũng có thể xóa project khi có sai sót hoặc muốn xóa bỏ.

- Người dùng sau khi mở một project bất kỳ, có thể tiến hành import một file model hoặc tạo mới file, có thể chỉnh sửa thông tin các tham số đầu vào trên giao diện.

- Cuối cùng người dùng có thể chạy hoặc dừng mô phỏng theo ý muốn.

***Server:***



Hình 3. Hoạt động ở phía server

Ở phía server:

- Khi người dùng đăng ký thông tin tài khoản, server xử lý lưu thông tin đăng ký vào database.

- Khi người dùng đăng nhập vào hệ thống, server tiến hành kiểm tra thông tin người dùng trên database, nếu trùng khớp sẽ cho login thành công và vào màn hình trang chủ. Nếu sai thông tin đăng nhập sẽ báo lỗi và ở lại màn hình login

- Ở màn hình trang chủ, khi người dùng thêm mới hoặc chỉnh sửa thông tin một project, server sẽ lưu thông tin vào trong database. Trường hợp người dùng xóa project thì server sẽ tiến hành xóa một bản ghi tương ứng trong database

- Ở màn hình mô phỏng, khi người dùng import file model, server sẽ lưu thông tin file model vào database, bên cạnh đó, server xử lý tạo một file xml tương ứng, sau đó tiến hành gọi gama chạy simulation ở chế độ headless.

- Kết quả server trả về cho clien một chuỗi ảnh mô phỏng (số lượng ảnh phụ thuộc vào số bước mô phỏng).

Công nghệ sử dụng trong chương trình:

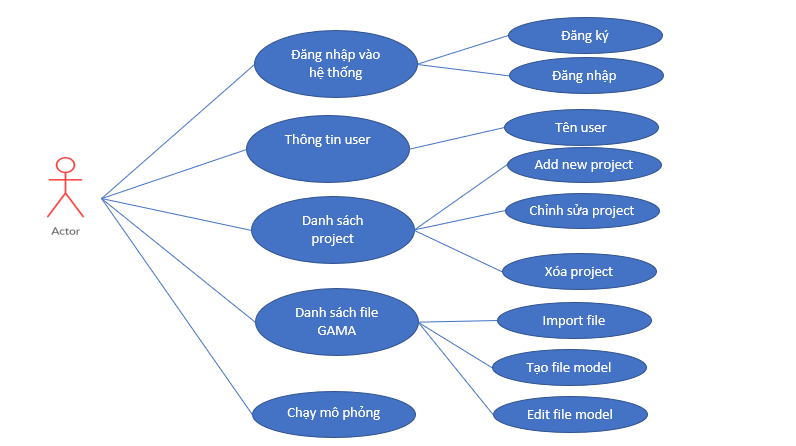
+ Front-end: Sử dụng framework Angular 8

+ Back-end: Springboot & Hibernate

+ Database: PostgreSQL

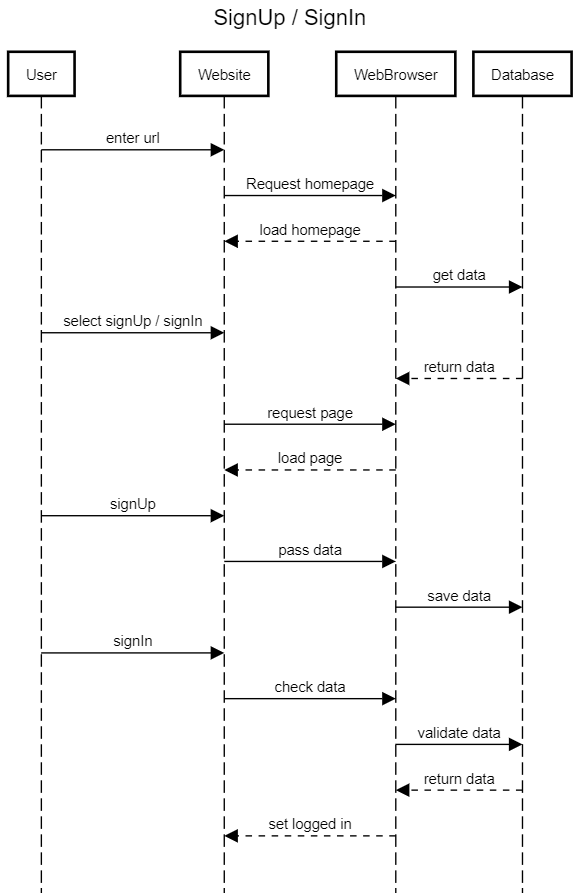
+ Simulation Engine: GAMA platform

## 3.2. Biểu đồ usecase



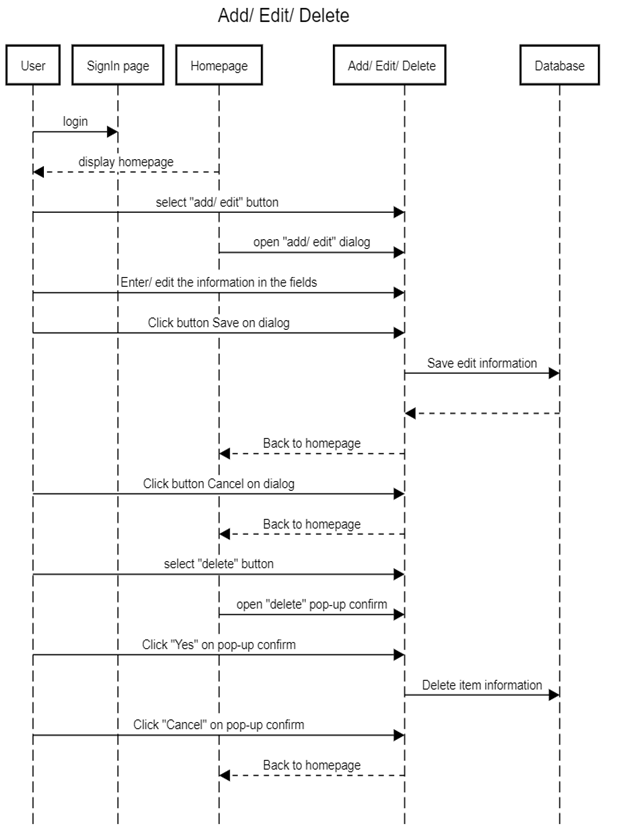
Hình 3. Biểu đồ usecase

## 3.3. Sequence diagram

3.3.1. Đăng ký, đăng nhập

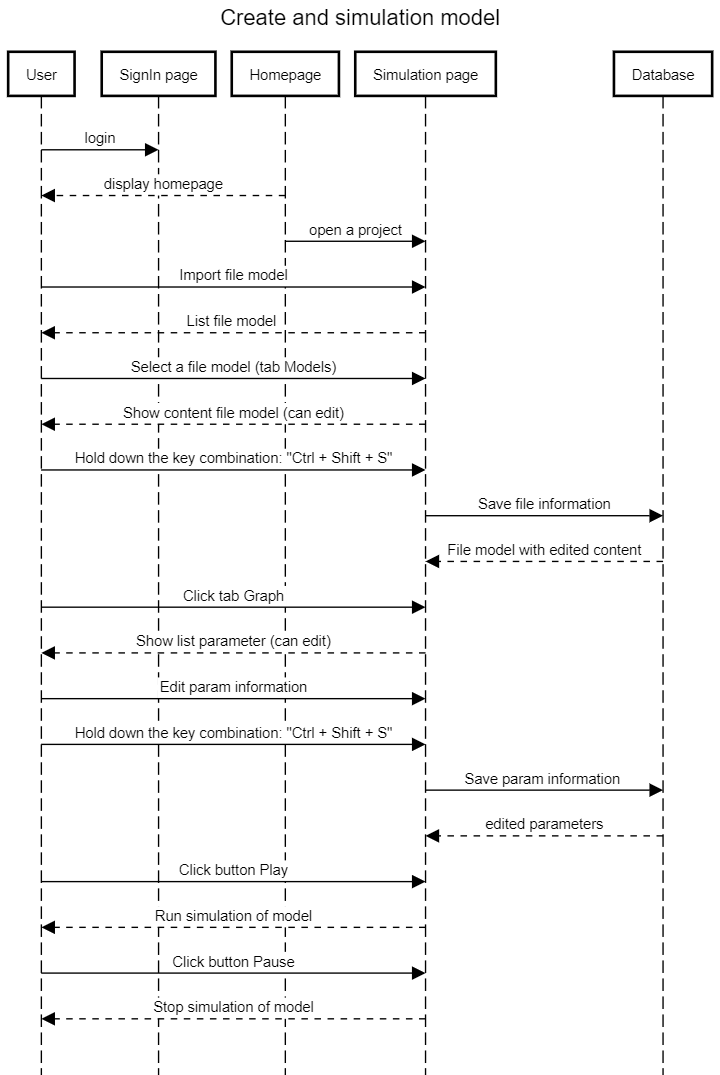
Hình 3. Sequence diagram sign in/sign up

