**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA CÔNG NGHỆ PHẦN MỀM**

**TÔ NGỌC KHÁNH**

**PHÙNG ĐỨC LÂM**

**KHÓA LUẬN TỐT NGHIỆP**

**XÂY DỰNG GAME MOBILE ONLINE**

**KỸ SƯ NGÀNH CÔNG NGHỆ PHẦN MỀM**

**TP. HỒ CHÍ MINH, 2014**

**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA CÔNG NGHỆ PHẦN MỀM**

**TÔ NGỌC KHÁNH – 10520225**

**PHÙNG ĐỨC LÂM – 10520253**

**KHÓA LUẬN TỐT NGHIỆP**

**XÂY DỰNG GAME MOBILE ONLINE**

**KỸ SƯ NGÀNH CÔNG NGHỆ PHẦN MỀM**

**GIẢNG VIÊN HƯỚNG DẪN**

**PGS. TS VŨ THANH NGUYÊN**

**KS LÂM VINH ĐẠO**

**TP. HỒ CHÍ MINH, 2014**

DANH SÁCH HỘI ĐỒNG BẢO VỆ KHÓA LUẬN

Hội đồng chấm khóa luận tốt nghiệp, thành lập theo Quyết định số …………………… ngày ………………….. của Hiệu trưởng Trường Đại học Công nghệ Thông tin.

* 1. …………………………………………. – Chủ tịch.
  2. …………………………………………. – Thư ký.
  3. …………………………………………. – Ủy viên.
  4. …………………………………………. – Ủy viên.

|  |  |
| --- | --- |
| ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH  **TRƯỜNG ĐẠI HỌC**  **CÔNG NGHỆ THÔNG TIN** | **CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**  **Độc Lập - Tự Do - Hạnh Phúc** |
|  | *TP. HCM, ngày…..tháng…..năm……..* |
|  |  |

**NHẬN XÉT KHÓA LUẬN TỐT NGHIỆP**

**(CỦA CÁN BỘ HƯỚNG DẪN)**

|  |  |  |
| --- | --- | --- |
| **Tên khóa luận:** | | |
| **XÂY DỰNG GAME MOBILE ONLINE – TETRIS ONLINE** | | |
| **Nhóm SV thực hiện:** | | **Cán bộ hướng dẫn:** |
| Tô Ngọc Khánh | 10520225 | PGS TS Vũ Thanh Nguyên |
| Phùng Đức Lâm | 10520253 | KS Lâm Vinh Đạo |
| **Đánh giá Khóa luận**   1. Về cuốn báo cáo:   Số trang Số chương  Số bảng số liệu Số hình vẽ  Số tài liệu tham khảo Sản phẩm  Một số nhận xét về hình thức cuốn báo cáo:  **………………………………………………………………………………………………………………………………………………………………………………………..……………………………………………………………………………………………..….………………………………………………………………………………………………………………………………………………………………………………**   1. Về nội dung nghiên cứu:   **…………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………**   1. Về chương trình ứng dụng:   **…………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………**   1. Về thái độ làm việc của sinh viên:   **…………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………**  **Đánh giá chung:**  **…………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………**  **Điểm từng sinh viên:**  Tô Ngọc Khánh**:………../10**  Phùng Đức Lâm**:………../10** | | |
|  | | |

. **Người nhận xét**

(Ký và ghi rõ họ tên)

|  |  |
| --- | --- |
| ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH  **TRƯỜNG ĐẠI HỌC**  **CÔNG NGHỆ THÔNG TIN** | **CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**  **Độc Lập - Tự Do - Hạnh Phúc** |
|  | *TP. HCM, ngày…..tháng…..năm……..* |
|  |  |

**NHẬN XÉT KHÓA LUẬN TỐT NGHIỆP**

**(CỦA CÁN BỘ HƯỚNG DẪN)**

|  |  |  |
| --- | --- | --- |
| **Tên khóa luận:** | | |
| **XÂY DỰNG GAME MOBILE ONLINE – TETRIS ONLINE** | | |
| **Nhóm SV thực hiện:** | | **Cán bộ phản biện:** |
| Tô Ngọc Khánh | 10520225 |  |
| Phùng Đức Lâm | 10520253 |  |
| **Đánh giá Khóa luận**   1. Về cuốn báo cáo:   Số trang Số chương  Số bảng số liệu Số hình vẽ  Số tài liệu tham khảo Sản phẩm  Một số nhận xét về hình thức cuốn báo cáo:  **………………………………………………………………………………………………………………………………………………………………………………………..……………………………………………………………………………………………..….………………………………………………………………………………………………………………………………………………………………………………**   1. Về nội dung nghiên cứu:   **…………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………**   1. Về chương trình ứng dụng:   **…………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………**   1. Về thái độ làm việc của sinh viên:   **…………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………**  **Đánh giá chung:**  **…………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………**  **Điểm từng sinh viên:**  Tô Ngọc Khánh**:………../10**  Phùng Đức Lâm**:………../10** | | |
|  | | |

