ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

🙢🙡🕮🙣🙠

****

**BÁO CÁO BÀI TẬP LỚN**

**CÔNG NGHỆ PHẦN MỀM NÂNG CAO**

***Đề tài:***

***Xây dựng ứng dụng truyền hình Online***

**Giảng viên: TS. Nguyễn Thị Huyền Châu**

**Học viên thực hiện:**

|  |  |
| --- | --- |
| 1. Nguyễn Thị Khánh Chi |  |
| 2. Nguyễn Văn Dương |  |
| 3. Nguyễn Doãn Tùng  4. Đỗ Huy Yên |  |
|  |  |

HÀ NỘI - 2016

**LỜI NÓI ĐẦU**

Hiện nay việc xem kênh truyền hình trên internet đang được phổ biến rộng rãi ở nhiều nước trên thế giới, trong đó có cả Việt Nam, chỉ cần có mạng internet là có thể xem được các kênh truyền hình dù bất kể ở đâu. Không chỉ người nước ngoài muốn xem các kênh truyền hình ở Việt Nam và người Việt Nam muốn xem các kênh truyển nước ngoài. Hơn thế, không nhất thiết chúng ta phải có tivi và đăng kí một dịch vụ truyền hình số vệ tinh mới có thể xem được các kênh truyền hình, với cơ sở hạ tầng sẵn có như: Hệ thống mạng Internet phát triển, hệ thống máy tính hiện đại việc truyển khai dịch vụ truyền hình trên Internet hoàn toàn khả thi và đáp ứng được nhu cầu của nhiều người. Xuất phát từ nhu cầu đó, nhiều công ty đã triển khai dịch vụ truyền hình trên Internet như: FPT Play của FPT, MyTV của VNPT, … Chính vì thế nhóm chúng em quyết định tìm hiểu kiến trúc của dịch vụ này, đồng thời dựng nên mô hình để mô phỏng dịch vụ.

*Báo cáo gồm 4 chương:*

*Chương 1: Giới thiệu tổng quan*

*Chương 2: Kiến trúc hệ thống*

*Chương 3: Đặc tả hệ thống*

*Chương 4: Kết luận và hướng phát triển*

Qua đây, Nhóm em xin chân thành cảm ơn TS. Nguyễn Thị Huyền Châu đã giúp đỡ tận tình cho nhóm em trong thời gian học tập vừa qua.

Xin chân thành cảm ơn.

**CHƯƠNG I: GIỚI THIỆU TỔNG QUAN**

# Mô hình phát triển

## Mô hình nhóm chọn là mô hình agile

Để nâng cao chất lượng và tiến độ thay vì tập trung ứng dụng vào phát triển phần mềm theo hướng kiểm thử, nhóm đã ứng dụng BDD(Behavior Driven Development) tập trung vào phát triển phần mềm theo hướng hành vi như nhóm đã trình bày và báo cáo ở bài trước. Dựa vào requirement các kịch bản test (Scenarios) sẽ được viết trước dưới dạng ngôn ngữ tự nhiên và dễ hiểu nhất sau đó mới thực hiện cài đặt source code đễ pass qua tất cả các stories đó

Những kịch bản test này được viết dưới dạng các feature file và đòi hỏi sự cộng tác từ tất cả các thành viên tham gia dự án.

# Ứng dụng

# Lời giới thiệu

Hiện nay có hàng nghìn người việt định cư ở nước ngoài mong muốn, quan tâm đến tình hình kinh tế và chính trị ở Việt Nam. Họ rất mong muốn được xem tin tức và tình hình ở quê hương, vấn đề đặt ra là phát các kênh truyền hình của việt nam ở nước ngoài chưa có hoặc chi phí rất đắt. Nhóm chúng em đề xuất mô hình tổng quan và sử dụng công nghệ streaming để truyền tải dữ liệu và công nghệ CDN( Content Delivery Network ), công nghệ nén, các kiến trúc hướng dịch vụ có thể giúp người cho hàng nghìn người định cư ở nước ngoài trên khắp thế giới có thể xem được các kênh truyền hình ở Việt Nam 1 cách nhanh nhất, độ phân giải đẹp nhất và băng thông ít nhất.

# Bảng thuật ngữ

|  |  |
| --- | --- |
| Thuật ngữ | Ý nghĩa |
| CDN | Content Delivery Network |
| API | Application Programming Interface |
| MVC | Model – View – Controler |
| M3U8 | Streaming Protocol |

# Yêu cầu hệ thống và người dùng

**Yêu cầu chức năng**

- Đăng ký gia hạn dịch vụ trên hệ thống

- Thanh toán qua paypal hoặc chuyển khoản 1 cách nhanh nhất.