3.3.2. Thêm mới, chỉnh sửa, xóa project



Hình 3. Sequence diagram add/edit/delete project

3.3.3. Tạo file model và chạy mô phỏng

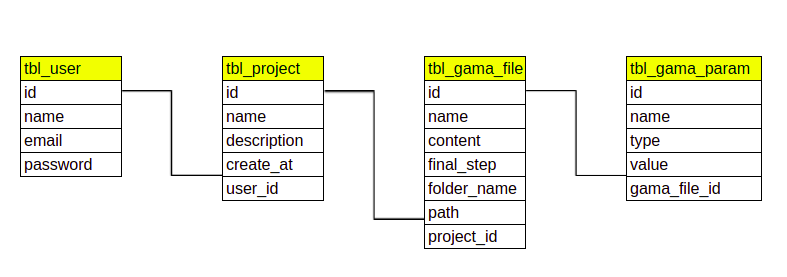


Hình 3. Sequence diagram run simulation model

## 3.4. Thiết kế cơ sở dữ liệu

Mỗi người dùng đều có thông tin cá nhân và danh sách project. Mỗi project cũng lưu thông tin project và quản lý các file mô hình theo từng chủ đề riêng. Mỗi project gồm nhiều đối tượng gama file chứa thông tin liên quan đến các thông số đầu vào cần thiết cho bài toán. Mỗi file mô hình có thể có danh sách parameter hoặc không tùy theo output của từng mô hình.

Chính vì vậy, để đáp ứng yêu cầu bài toán đặt ra tôi đã xây dựng cấu trúc cơ sở dữ liệu như sau:

****

Hình 3. Lược đồ cơ sở dữ liệu

**- Bảng User:**

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Mô tả** |
| id | integer | Id của người dùng |
| name | varchar | Tên người dùng |
| email | varchar | Email người dùng |
| password | varchar | Mật khẩu người dùng |

**- Bảng Project:**

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Mô tả** |
| id | integer | Id của project |
| name | varchar | Tên project |
| description | varchar | Mô tả về project |
| create\_at | varchar | Ngày tạo project |
| user\_id | integer | Id của người dùng |

**- Bảng GAMA file:**

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Mô tả** |
| id | integer | Id của GAMA file |
| name | varchar | Tên file model |
| content | text | Nội dung file model |
| final\_step | integer | Số bước mô phỏng |
| folder\_name | varchar | Tên của folder chứa file GAMA |
| path | varchar | Đường dẫn đến file GAMA |
| project\_id | integer | Id của project |

**- Bảng GAMA param:**

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Kiểu dữ liệu** | **Mô tả** |
| id | integer | Id của GAMA parameter |
| gama\_file\_id | integer | Id của file GAMA |
| name | varchar | Tên của parameter |
| type | varchar | Kiểu parameter |
| value | varchar | Giá trị của parameter |

# CHƯƠNG 4: TRIỂN KHAI CHƯƠNG TRÌNH VÀ KẾT QUẢ

## 4.1. Xác định hệ thống

*4.1.1. Xác định yêu cầu*

- Yêu cầu đối với hệ thống:

* + Đăng nhập và Đăng xuất: Người dùng sử dụng chức năng này để có thể vào hệ thống và sử dụng các chức năng của chương trình.
  + Thêm, sửa, xóa: Người dùng sử dụng chức năng này để có thể thêm sửa xóa các đối tượng quản lý của chương trình (quản lý project, quản lý file model)
  + Tra cứu hay Tìm kiếm: Chương trình hỗ trợ chức năng này cho người dùng tìm kiếm project.
  + Chạy mô phỏng

*4.1.2. Phân tích yêu cầu*

* + Đăng nhập và Đăng xuất: Khi muốn sử dụng một chức năng nào đó của hệ thống, người dùng tạo một tài khoản đăng nhập cho mình .Sau đó sử dụng tài khoản có Username và Password để đăng nhập vào hệ thống.
  + Thêm, sửa, xóa: Khi tạo mới project phục vụ việc xây dựng và mô phỏng hệ thống đa tác tử nào đó, người dùng phải lưu trữ thông tin project, file model của từng hệ thống, hay xóa bỏ projet, file model ra khỏi hệ thống, hay sửa đổi một vài thông tin của các đối tượng quản lý khi có sự nhầm lẫn hoặc sai sót.
  + Tra cứu hay Tìm kiếm: Khi tra cứu thông tin về projet, các file model sẽ sử dụng để phát triển chức năng của hệ thống mô phỏng.
  + Chạy mô phỏng: Mục đích của việc chạy mô phỏng có thể cho thấy được một cách trực quan về các vấn đề được xây dựng và phát triển dựa trên nền tảng GAMA.

*4.1.3. Các chức năng của hệ thống*

**Quản lý project:**

Hình 4. Các chức năng quản lý project

Quản lý project

Xóa project

Thêm mới project

Edit thông tin project

Tìm kiếm project

- Thêm mới project: nhập các thông tin có liên quan đến project khi người dùng có nhu cầu thêm một project để quản lý file model nào đó

- Sửa thông tin project: chức năng dùng để thay đổi thông tin về một project khi có sự nhầm lẫn hoặc sai sót

- Xóa project: Có chức năng dùng để loại bỏ project khi người dùng muốn xóa

- Tìm kiếm project: Tìm kiếm theo tên project

- Sắp xếp project: Sắp xếp theo tăng dần/ giảm dần tên project

- Hiển thị danh sách project dạng List/ Grid

**Quản lý file model:**

Quản lý file model

Xóa file

Tạo mới file .gaml

Import file có sẵn

Tìm kiếm file

Edit file

Hình 4. Các chức năng trong quản lý file model

- Tạo mới file: Tạo mới file gaml khi người dùng có nhu cầu mô phỏng hệ thống đa tác tử theo mong muốn của mình trên giao diện hệ thống.

- Import file có sẵn: Import file gaml có sẵn (có thể là các file sample) hoặc các file tự tạo trước đó.

- Edit file: Cho phép người dùng thay đổi nội dung file model để cho ra mô phỏng theo mong muốn.

- Xóa file: Có chức năng dùng để loại bỏ file khi người dùng muốn xóa

- Tìm kiếm file: Tìm kiếm theo tên file model

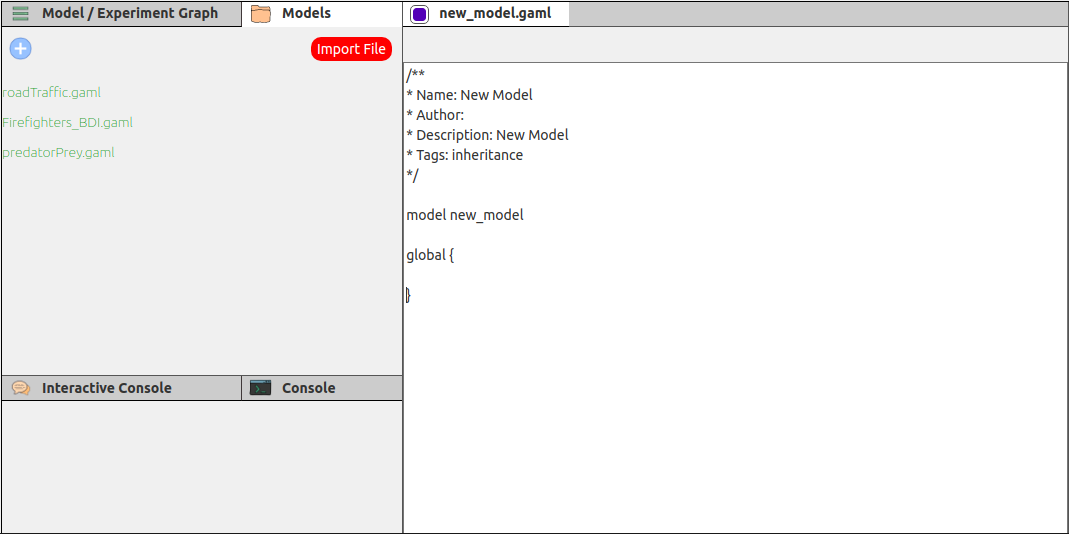
## 4.2. Giải pháp

Đầu vào của bài toán là file mô hình, các giá trị parameter, số bước mô phỏng và các output cần thiết cho quá trình mô phỏng.

Đầu tiên là file mô hình, có hai cách để tạo file mô hình (.gaml):

Cách 1: Chuẩn bị sẵn file mô hình (có thể là file sample, hoặc file mô hình tự tạo). Sau đó, tiến hành import file mô hình vào project tương ứng.

Cách 2: Click button (+) trên giao diện để tạo mới file mô hình default new\_model.gaml và triển khai các dòng code trên giao diện đó.