. **Người nhận xét**

(Ký và ghi rõ họ tên)

# LỜI CẢM ƠN

Qua quá trình thực hiện khóa luận với đề tài **“Xây dựng Game Mobile Online – Tetris Online”**, chúng em đã biết thêm được nhiều kiến thức bổ ích về bài toán game mobile online, cùng với đó là kiến thức về kiến trúc, cơ chế hoạt động cũng như cách thức sử dụng một số engine, nền tảng hỗ trợ quá trình xây dựng 1 game mobile online.

Trong suốt thời gian từ lúc nhận đề tài cho tới khi hoàn thành khóa luận, chúng em luôn nhận được sự giúp đỡ tận tình từ các thầy cô trong khoa. Chúng em xin gửi lời cảm ơn sâu sắc tới quý thầy cô. Đặc biệt xin chân thành cảm ơn **PGS.TS Vũ Thanh Nguyên** và **KS. Lâm Vinh Đạo** – những người đã trực tiếp hướng dẫn và tạo mọi điều kiện để chúng em có thể hoàn thành khóa luận này.

Cuối cùng xin cảm ơn gia đình và bạn bè đã nhiệt tình giúp đỡ, động viên chúng em trong thời gian vừa qua.

Mặc dù đã rất cố gắng nhưng chắc chắn rằng khóa luận không thể nào tránh khỏi những thiếu sót. Vì vậy chúng em rất mong nhận được sự góp ý quý báu của các thầy cô để khóa luận hoàn chỉnh hơn.

Chúng em xin chân thành cảm ơn!

Nhóm thực hiện

**Tô Ngọc Khánh – Phùng Đức Lâm**

|  |  |
| --- | --- |
| ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH  **TRƯỜNG ĐẠI HỌC**  **CÔNG NGHỆ THÔNG TIN** | **CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**  **Độc Lập - Tự Do - Hạnh Phúc** |

# ĐỀ CƯƠNG CHI TIẾT

|  |  |
| --- | --- |
| **TÊN ĐỀ TÀI: Xây dựng game online mobile – Tetris Online** | |
| **Cán bộ hướng dẫn: PGS.Ts. Vũ Thanh Nguyên – Ks. Lâm Vinh Đạo** | |
| **Thời gian thực hiện:** Từ ngày………………..đến ngày…………… | |
| **Sinh viên thực hiện:**  **Tô Ngọc Khánh – 10520225**  **Phùng Đức Lâm – 10520253** | |
| **Nội dung đề tài:**   * Mục tiêu: * Nghiên cứu cách thức xây dựng và phát triển của một game mobile online * Xây dựng một game mobile online hoàn chỉnh * Đánh giá kết quả thực hiện * Phạm vi: * Nhóm phát triển Game Online cho nền tảng Android, bao gồm các tính năng như Chơi đơn, chơi nhiều người, hệ thống mua bán kỹ năng … * Đối tượng: * Đối tượng mà nhóm hướng đến là những người thích chơi game đơn thuần giải trí, giết thời gian nên game play sẽ đơn giản, quen thuộc với mọi người. * Phương pháp thực hiện: * Nghiên cứu các phương pháp phát triển game phổ biến, từ đó lựa chọn phương pháp phù hợp để phát triển game. * Kết quả mong đợi: * Hoàn thành một game mobile hoàn chỉnh về tính năng cũng như hấp dẫn về gameplay. | |
| **Kế hoạch thực hiện:**  25/09/2014 – 25/10/2014: Nghiên cứu các nền tảng và lựa chọn nền tảng phát triển.  25/10/2014 – 10/11/2014: Nghiên cứu công nghệ đã lựa chọn.  10/11/2014 – 10/12/2014: Phân tích, thiết kế hệ thống, cài đặt hệ thống  01/12/2014 – 20/12/2014: Test game, fix bug, cập nhật …  20/12/2014 – 05/01/2015: Viết báo cáo. | |
| **Xác nhận của CBHD**  (Ký tên và ghi rõ họ tên) | **TP. HCM, ngày….tháng …..năm…..**  **Sinh viên**  (Ký tên và ghi rõ họ tên) |