- Xem được tin tức về dịch vụ, gói đăng ký,

- Vì lệch múi giờ ở các nước, người dùng yêu cầu có thể xem lại được bất cứ khi nào, nơi đâu chỉ cần có mạng 3G hoặc internet.

**Yêu cầu phi chức năng**

- Xem được trên tất cả các thiết bị

- Xem các kênh fullHD

- Xem được với băng thông nhỏ nhất kể cả 3G

- Xem được trên các thiết bị có cấu hình yếu

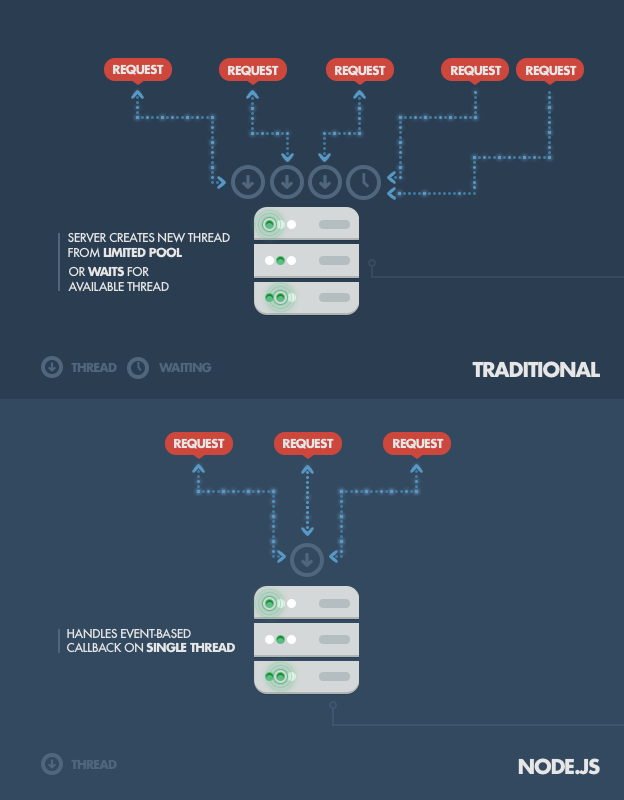
# Chương II: Kiến trúc hệ thống

# Kiến trúc NodeJS.

Node JS được phát triển bởi Ryan Dahl vào năm 2009. NodeJS bắt đầu làm nên cuộc tiến hóa cho các ứng dụng Web Với sáng tạo của mình, Ryan Dahl đã tìm cách để cung cấp cho các nhà phát triển Web cơ hội để tạo ra các trang web có khả năng tương tác cao với khả năng đẩy (push) để tối ưu hóa băng thông và hiệu quả. Ngày nay, hàng chục công ty trong đó có LinkedIn, The New York Times, PayPal, and eBay đã tận dụng mô hình sự kiện I/O của Node để cung cấp sức mạnh cho các chương trình mạng diện rộng của họ trên Web.

Một vài năm gần đây, các trang Web là một môi trường không trạng thái (stateless - có nghĩa là nó không lưu lại bất kì trạng thái gì của lần gọi trước và nó cũng không biết gì về phía client đang thực hiện request) dựa trên mô hình yêu cầu – hồi đáp không trạng thái (stateless request – response). Hầu hết các tính năng tương tác được gói gọn trong Flash hoặc JavaApplet là đơn vị cô lập trong một môi trường Web. Node cho phép các ứng dụng Web thiết lập thời gian thực, kết nối hai chiều client và server, cho phép trao đổi dữ liệu một cách tự do. Lợi thế chính của nó nằm trong việc sử dụng tính bất đồng bộ, hướng sự kiện I/O, do đó việc quản lí các ứng dụng sử dụng nhiều dữ liệu phân tán trên nhiều thiết bị sẽ trở nên nhẹ nhàng và hiệu quả. Node hiện đang được tài trợ bởi Joyent, một công ty phần mềm chuyên về điện toán đám mây hiệu suất cao. Kể từ khi phiên bản đầu tiên được phát hành trên nền tảng hệ điều hành Linux. Node đã mua tương thích với Mac OS X, Windows, Solaris, FreeBSD và OpenBSD hệ điều hành [2]. Node ngày càng được các nhà phát triển thừa nhận, tính đến nay đã có 230 phiên bản được ban hành trong vòng 4 năm.