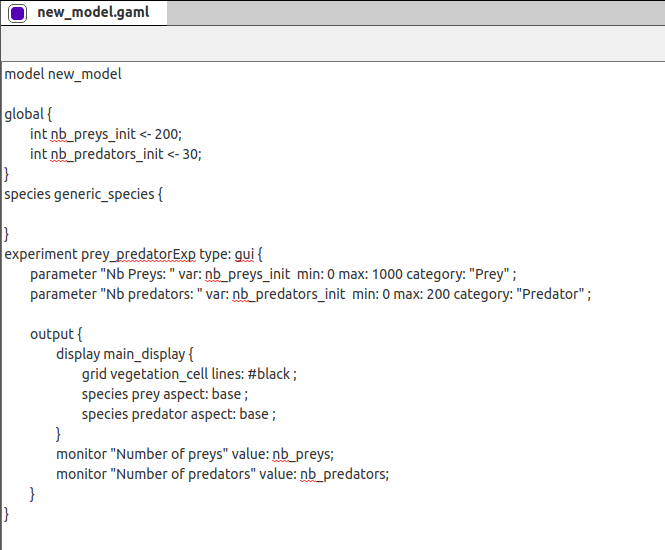
Hình 4. Các cách tạo file mô hình

Cấu trúc mô hình của một file model (.gaml) bao gồm:

**global**: đại diện cho tất cả những gì mang tính toàn cục đối với mô hình: động lực, biến số, hành động. Ngoài ra, nó cho phép khởi tạo khối mô phỏng.

**species và grid**: các phần này xác định các loại tác tử tạo nên mô hình.

**experiment**: các phần này xác định bối cảnh thực thi của các mô phỏng. Đặc biệt, nó xác định đầu vào (tham số) và đầu ra (hiển thị, tệp ...) của một mô hình.



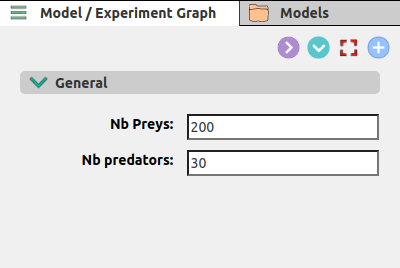
Hình 4. Giao diện chỉnh sửa file mô hình

Có thể chỉnh sửa nội dung file (các thông số đầu vào, đầu ra,…) trực tiếp trên file, thay đổi số bước mô phỏng bằng cách thanh cuộn ngang trên giao diện



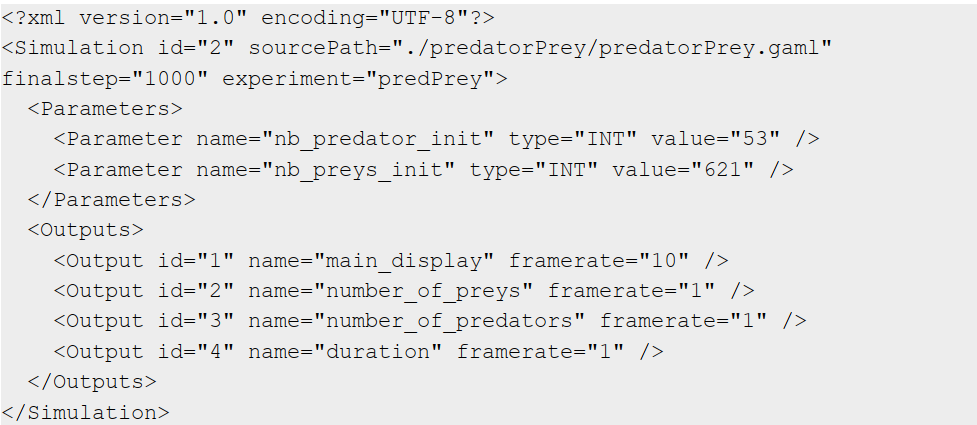
Hình 4. Giao diện chỉnh sửa số bước mô phỏng

Thay đổi giá trị các parameter bằng cách nhập giá trị của các parameter trên giao diện:



Hình 4. Giao diện thay đổi các giá trị parameter

Sau đó thực hiện lưu thay đổi và thực hiện generator file có đuôi mở rộng ".xml" có dạng: {{file name}}.xml. File .xml có dạng như sau:



Hình 4. Ví dụ về file có đuôi mở rộng .xml

Khi thực hiện generator file .xml cần chú ý các tham số sourcePath, experiment, các parameter, các output tương ứng với từng file model. Các thông số ghi ra file .xml cần chính xác để tránh gây ra lỗi khi chạy mô phỏng mà cho ra kết quả sai sót. Nội dung file .xml có thể thay đổi tùy theo giá trị tương ứng sau mỗi lần thao tác chỉnh sửa trên giao diện.

Cùng với việc thực hiện lưu chỉnh sửa các thông số đầu vào lên database, chương trình sẽ tiến hành chạy song song dòng lệnh trên terminal:

bash gama-headless.sh $1 $2

Trong đó: $1 là đường dẫn tương đối hoặc tuyệt đối đến thư mục chứa file model

$2 là đường dẫn tương đối hoặc tuyệt đối đến thư mục chứa kết quả đầu ra.

Click button Play/Pause trên giao diện để hiển thị mô phỏng ở dạng lưới, dạng biểu đồ, … Có thể dừng quá trình mô phỏng để chỉnh sửa các giá trị đầu vào.

## 4.3. Quá trình cài đặt hệ thống

- Ở Frontend:

+ Dựng source code bao gồm 3 giao diện chính: giao diện đăng nhập vào hệ thống,giao diện quản lý project, giao diện quản lý và mô phỏng mô hình.

+ Gọi api đã dựng ở backend để xử lý các logic đăng nhập hệ thống, thêm, sửa, xóa các đối tượng

+ Đọc file model sau khi import

+ Sau khi backend cho ra output cần thiết, xử lý hiển thị ảnh lên theo các output tương ứng.

- Ở Backend:

+ Sử dụng hibernate framework để truy vấn dữ liệu thông qua Java Persistence API và tạo bảng tự động.

+ Dựng các api phục vụ cho mục đích thêm, sửa, xóa các đối tượng

+ Xử lý ghi file gaml sau khi lưu thay đổi ở front-end

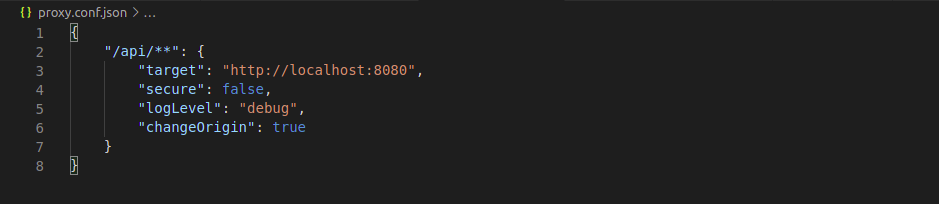
+ Xử lý tạo file xml theo các thông số đầu vào được lấy ra ở front-end

+ Xử lý chạy lệnh ở terminal để cho ra thư mục output: trong đó gồm thư mục snapshot chứa các file ảnh, file console output và file simulation output.

- Quá trình build code

+ Frontend: (port: 4200)

Để frondend có thể gọi api ở backend với port khác nhau: tạo thêm file cấu hình: proxy.conf.json