MỤC LỤC

[LỜI CẢM ƠN 8](#_Toc408095390)

[ĐỀ CƯƠNG CHI TIẾT 9](#_Toc408095391)

[Chương 1. MỞ ĐẦU 4](#_Toc408095392)

[1.1. Tổng quan về Game Mobile 4](#_Toc408095393)

[1.1.1. Khái niệm Game Mobile 4](#_Toc408095394)

[1.1.2. Tiềm năng thị trường Game Online 4](#_Toc408095395)

[1.1.3. Thị trường Game Việt Nam 4](#_Toc408095396)

[1.2. Bài toán đặt ra 6](#_Toc408095397)

[1.3. Phạm vi đề tài 6](#_Toc408095398)

[1.4. Đối tượng đề tài 6](#_Toc408095399)

[1.5. Yêu cầu đề tài 6](#_Toc408095400)

[Chương 2. NGHIÊN CỨU NỀN TẢNG CÔNG NGHỆ 7](#_Toc408095401)

[2.1. Mô hình Client - Server 7](#_Toc408095402)

[2.2. Netty Framework 8](#_Toc408095403)

[2.3. OpenFL 9](#_Toc408095404)

[2.4. Haxe 10](#_Toc408095405)

[2.5. Blocking và Non – Blocking 10](#_Toc408095406)

[2.5.1. Blocking 10](#_Toc408095407)

[2.5.2. Non – blocking 11](#_Toc408095408)

[2.6. NoSQL Database 12](#_Toc408095409)

[2.7. NoSQL và RDBM 12](#_Toc408095410)

[2.8. MongoDB 14](#_Toc408095411)

[Chương 3. Phân tích yêu cầu 16](#_Toc408095412)

[3.1. Sơ đồ đặc tả hệ thống 16](#_Toc408095413)

[3.1.1. Sơ đồ Use case 16](#_Toc408095414)

[3.1.2. Danh sách actor 16](#_Toc408095415)

[3.1.3. Danh sách use case 16](#_Toc408095416)

[3.1.4. Đặc tả Use case 17](#_Toc408095417)

[3.1.4.1. Use case Sign up 17](#_Toc408095418)

[3.1.4.2. Use case Login 17](#_Toc408095419)

[3.1.4.3. Use case Single play 18](#_Toc408095420)

[3.1.4.4. Use case Battle 18](#_Toc408095421)

[3.1.4.5. Use case Shop 19](#_Toc408095422)

[3.1.4.6. Use case Leaderboard 19](#_Toc408095423)

[3.2. Hiện thực hóa – sơ đồ tuần tự 20](#_Toc408095424)

[3.2.1. Use case Sign up 20](#_Toc408095425)

[3.2.2. Use case Login 21](#_Toc408095426)

[3.2.3. Use case Single play 22](#_Toc408095427)

[3.2.4. Use case Battle 23](#_Toc408095428)

[3.2.5. Use case Shop 24](#_Toc408095429)

[3.2.6. Use case Leaderboard 25](#_Toc408095430)

[Chương 4. Thiết kế hệ thống 26](#_Toc408095431)

[4.1. Thiết kế hệ thống 26](#_Toc408095432)

[4.1.1. Hệ thống chức năng của trò chơi 26](#_Toc408095433)

[4.1.2. Kiến trúc tổng quát của trò chơi 26](#_Toc408095434)

[4.2. Thiết kế Database 27](#_Toc408095435)

[4.3. Thiết kế Server 28](#_Toc408095436)

[4.3.1. Gói Server 28](#_Toc408095437)

[4.3.2. Gói Connection 28](#_Toc408095438)

[4.3.3. Gói Database 28](#_Toc408095439)

[4.3.4. Gói Player 29](#_Toc408095440)

[4.3.5. Gói Misc 29](#_Toc408095441)

[4.4. Thiết kế Client 29](#_Toc408095442)

[4.4.1. Gói Core.display 29](#_Toc408095443)

[4.4.2. Gói resource 30](#_Toc408095444)

[4.4.3. Gói Const 30](#_Toc408095445)

[4.4.4. Gói Data 30](#_Toc408095446)