Hoạt động Nodejs



Hình 6: Cơ chế tiếp nhận các request của nodejs

Node js khả năng xử lý một số lượng lớn các kết nối đồng thời bằng thông lượng cao. Nếu như các ứng dụng web truyền thống, các request tạo ra một luồng xử lý yêu cầu mới và chiếm RAM của hệ thống thì việc tài nguyên của hệ thống sẽ được sử dụng không hiệu quả.

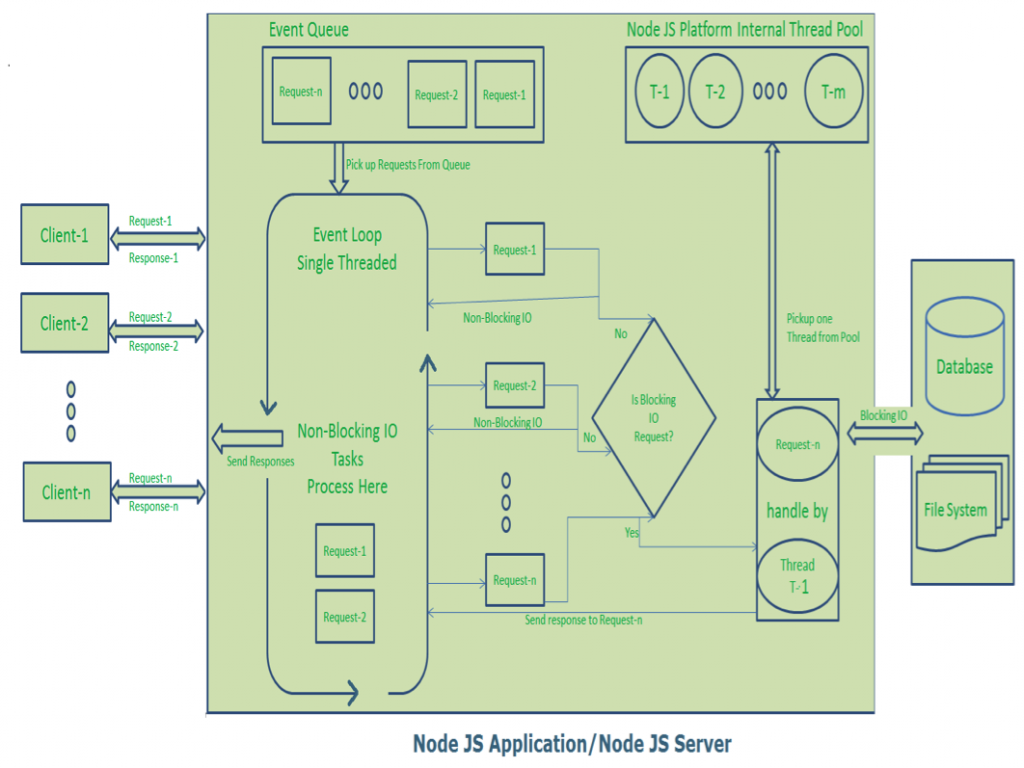
Cơ chế của Node js như hình 6 sẽ được giải thich như sau:

* Client gửi request đến Web Server
* Node js Web Service duy trì một luồng giới hạn để cung cấp dịch vụ cho Client Request.
* Node js Web Service nhận tất cả các request và đặt chúng vào một trong Queue. Nó được gọi là một Event Queue.
* Node js Web Service có một thành phần được gọi là "Event Loop".
* Event Loop chỉ sử dụng một luồng đơn để xử lý.
* Event Loop kiểm tra tất cả các Request đặt trong Event Queue. Nếu không có request nào thì chờ request đến vô thời hạn.
* Nếu có request thì sẽ lấy một request từ Event Queue:

+ Khởi động quá trình xử lý tiến trình từ client request

+ Nếu Client Request không chứa nhiều Blocking I/O thì xử lý tất cả mọi thứ và chuẩn bị cho quá trình gửi lại phản hồi cho phía client.

+Nếu Client Request chứa nhiều Blocking I/O như việc tương tác với cơ sở dữ liệu, tập tin hệ thống, dịch vụ mở rộng, thì nó sẽ thực hiện theo các phương pháp khác.