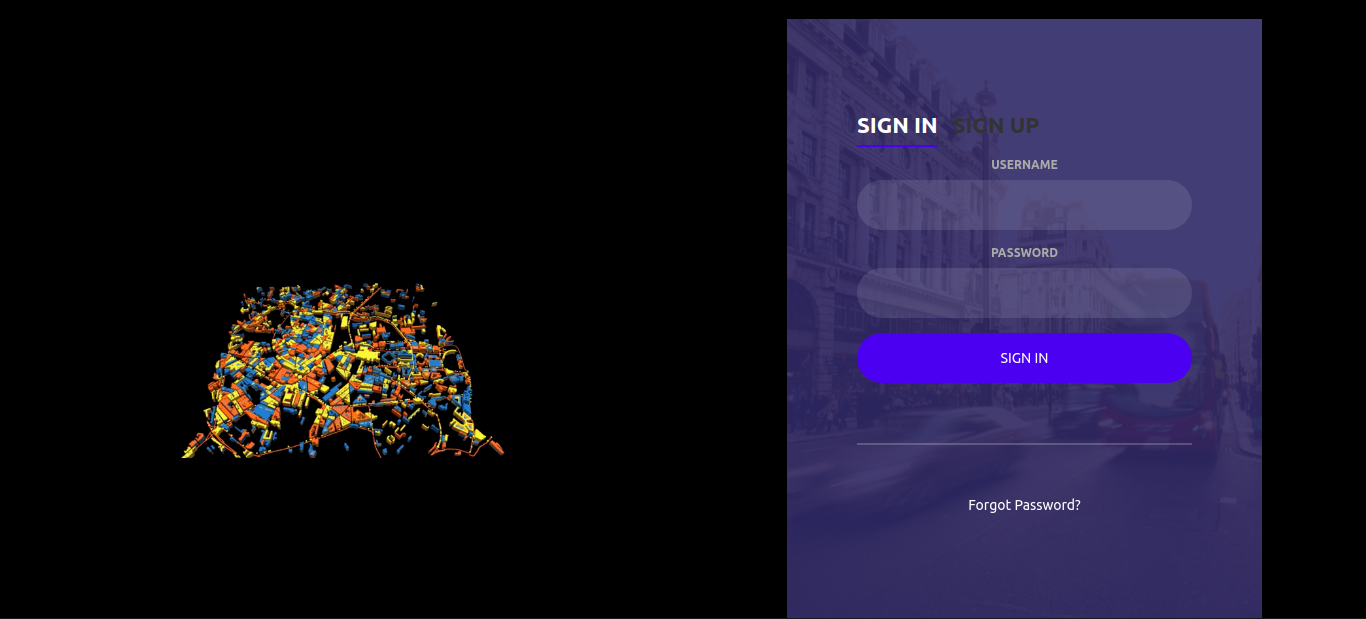
Hình 4. File cấu hình proxy.conf.json

Mở terminal, đến đường dẫn chứa thư mục source code, chạy lệnh: npm start

+ Backend: (port: 8080) Chạy chương trình trên eclipse

- Link source code: https://github.com/HaVu279/GamaNew

## 3.4. Giao diện chương trình

*3.4.1. Đăng ký, đăng nhập*

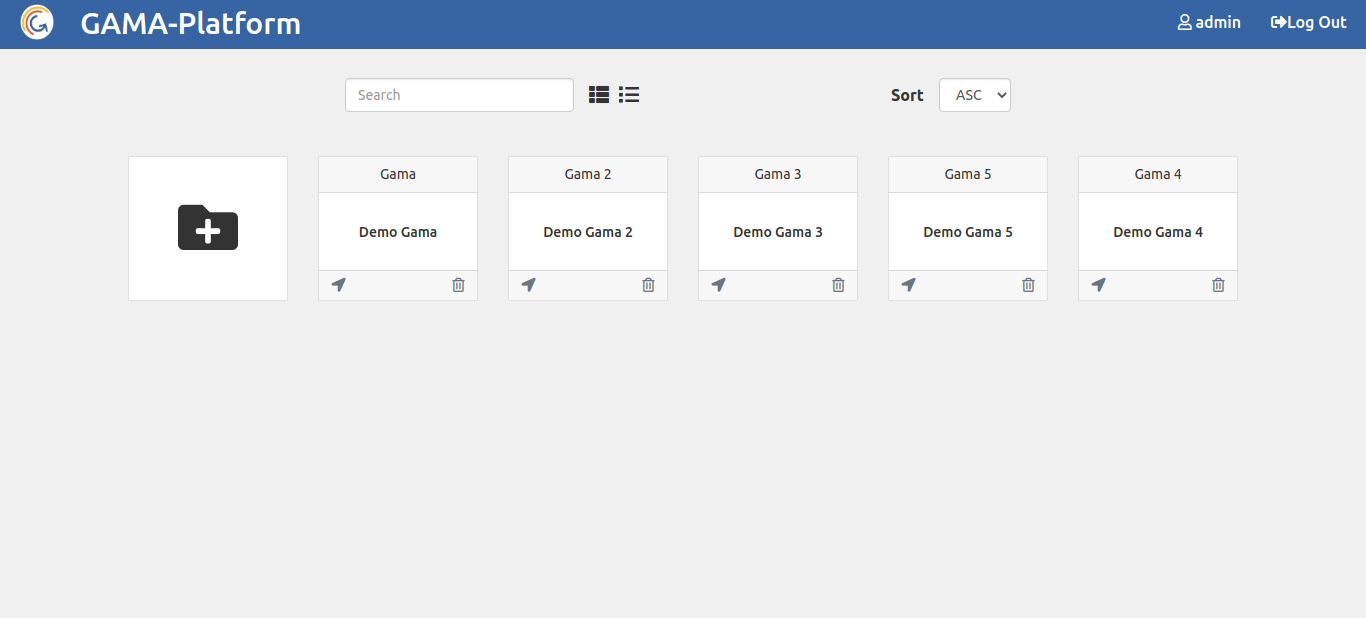
Hình 4. Màn hình Đăng nhập

**-** Đăng nhập:

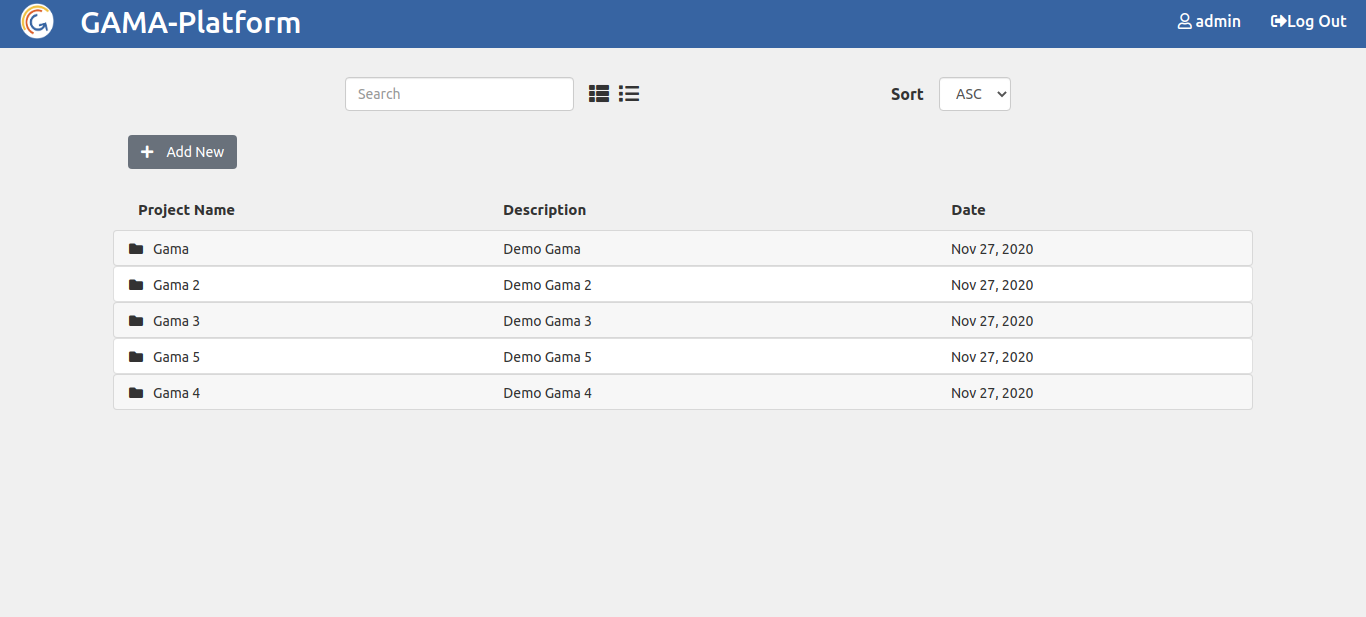
- Đăng ký:

Hình 4. Màn hình Đăng ký

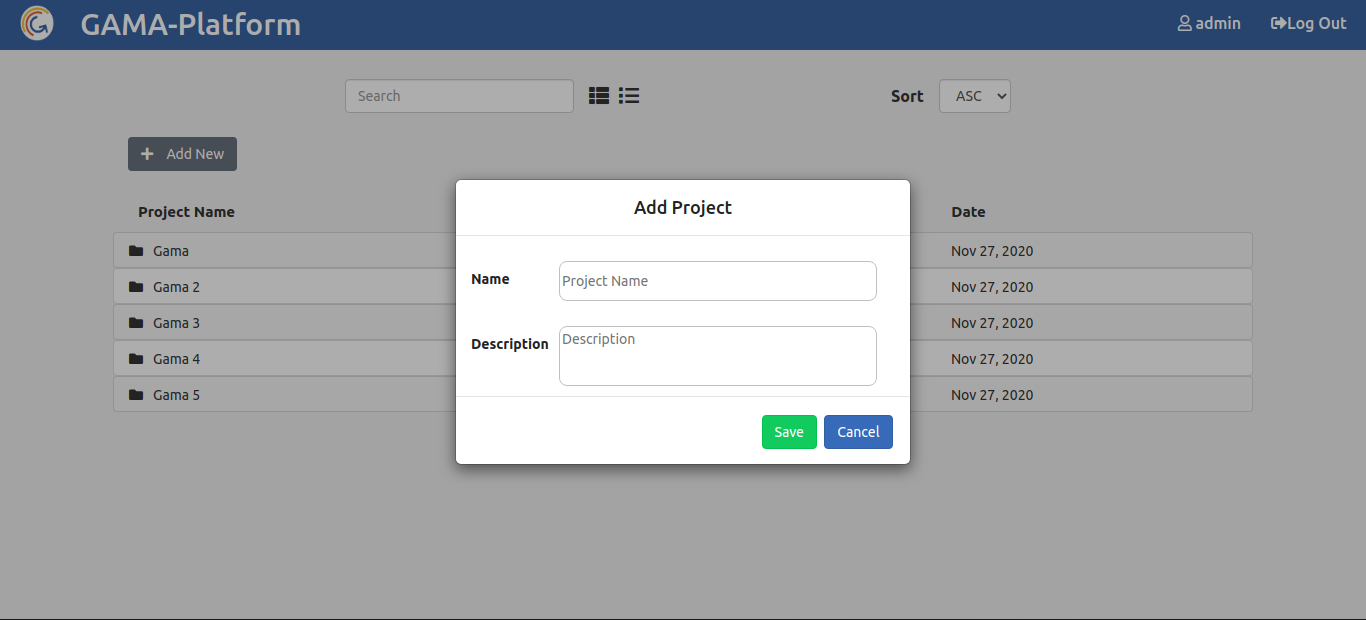
*3.2.2. Giao diện màn hình project*

- Màn hình hiển thị danh sách project theo từng user (dạng folder/ list)