[4.4.5. GameObject 31](#_Toc408095447)

[4.4.6. Popup 32](#_Toc408095448)

[4.4.7. Scene 32](#_Toc408095449)

[4.5. Thiết kế gói tin 33](#_Toc408095450)

[Chương 5. Thiết kế giao diện 34](#_Toc408095451)

[5.1. Danh sách các màn hình. 34](#_Toc408095452)

[5.2. Chi tiết các màn hình 35](#_Toc408095453)

[5.2.1. Đăng nhập/ Đăng ký 35](#_Toc408095454)

[5.2.1.1. Hình ảnh 35](#_Toc408095455)

[5.2.1.2. Mô tả 35](#_Toc408095456)

[5.2.2. Menu chính 36](#_Toc408095457)

[5.2.2.1. Hình ảnh 36](#_Toc408095458)

[5.2.2.1. Mô tả 36](#_Toc408095459)

[5.2.3. Vật phẩm/kỹ năng 37](#_Toc408095460)

[5.2.3.1. Hình ảnh 37](#_Toc408095461)

[5.2.3.1. Mô tả 38](#_Toc408095462)

[5.2.4. Tìm kiếm đối thủ 39](#_Toc408095463)

[5.2.4.1. Hình ảnh 39](#_Toc408095464)

[5.2.4.1. Mô tả 40](#_Toc408095465)

[5.2.5. Màn hình in-game 41](#_Toc408095466)

[5.2.5.1. Hình ảnh 41](#_Toc408095467)

[5.2.5.1. Mô tả 41](#_Toc408095468)

[5.2.6. Màn hình kết thúc game 42](#_Toc408095469)

[5.2.6.1. Hình ảnh 42](#_Toc408095470)

[5.2.6.1. Mô tả 42](#_Toc408095471)

[5.2.7. Bảng xếp hạng 43](#_Toc408095472)

[5.2.7.1. Hình ảnh 43](#_Toc408095473)

[5.2.7.1. Mô tả 43](#_Toc408095474)

[Chương 6. Môi trường lập trình và triển khai ứng dụng 45](#_Toc408095475)

[6.1. Môi trường lập trình 45](#_Toc408095476)

[6.2. Triển khai ứng dụng 45](#_Toc408095477)

[6.2.1. Yêu cầu ứng dụng 45](#_Toc408095478)

[6.2.2. Triển khai 45](#_Toc408095479)

[Chương 7. Kết luận và hướng phát triển 46](#_Toc408095480)

[7.1. Kết luận 46](#_Toc408095481)

[7.2. Hướng phát triển 47](#_Toc408095482)

[TÀI LIỆU THAM KHẢO 48](#_Toc408095483)

[PHỤ LỤC 49](#_Toc408095484)

DANH MỤC HÌNH VẼ

[Hình 2.1: Mô hình Client – Server 7](#_Toc408094856)

[Hình 2.2: Netty 8](#_Toc408094857)

[Hình 2.3: Kiến trúc Netty 8](#_Toc408094858)

[Hình 2.4: OpenFL 9](#_Toc408094859)

[Hình 2.5: Haxe 10](#_Toc408094860)

[Hình 2.6: NoSQL Database 12](#_Toc408094861)

[Hình 2.7: MongoDB 14](#_Toc408094862)

[Hình 5.1: Màn hình Đăng nhập/ Đăng ký. 35](#_Toc408094863)

[Hình 5.2: Màn hình Menu chính. 36](#_Toc408094864)

[Hình 5.3: Màn hình Vật phẩm/kỹ năng. 37](#_Toc408094865)

[Hình 5.4: Màn hình tìm kiếm đối thủ. 39](#_Toc408094866)

[Hình 5.5: Màn hình in-game. 41](#_Toc408094867)

[Hình 5.6: Màn hình kết thúc game. 42](#_Toc408094868)

[Hình 5.7: Màn hình bảng xếp hạng. 43](#_Toc408094869)

[Hình phụ lục 1: Xử lý ở Client 51](#_Toc408094870)

[Hình phụ lục 2: Xử lý ở Server 51](#_Toc408094871)

[Hình phụ lục 3: FlashDevelop 54](#_Toc408094872)

[Hình phụ lục 4: Benchmark của MongoDB và MySQL 57](#_Toc408094873)

DANH MỤC BẢNG

[Bảng 2.1: So sánh CSDL quan hệ và NoSQL 14](#_Toc408094874)