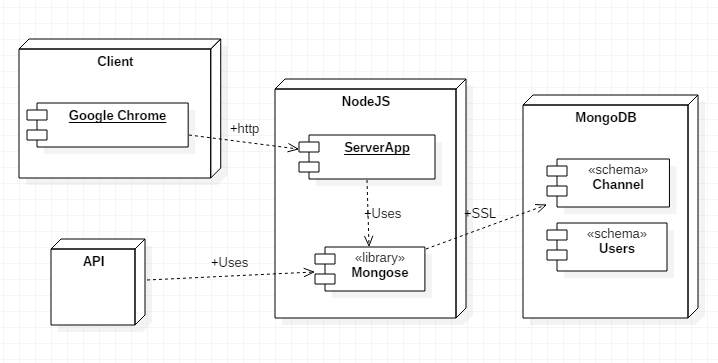
Hình 7: cơ chế xử lý các request của nodejs

Trong hình 7 giả xử có n số lượng client gửi request lên web service. Client là Client-1, Client-2, ...client-n. Web server duy trì một vùng các luồng có giới hạn và m là số luồng của vùng luồng đó. Node js Web service nhận các request từ client-1, client-2, ...client-n và đặt vào trong Event Queue. Node js Event Loop chọn các request theo dạng một đối một. Event Loop chọn client-1 request-1. Kiểm tra trong client-1 request-1 có không yêu cầu bất kì hoạt động Blocking I/O hoặc mất nhiều thời gian cho việc tính toán phức tạp. Request này thì đơn giản và Non-blocking I/O, nó không có đòi hỏi một xử lý nào riêng biệt.Event Loop xử lý tất cả các giai đoạn để cung cấp cho hoạt động của client-1 request-1 (hoạt động ở đây nghĩa là các function của javascript) và chuẩn bị response-1.Event Loop gửi Response-1 đến Client-1. Event Loop chọn client-2 request-2. Kiểm tra trong client-2 request-2 có không yêu cầu bất kì hoạt động Blocking I/O hoặc mất nhiều thời gian cho việc tính toán phức tạp.Request này thì đơn giản và Non-blocking I/O.

Event Loop xử lý tất cả các giai đoạn để cung cấp cho hoạt động của client-2 request-2 và chuẩn bị response-2.Event Loop gửi Response-2 đến Client-2, Event Loop chọn client-n request-n. Kiểm tra trong client-n request-n có không yêu cầu bất kì hoạt động Blocking I/O hoặc mất nhiều thời gian cho việc tính toán phức tạp.Request này thì phức tạp và nhiệm vụ Blocking I/O. Event Loop thì không thể xử lý request này ngay được.

Event Loop lựa chọn Thread T-1 từ nội bộ khu vực Thread và chỉ định Client-n Request-n này ánh xạ với Thread T-1. Thread T-1 đọc và xử lý Request-n, thực hiện Blocking IO cần thiết hoặc tính toán nhiệm vụ, và hoàn tất để chuẩn bị Response-n.Thread T-1 gửi Response-n này đến Event Loop. Event Loop trong công đoạn này gửi Response-n này tới Client-n.

## Kiến trúc hệ thống



Nhìn một cách tổng quan, hệ thống được xây dựng dưới dạng ứng dụng Web, với một địa chỉ URL xác định, người dùng sẽ sử dụng trình duyệt Web như: Chrome, IE, Firefox, … để thao tác với ứng dụng. Ứng dụng Web sẽ sử dụng thư viện của MongoDB là “mongose” để thao tác với cơ sở dữ liệu (CSDL). “mongose” là một thư viện, cung cấp một môi trường để mô hình hóa dữ liệu, thực thi các cấu trúc cần thiết mà vẫn duy trì sự linh hoạt cho MongoDB. Các dữ liệu lưu trong CSDL đều dưới dạng “schema”. Một dạng cấu trúc để lưu trữ dữ liệu thay vì thao tác trực tiếp lênh CSDL. Thêm vào đó, các kênh truyền hình sẽ được cung cấp cho bên thứ ba sử dụng, việc họ thao tác với CSDL để thêm, sửa, xóa các kênh là cần thiết vì thế chúng ta cần phải cung cấp cho họ API để họ thao tác được với hệ thống khi họ thao tác với CSDL hệ thống. Vì thế thông qua thư viện “mongose” bên thứ ba sẽ thao tác được với CSDL thông qua các các dữ liệu “schema”.