Hình 4. Màn hình Project (dạng folder)

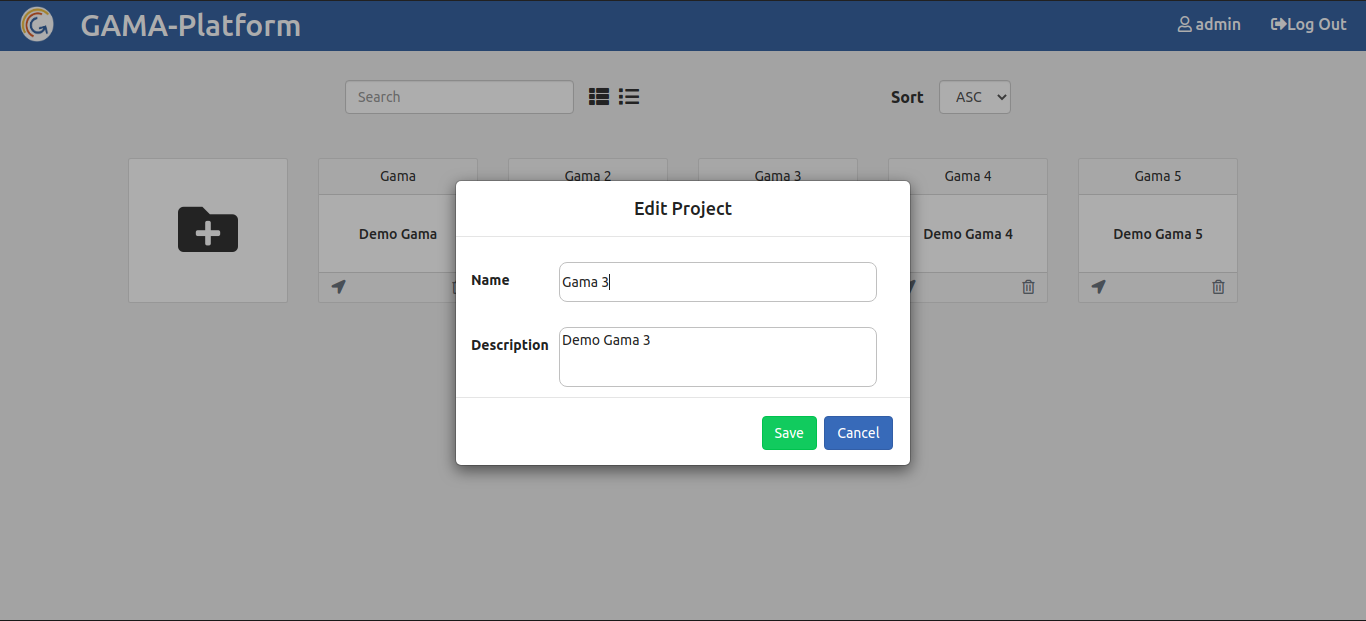


Hình 4. Màn hình Project (dạng list)