Ví dụ về đoạn mã “schema” của CardNumber:

var CardnumberSchema = **new** Schema({

\_id: { type: Number, index: **true**, **default**: 0},

number: { type: String, unique: **true**, required: **true**},

serial: { type: String, unique: **true**, required: **true**},

server\_id: {type: Number, **default**: 1},

capacity: {type: Number, **default**: 0},

user\_id: Number,

use\_date: Date,

free\_day: Number,

note: String,

token: String,

user\_create: {type: Number, required: **true**},

status: {type: Boolean, **default**: **true**},

active: {type: Boolean, **default**: **true**},

is\_lock: {type: Boolean, **default**: **false**},

deleted: {type: Boolean, **default**: **false**},

user\_delete: Number,

date\_delete: Date,

created\_at: { type: Date, **default**: Date.now },

updated\_at: { type: Date, **default**: Date.now }

});

* 1. **Mô hình MVC**

*- Controler***:** Các event của hệ thống đều được định nghĩa ở trong file SocketClient.js, file này sẽ tham chiếu đến thư mục ***controller,*** nơi định nghĩa và xử lí từng event của hệ thống.

* *Model***:** Chứa các hàm thao tác trực tiếp với cơ sở dữ liệu MongoDB, đều được định nghĩa trong thư mục ***model***.
* *View*: Chứa nội dung hiển thị ra đối với người dùng. Các giao diện đều được đặt trong thư mục ***view***



Hình 1: Mô hình MVC của hệ thống

**Luồng đi mô hình**

Khi có một yêu cầu từ phía client gửi đến server, Bộ phận controller có nhiệm vụ nhận yêu cầu, xử lý yêu cầu đó. Và nếu cần, nó sẽ gọi đến phần model, vốn là bộ phần làm việc với Database..

Sau khi xử lý xong, toàn bộ kết quả được đẩy về phần View. Tại View, sẽ gen ra mã Html tạo nên giao diện, và trả toàn bộ html về trình duyệt để hiển thị.

Ưu điểm và nhược điểm của MVC

**Ưu điểm**

Thể hiện tính chuyên nghiệp trong lập trình, phân tích thiết kế. Do được chia thành các thành phần độc lập nên giúp phát triển ứng dụng nhanh, đơn giản, dễ nâng cấp, bảo trì..

**Nhược điểm**

Đối với dự án nhỏ việc áp dụng mô hình MC gây cồng kềnh, tốn thời gian trong quá trình phát triển. Tốn thời gian trung chuyển dữ liệu của các thành phần.

Ví dụ:

File SocketClient.js định nghĩa các sự kiện của hệ thống. Khi người dùng login, các thông tin mà người dùng nhập vào để login sẽ được controler lấy ra dưới dạng:

{"action":"login","type\_login":"card\_login","server\_id":1,"card\_number":"729X1ESYHK"}

Với action được nhập là “login”, file SocketClient.js sẽ nhận biết và gọi đến action xử lí là file LoginController.js, với hàm để xử lí action login là: function(client,json\_request,response,callback,logger)

Các thông tin này sẽ được phía controler xử lí và thao tác với phía model:

* 1. **Web socket**

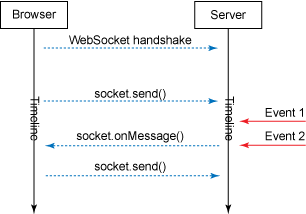
WebSoket là công nghệ hỗ trợ giao tiếp hai chiều giữa client và server bằng cách sử dụng một TCP socket để tạo một kết nối hiệu quả và ít tốn kém. Mặc dù được thiết kế để chuyên sử dụng cho các ứng dụng web, lập trình viên vẫn có thể đưa chúng vào bất kì loại ứng dụng nào. Dữ liệu truyền tải thông qua giao thức HTTP (thường dùng với kĩ thuật Ajax) chứa nhiều dữ liệu không cần thiết trong phần header. Một header request/response của HTTP có kích thước khoảng 871 byte, trong khi với WebSocket, kích thước này chỉ là 2 byte (sau khi đã kết nối).

Vậy giả sử bạn làm một ứng dụng game có thể tới 10,000 người chơi đăng nhập cùng lúc, và mỗi giây họ sẽ gửi/nhận dữ liệu từ server. Hãy so sánh lượng dữ liệu header mà giao thức HTTP và WebSocket trong mỗi giây:

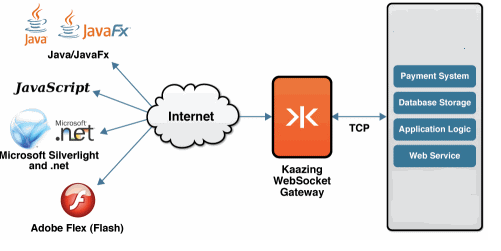
* HTTP: 871 x 10,000 = 8,710,000 bytes = 69,680,000 bits per second (66 Mbps)
* WebSocket: 2 x 10,000 = 20,000 bytes = 160,000 bits per second (0.153 Kbps)

Như bạn thấy chỉ riêng phần header thôi cũng đã chiếm một phần lưu lượng đáng kể với giao thức HTTP truyền thống

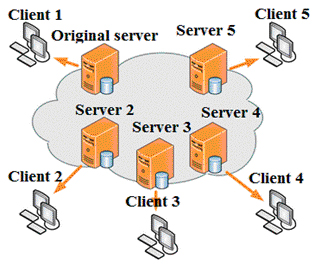
**Giao thức bắt tay websocket**

.

websocket sử dụng 3 events sau:

* **onopen:** Khi websocket đã được mở
* **onmessage:** Khi nhận được một message
* **onclose:** Khi websocket được đóng lại
* **Mô hình websocket xem truyền hình**
* ****
  1. **Công nghệ CDN(Content Delivery Network )**

Mạng giao dịch nội dung (Content Delivery Network ): Hệ thống các máy tính được kết nối với nhau qua Internet để truyền nội dung tới người sử dụng. CDN (Content Devilery Network) được tạm hiểu là một hệ thống máy chủ được đặt ở nhiều nơi khác nhau trên thế giới và chứa những bản sao dữ liệu của nội dung website trong hệ thống và khi người dùng truy cập vào thì các bản sao đó nằm tại một máy chủ gần với người dùng nhất sẽ được thay thế với dữ liệu nội dung gốc của website. Giả sử như máy chủ website bạn ở Châu Âu nhưng khi một người dùng ở Việt Nam truy cập vào thì những dữ liệu mà người dùng nhận được là bản sao của máy chủ gốc được lưu trữ tại những máy chủ trong hệ thống CDN ở khu vực Đông Nam Á hoặc hoặc tại Việt Nam nơi gần người dùng nhất. Nội dung các loại được phân phối tốt nhất và hiệu quả nhất khi ứng dụng giải pháp CDN bao gồm các đối tượng web, các tập tin media, phần mềm, tài liệu, video streaming thời gian thực,.. Mất 80%-90% thời gian để trình duyệt hiển thị ra một trang web mà bạn đang xem là để tải các thành phần trong trang như: hình ảnh (images,) các dữ liệu để định dạng tài liệu HTML (stylesheet – CSS), các đoạn script, flash,… Phương pháp này không chỉ giúp cho thời gian phản hồi nhanh hơn mà còn dễ dàng hơn nhờ sự hỗ trợ của các hệ thông máy chủ CDN (Content Delivery Network) – Hệ thống mạng lưới phân phối nội dung.



**Chương III: ĐẶC TẢ HỆ THỐNG**

Hệ thống có ba tác nhân (actor) chính tương tác gồm: Người dùng, người quản trị và bên thứ ba. Người dùng muốn dùng được dịch vụ thì phải đăng ký mua 1 mã thẻ trên hệ thống, người dùng sẽ chọn gói cước cần mua tương ứng với thời gian sử dụng, khi hết hạn sử dụng người dùng chỉ việc gia hạn thêm thời gian sử dụng. Mã thẻ này đơn giản là một dãy số để xác thực người dùng. Khi có mã thẻ này thì mới có thể xem được dịch vụ. Tuy nhiên, chúng ta sẽ dễ dàng gặp phải trường hợp là có quá nhiều client kết nối đến server sẽ làm server quá tải và kết quả là hệ thống không thể hoạt động. Vì vậy việc dùng http để thiết kế là điều không hợp lí do lượng thông tin để kết nối liên tục server sẽ bị chậm. Do đó hệ thống sử dụng websocket để người dùng sẽ kết nối và xem dịch vụ. Trang chủ của hệ thống sẽ là một trang web quảng cáo để giới thiệu dịch vụ, người dùng sẽ đăng kí mua mã thẻ thông qua trang chủ. Để người dùng đăng ký mã thẻ hệ thống sử dụng một ứng dụng cho phép vừa làm làm trang giới thiệu, quảng cáo, vừa để người dùng đăng ký mua dịch vụ và xem qua máy tính. Việc giao tiếp giữa server và client sử dụng socket nên bất kể thiết bị nào cũng tương thích.