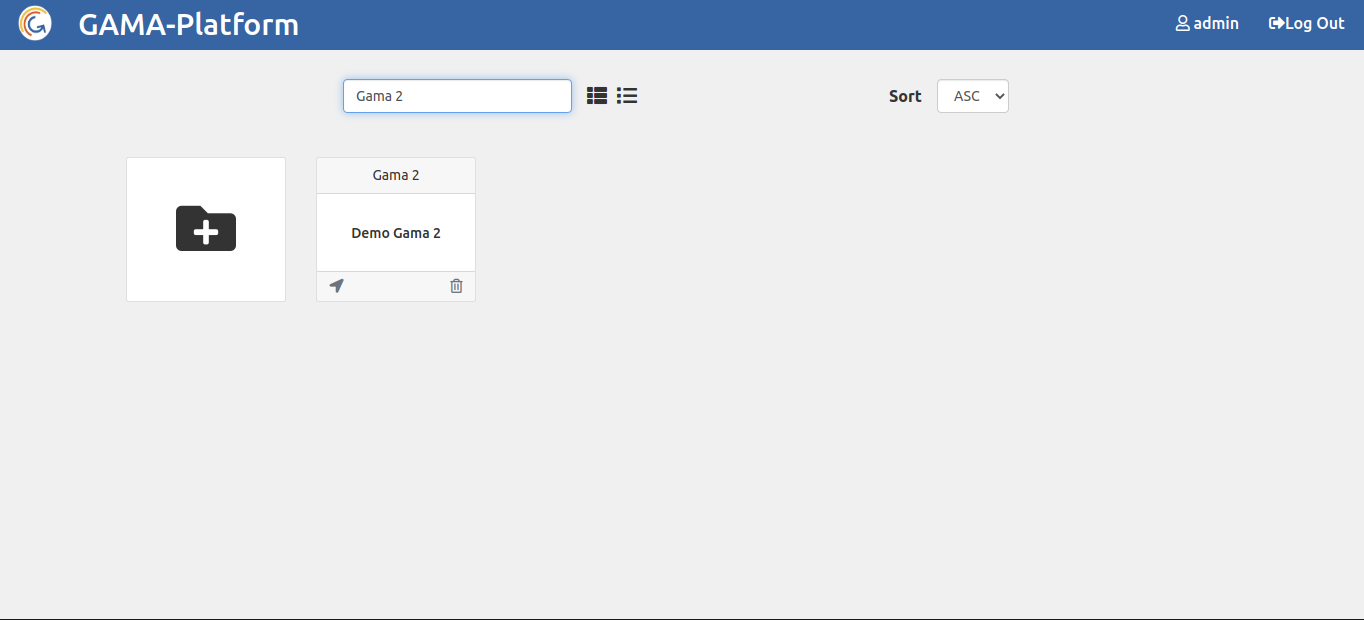
Hình 4. Dialog thêm mới project

- Thêm mới project

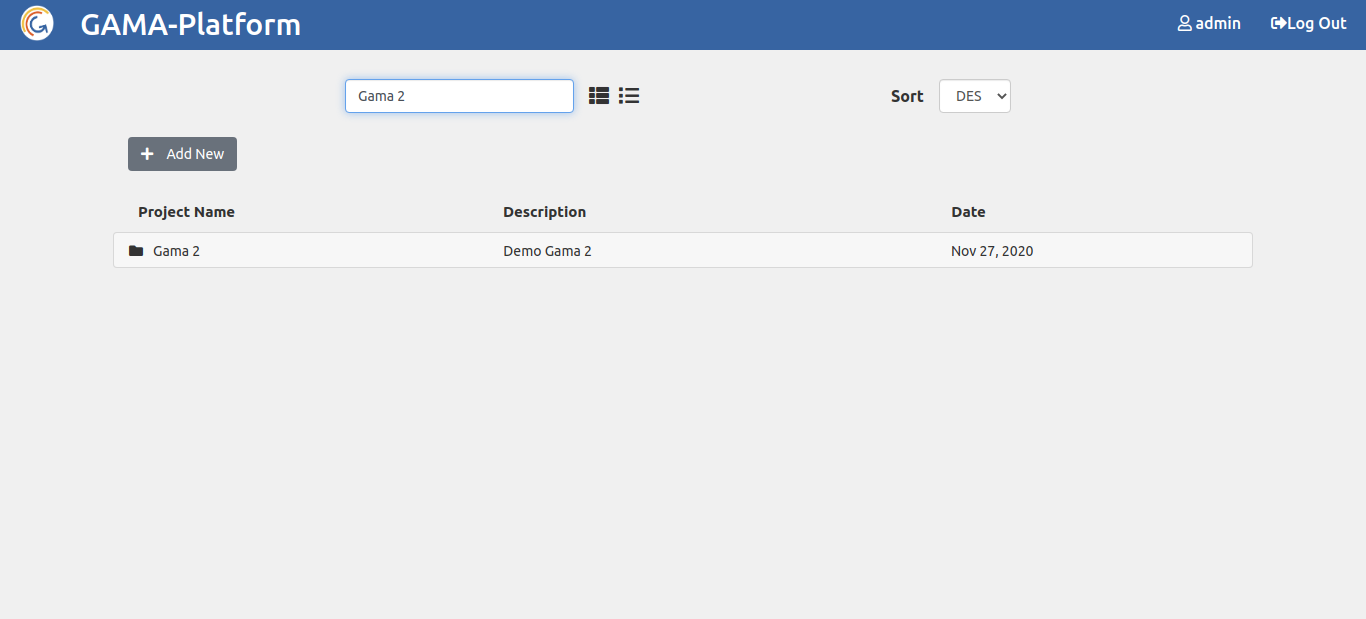
- Edit project

Hình 4. Dialog edit thông tin project

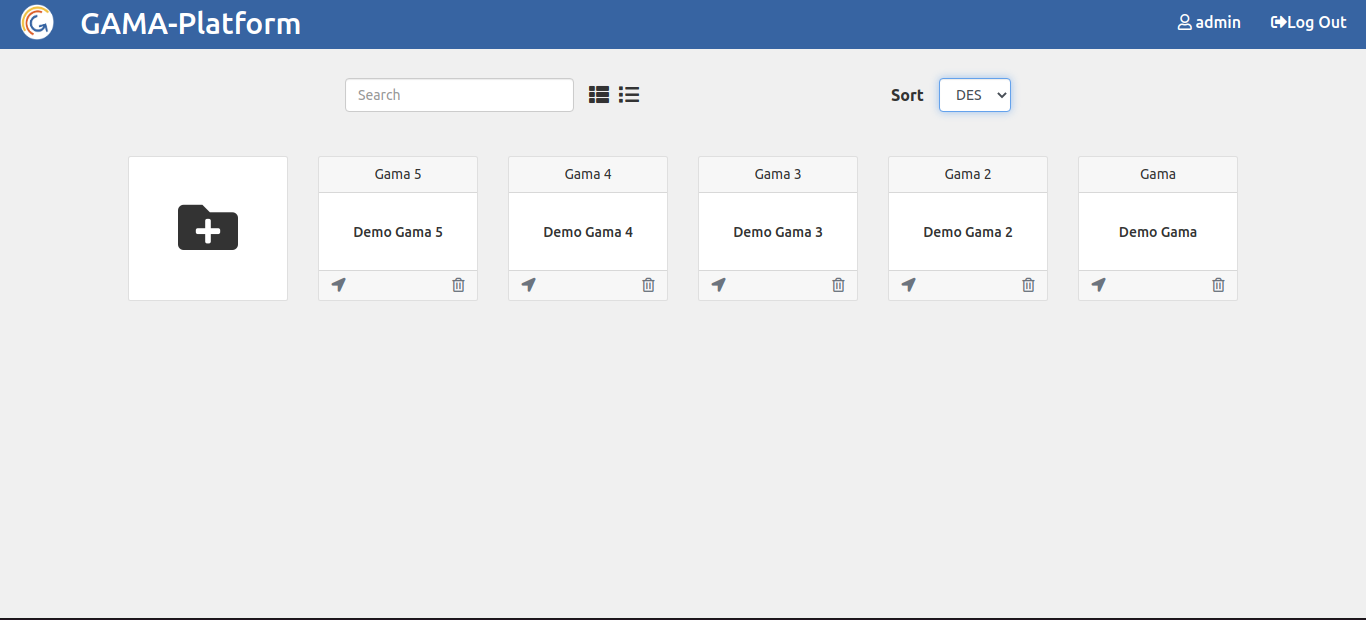
- Ngoài ra có chức năng search, sort theo tên project



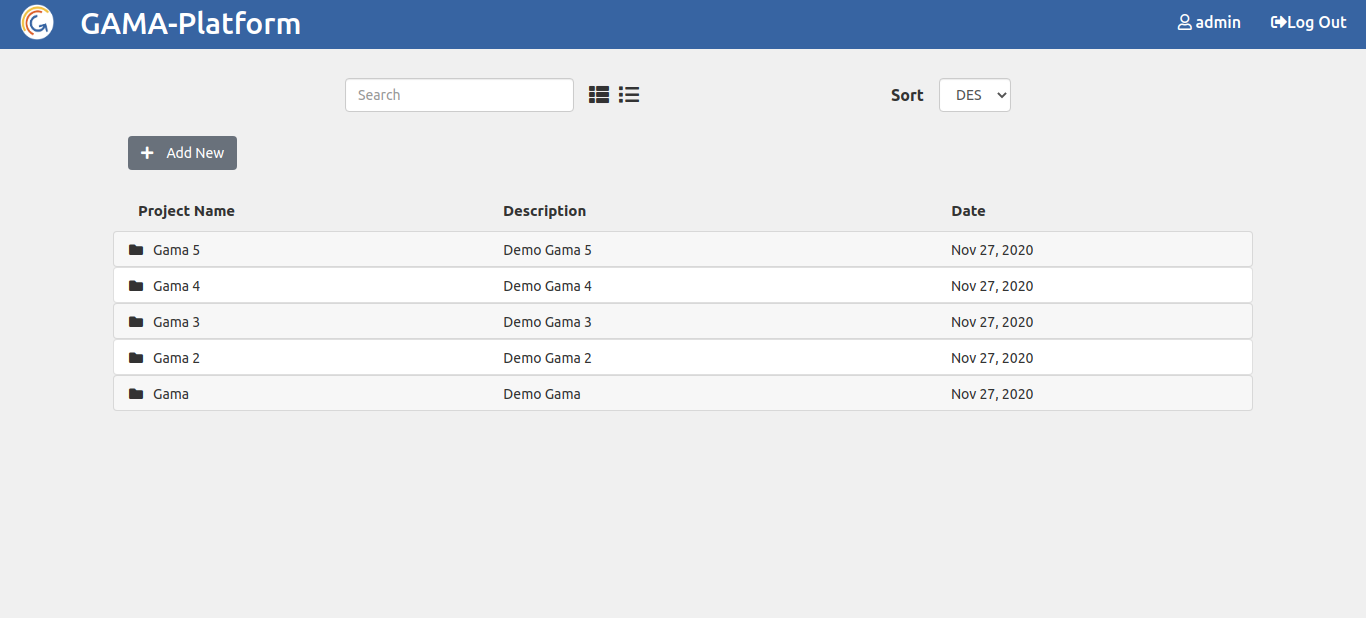
*Hình 4.15. Search theo tên project(dạng folder)*



*Hình 4.16. Search theo tên project(dạng list)*

**

*Hình 4.17. Sort theo tên project(dạng folder)*

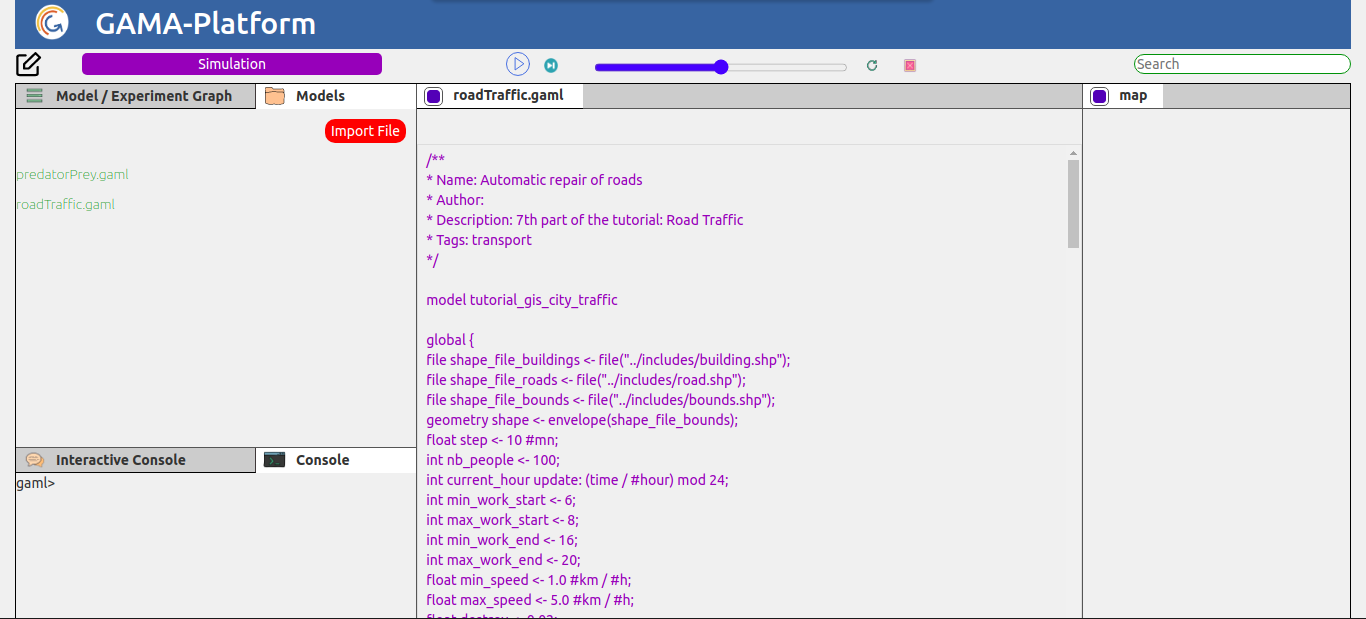
**

*Hình 4.18. Sort theo tên project(dạng list)*

*3.2.3. Giao diện mô phỏng*

- Ở tab Models

+ Chọn file model cần mô phỏng

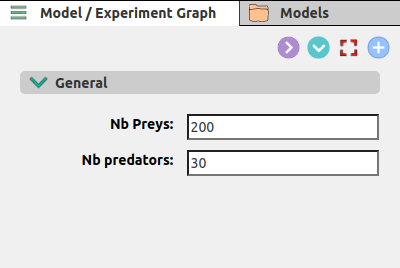
+ Có thể chỉnh sửa các thông số trực tiếp trên file .gaml

Hình 4. Giao diện thêm mới, edit file model

- Ở tab Model/ Experiment Graph

+ Hiển thị danh sách các parameter theo type của nó

+ Chỉnh sửa giá trị của các parameter.