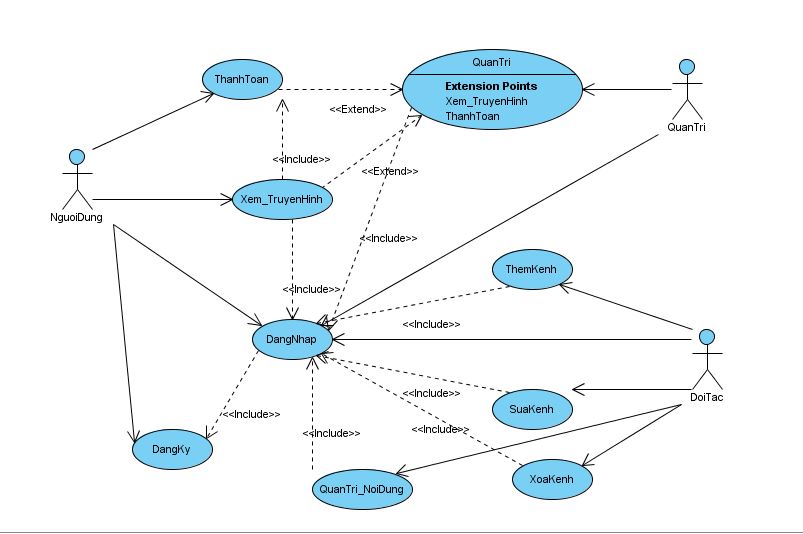
Ngoài ra, hệ thống sẽ có một bên thứ ba tham gia vào việc quản trị, bên thứ ba này sẽ là các nhà cung cấp các kênh truyền hình. Nhiệm vụ chính của bên thứ ba này chính là việc duy trì ổn định của các kênh truyền hình, ngoài ra họ có thể cung cấp thêm các kênh mới, hoặc xóa một số kênh. Tuy nhiên việc thao tác của bên thứ ba vào hệ thống sẽ phải sử dụng API để đảm bảo tính đóng hệ thống. Như vậy hệ thống xem kênh truyền trên Internet sẽ bao gồm 3 mô hình:

* Ứng dụng Web
* Websocket
* API để cho bên thứ ba sử dụng.

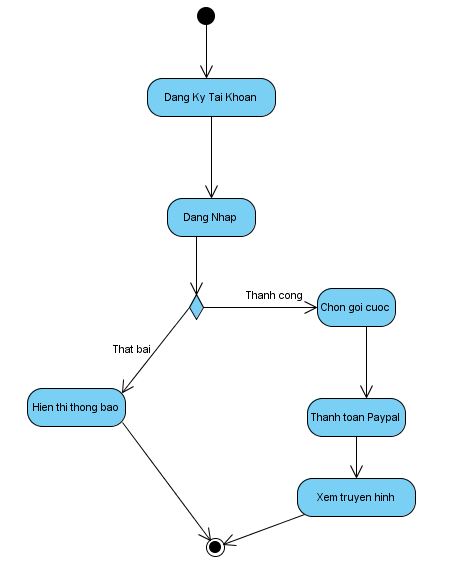
Mỗi một mô hình sẽ được đặc trên một server khác nhau nhằm giảm tải, như vậy sẽ sinh ra 4 server, một server để lưu trữ dữ liệu sử dụng MongoDB làm hệ quản trị, 3 server còn lại nhằm phục vụ 3 mô hình như đã nêu. Tất cả các mô hình đều viết theo mô hinh MVC (Model – View – Controler) và ứng dụng công nghệ Nodejs như đã trình bày ở trên.

* Để xem được dịch vụ yêu cầu người dùng phải dùng 1 tài khoản để đăng ký, người dùng đăng ký xong chọn gói cước tương ứng thời gian xem, ví dụ gói cước 6 tháng, gói cước 1 năm… Sau khi đăng ký chọn dịch vụ xong người dùng sẽ thanh toán tự động qua paypal hoặc chuyển khoản. Hệ thống sẽ tự động xác nhận và thêm vào cơ sở dữ liệu.
* Để quản trị nội dung sẽ thuê đối tác để quản trị nội dung, chất lượng của các kênh truyền hình.

**3.1. Mô hình usercase hệ thống**

****

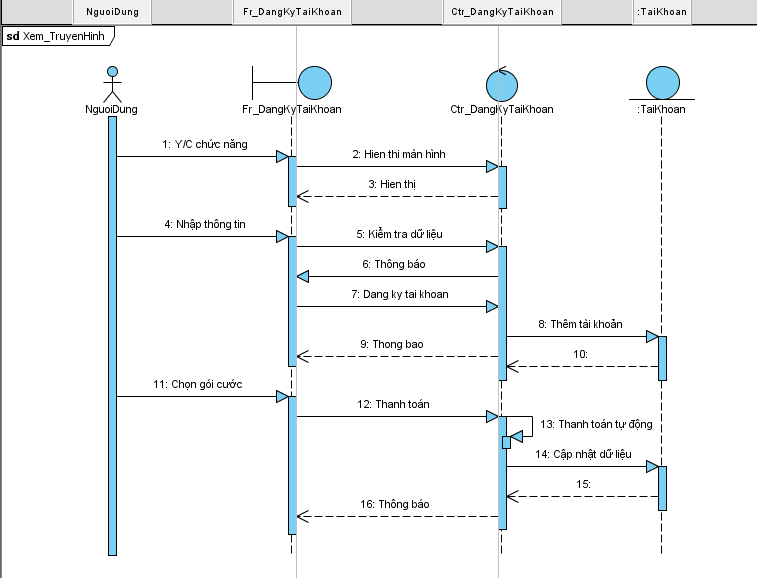
**3.2. Mô hình hoạt động**

****

**Giải thích biểu đồ:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Khái quát về xử lý** | **Hệ bất thường** | **Remarks** |
| **1.** | Người dùng thực hiện đăng ký tài khoản |  |  |
| **2.** | Sau khi đăng ký xong. Người dùng sử dụng tài khoản đó để đăng nhập vào hệ thống | Đăng nhập thất bại hệ thống sẽ hiển thị thông báo. |  |
| **3.** | Người dùng chọn gói cước sử dụng sau đó thực hiện thanh toán qua Paypal. |  |  |
| **4.** | Sau khi thực hiện xong các bước trên thì co thể bắt đầu xem truyền hình |  |  |

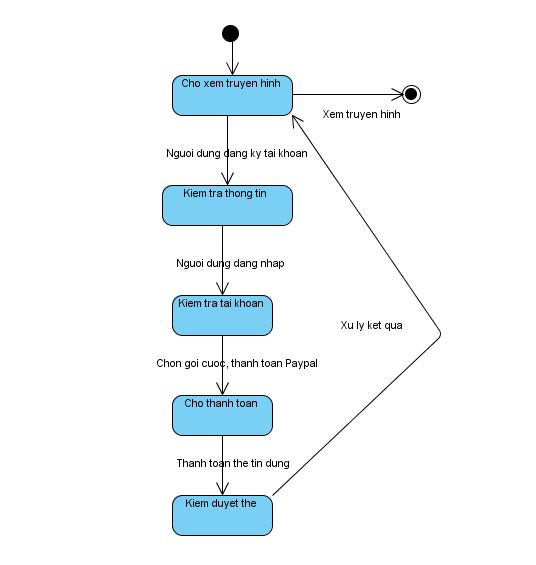
**3.3. Mô hình tuần tự**

****

**Giải thích biểu đồ:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Khái quát về xử lý** | **Hệ bất thường** | **Remarks** |
| **1.** | Người dùng chọn chức năng đăng ký tài khoản. Hệ thống thực hiện hiển thị màn hình đăng ký. |  |  |
| **2.** | Người dùng nhập thông tin đăng ký. Hệ thống thực hiện kiểm tra dữ liệu rồi thêm tài khoản vào cơ sở dữ liệu. | Dữ liệu không chính xác hệ thống hiển thị message thông báo |  |
| **3.** | Người dùng chọn gói cước sử dụng sau đó thực hiện thanh toán. Hệ thống tự động gọi dịch vụ thanh toán trực tuyến và cập nhật thông tin gói cước vào cơ sở dữ liệu. |  |  |

**3.3. Mô hình trạng thái**

****

**Giải thích biểu đồ:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Khái quát về xử lý** | **Hệ bất thường** | **Remarks** |
| **1.** | Người dùng ở trạng thái chờ xem truyền hình, họ cần đăng ký một tài khoản. |  |  |
| **2.** | Người dùng chờ hệ thống kiểm tra thông tin, sau đó thực hiện đăng nhập vào hệ thống |  |  |
| **3.** | Người dùng chờ hệ thống kiểm tra tài khoản, tiếp theo người dùng chọn gói cước và thanh toán paypal. |  |  |
| **4.** | Người dùng chờ thanh toán và sử dụng thẻ tín dụng để thanh toán. |  |  |
| **5.** | Hệ thống kiểm duyệt thẻ và xử lý thông tin. Sau đó sẽ chuyển sang trạng thái xem truyền hình |  |  |

# CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Sau một khoảng thời gian nghiên cứu và tìm hiểu về nhu cầu của người sử dụng các dịch vụ truyền hình thì nhóm em đã quyết định xây dựng ứng dụng mô phỏng truyền hình trên internet. ứng dụng phát triển dựa trên mô hình MVC và một số mô hình : Ứng dụng Web, Websocket, API.