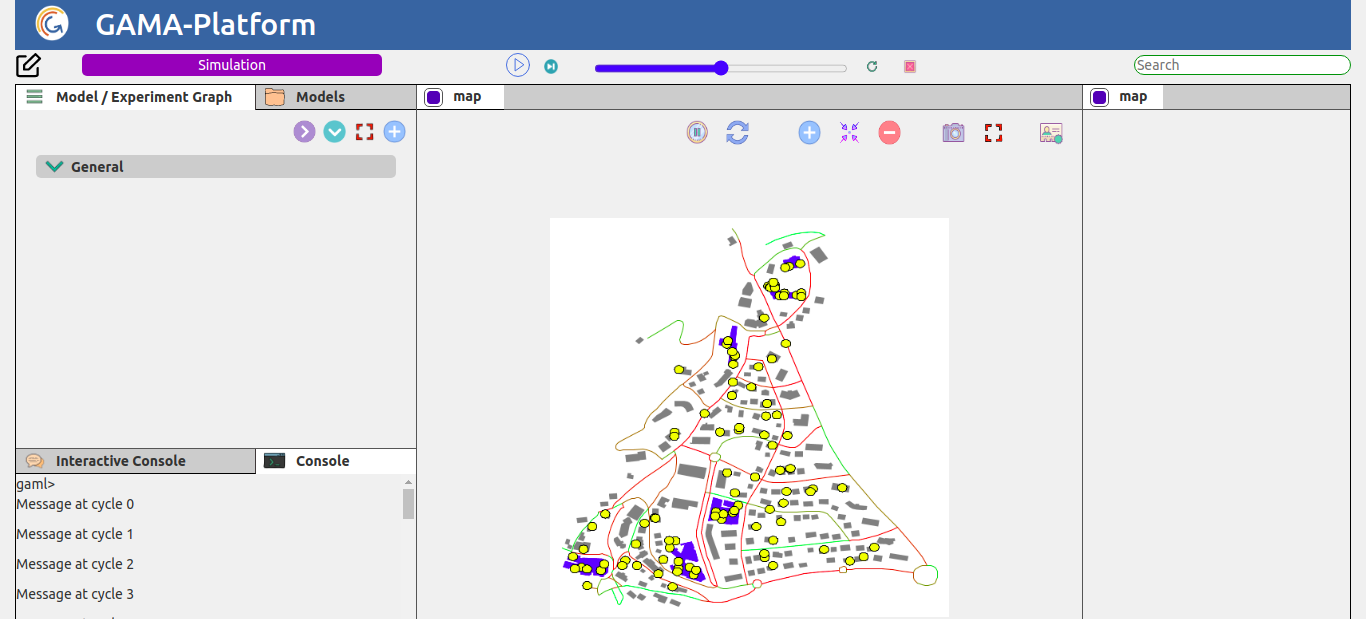


Hình 4.20 Chỉnh sửa giá trị của các parameter

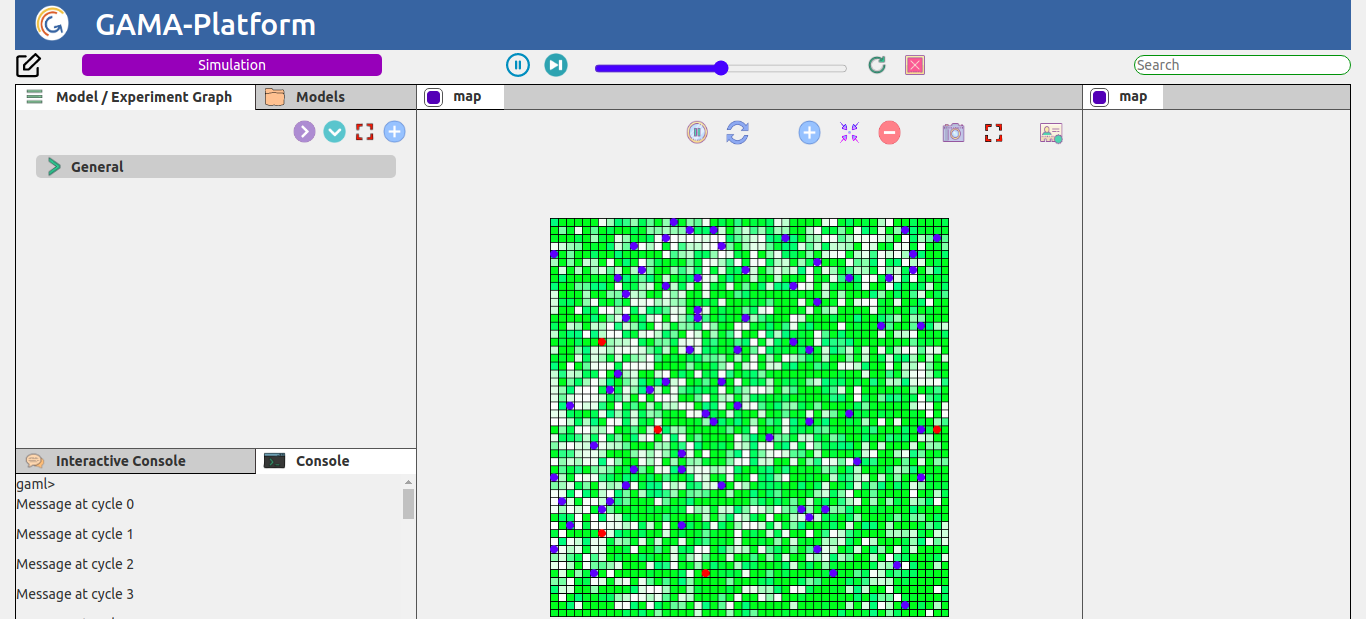
- Sau đó, tiến hành lưu thay đổi, tạo file xml, chạy lệnh trên terminal. Kết quả thu được thư mục kết quả tương ứng với từng file mô hình.

- Cuối cùng, cho hiển thị chuỗi ảnh mô phỏng và hiển thị message console lên giao diện.

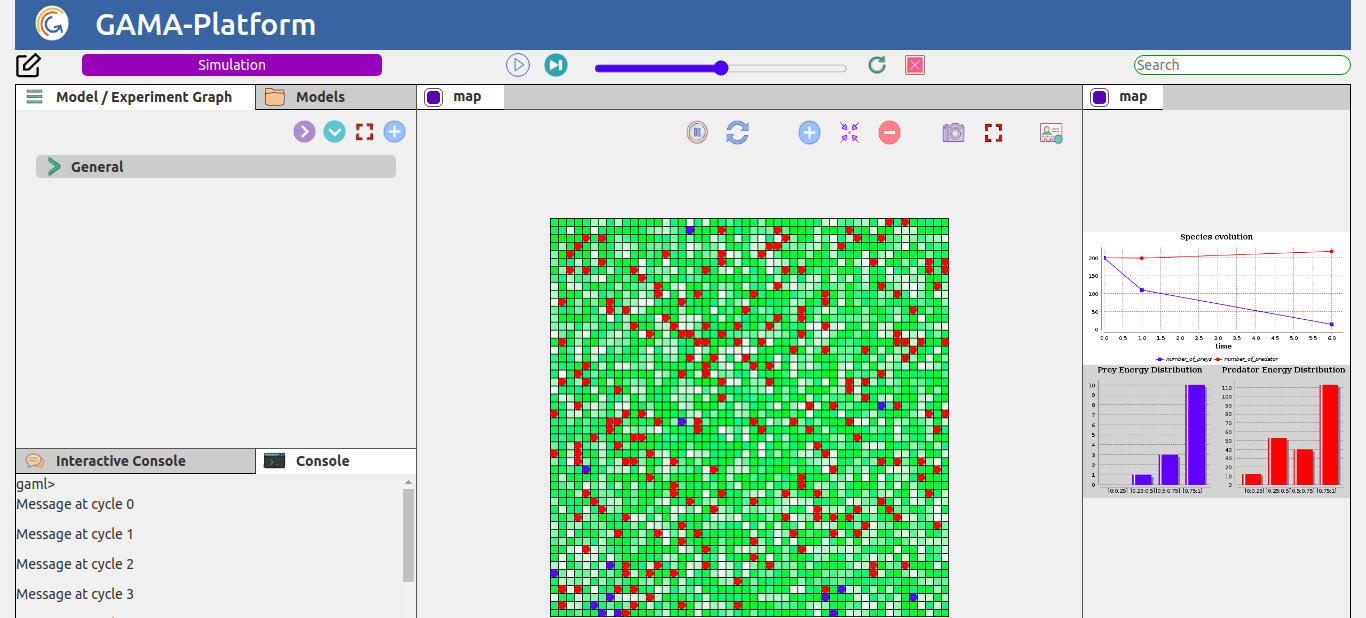
Dưới đây là kết quả chạy mô phỏng của một số file mô hình có sẵn:



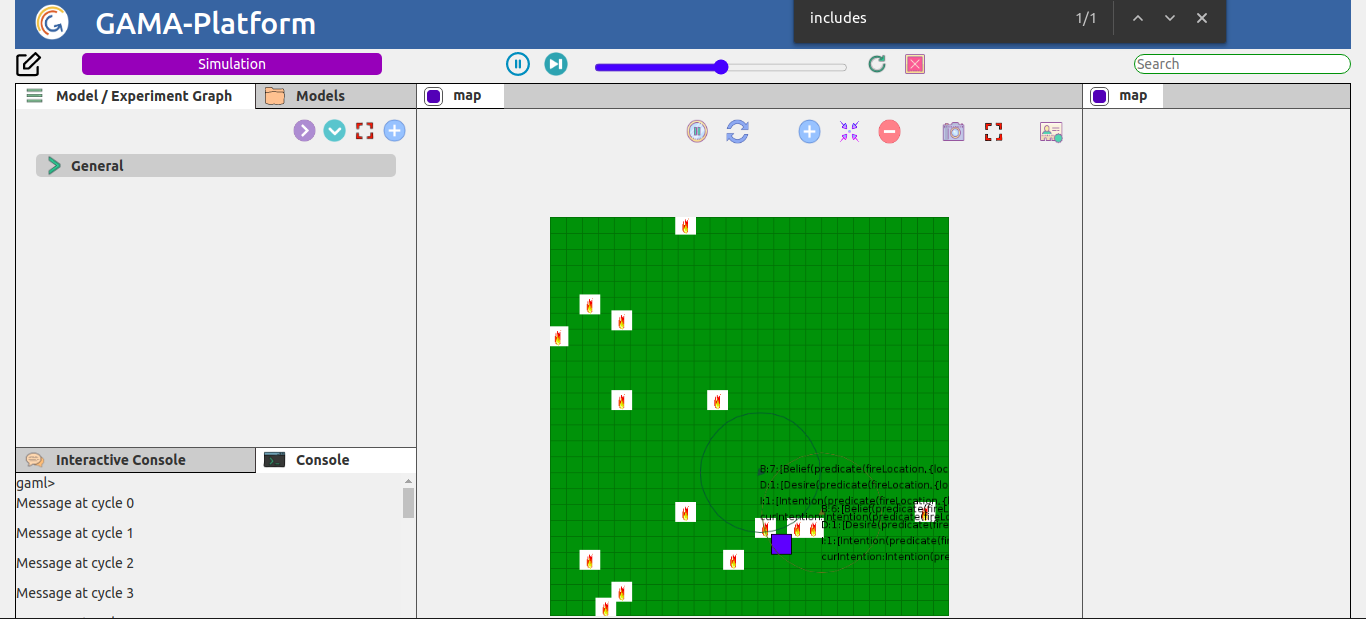
Hình 4.21 Giao diện chạy mô phỏng (1)



Hình 4.22 Giao diện mô phỏng dạng lưới (2)



Hình 4.23 Giao diện mô phỏng dạng biểu đồ (3)



Hình 4.4 Giao diện mô phỏng (4)

# KẾT LUẬN

**Ưu điểm**

* + Đáp ứng được yêu cầu bài toán đặt ra là đưa một ứng dụng trên nền desktop lên nền web để mô phỏng các mô hình hệ thống đa tác tử.
  + Chương trình có giao diện thân thiện, thuận tiện cho người dùng
  + Chức năng cập nhật (thêm, sửa, xóa), tìm kiếm được thực hiện nhanh chóng
  + Chạy được mô phỏng theo các file mô hình có sẵn.

**Nhược điểm**

Do thời gian và kinh nghiệm còn hạn chế nên kết quả vẫn còn một số thiếu sót cần tiếp tục được hoàn thiện để hệ thống có thể ứng dụng trong thực tiễn đạt hiệu quả tốt hơn như:

* Khả năng xử lý nghiệp vụ chưa linh hoạt nhạy bén
* Chưa giải quyết trọn vẹn các vấn đề nảy sinh trong quá trình quản lý
* Một số chức năng của hệ thống chưa được mở rộng (phạm vi yêu cầu còn hẹp)

**Hướng phát triển**

Hướng phát triển của website là tiến đến hoàn thiện đầy đủ hơn về các tính năng chưa làm được so với ứng dụng trên nền desktop cũng như hoàn thiện về mặt giao diện:

* Danh sách hiển thị file mô hình có thể thay vào đó là hiển thị dạng cây thư mục gồm thư mục cha và hai thư mục con: thư mục models chứa các file model (.gaml) và thư mục includes chứa các file ảnh phục vụ cho việc vẽ mô phỏng. Ở thư mục includes có thể import hoặc kéo thả file ảnh từ PC vào thay vì là các file ảnh có sẵn như hiện giờ.
* Phần hiển thị giao diện trong quá trình mô phỏng có thể hiển thị linh động về mặt số lượng output theo từng file model thay vì hiển thị cố định và chỉ có thể hiển thị tối đa hai loại hình mô phỏng như hiện giờ.

Thông qua quá trình nghiên cứu chuyên sâu về phát triển ứng dụng mô phỏng nền Web dựa trên tác tử sử dụng nền tảng GAMA, đồ án đã bước đầu hoàn thiện việc áp dụng nền tảng GAMA vào quá trình phát triển ứng dụng mô phỏng nền Web, khắc phục được một số nhược điểm phát sinh khi vận hành hệ thống trên desktop app. Bên cạnh đó, đồ án còn đưa ra một số giải pháp nhằm phát triển ứng dụng mô phỏng nền Web dựa trên tác tử, góp phần nâng cao ứng dụng thực tiễn của nền tảng GAMA vào cuộc sống. Mặc dù đã cố gắng hoàn chỉnh các yêu cầu nhưng đồ án vẫn còn rất nhiều thiếu sótmong nhận được sự chỉ bảo hướng dẫn của các thầy cô khác giúp đỡ xem xét, đề xuất thêm các ý kiến cũng như bổ sung các vấn đề phục vụ cho việc xây dựng chương trình để em có thể hoàn chỉnh hơn. Em xin cảm ơn các quý thầy cô.

# TÀI LIỆU THAM KHẢO

[1] Phan Tuấn Kiệt, *Xây dựng hệ thống mô phỏng dự báo sự phát triển đô thị,* Luận văn tốt nghiệp đại học, Trường đại học Cần Thơ, 2016 [Accessed 30-11-2020].

[2] Nguyễn Hồng Quang, Hồ Tường Vinh, Nguyễn Mạnh Hùng, *Xử lý song song mô phỏng đa tác tử GIS bằng phương pháp chia mảnh,* Phòng nghiên cứu MSI, Viện Quốc tế Pháp ngữ – ĐHQGHN, Khoa CNTT-TT, Học viện Bưu chính Viễn thông Hà Nội, 2016 [Accessed 30-11-2020].

*Nguồn Internet*

[1] Gama-flatform: <https://gama-platform.github.io/wiki/Home> [Accessed 30-11-2020]

[2] Spring Boot: <https://spring.io/projects/spring-boot> [Accessed 30-11-2020].

[3] Hibernate: <https://viettuts.vn/hibernate> [Accessed 30-11-2020].

[3] Angular: <https://angular.io/docs> [Accessed 30-11-2020].

[4] PostgreSQL: <https://www.postgresql.org/docs/> [Accessed 30-11-2020].

[5] Github: <https://topdev.vn/blog/github-la-gi/> [Accessed 30-11-2020].

[6] <https://spring.io/guides/gs/securing-web/> [Accessed 30-11-2020].

[7] <https://angular.io/> [Accessed 30-11-2020].

[8] <http://sqladvice.com/postgresql-la-gi-so-sanh-mysql-va-postgresql/> [Accessed 30-11-2020].

[9] <https://gama-platform.github.io/wiki/Installation> [Accessed 30-11-2020].

[10] <https://luonghoanghuong.wordpress.com/2015/10/25/gama-platform/> [Accessed 30-11-2020].

[11] <https://gama-platform.github.io/wiki/Headless> [Accessed 30-11-2020].

[12] <https://gama-platform.github.io/wiki/LearnGAMLStepByStep> [Accessed 30-11-2020].