[Bảng 3.1: Bảng Actor 16](#_Toc408094875)

[Bảng 3.2: Bảng danh sách use case 17](#_Toc408094876)

[Bảng 3.3: Use case Sign up 17](#_Toc408094877)

[Bảng 3.4: Use case Sign in 18](#_Toc408094878)

[Bảng 3.5: Use case Single play 18](#_Toc408094879)

[Bảng 3.6: Use case Battle 19](#_Toc408094880)

[Bảng 3.7: Use case Shop 19](#_Toc408094881)

[Bảng 3.8: Use case Leaderboard 20](#_Toc408094882)

[Bảng 4.1: Document lưu trữ player 27](#_Toc408094883)

[Bảng 4.2: Document lưu trữ kỹ năng/vật phẩm 27](#_Toc408094884)

[Bảng 4.3: Gói server 28](#_Toc408094885)

[Bảng 4.4: Gói Connection 28](#_Toc408094886)

[Bảng 4.5: Gói Database 29](#_Toc408094887)

[Bảng 4.6: Gói Connection 29](#_Toc408094888)

[Bảng 4.7: Gói Connection 29](#_Toc408094889)

[Bảng 4.8: Gói Core.display 30](#_Toc408094890)

[Bảng 4.9: Gói resouce 30](#_Toc408094891)

[Bảng 4.10: Gói Const 30](#_Toc408094892)

[Bảng 4.11: Gói Data 31](#_Toc408094893)

[Bảng 4.12: Gói GameObject 31](#_Toc408094894)

[Bảng 4.13: Gói Gameplay 32](#_Toc408094895)

[Bảng 4.14: Gói Skill 32](#_Toc408094896)

[Bảng 4.15: Gói Popup 32](#_Toc408094897)

[Bảng 4.16: Gói Scene 33](#_Toc408094898)

[Bảng 5.1: Danh sách các màn hình 34](#_Toc408094899)

[Bảng 6.1: Yêu cầu ứng dụng 45](#_Toc408094900)

[Bảng phụ lục 2.1: Các ngôn ngữ có thể biên dịch ra được khi dùng Haxe 53](#_Toc408094901)

DANH MỤC SƠ ĐỒ

[Sơ đồ 3.1: Sơ đồ Use case 16](#_Toc408094902)

[Sơ đồ 3.2: Sơ đồ tuần tự Sign up 20](#_Toc408094903)

[Sơ đồ 3.3: Sơ đồ tuần tự Login 21](#_Toc408094904)

[Sơ đồ 3.4: Sơ đồ tuần tự Single play 22](#_Toc408094905)

[Sơ đồ 3.5: Sơ đồ tuần tự Battle 23](#_Toc408094906)

[Sơ đồ 3.6: Sơ đồ tuần tự Shop 24](#_Toc408094907)

[Sơ đồ 3.7: Sơ đồ tuần tự Leaderboard 25](#_Toc408094908)

[Sơ đồ 4.1: Tổng quan chức năng trò chơi 26](#_Toc408094909)

[Sơ đồ 4.2: Kiến trúc tổng quát của trò chơi 27](#_Toc408094910)

DANH MỤC TỪ VIẾT TẮT

|  |  |
| --- | --- |
| **Từ viết tắt** | **Ý nghĩa** |
| **CSDL** | **Cơ sở dữ liệu** |
| **NIO** | **Non – blocking IO** |
|  |  |
|  |  |
|  |  |
|  |  |

TÓM TẮT KHÓA LUẬN

Với sự phát triển mạnh mẽ của lĩnh vực di động, viễn thông ngày nay thì ngành công nghiệp sản xuất phần mềm, dịch vụ di động đã có một “mảnh đất màu mỡ” để phát triển không ngừng.

Khó luận tốt nghiệp này của nhóm sẽ giải quyết bài toán **Game mobile online**. Nội dung của khóa luận bao gồm

**Phần 1: Mở đầu**

Đặt vấn đề, trình bày lý do lựa chọn đề tài, mục đích, đối tượng cũng như phạm vi của đề tài.

**Phần 2: Nghiên cứu nền tảng công nghệ**

Phân tích, làm rõ thế nào là một game online, những điều gì cần lưu ý khi thực hiện game mobile online, những công nghệ cần thiết để xây dựng một game mobile online.

So sánh, lựa chọn một bộ framework, engine hỗ trợ xây dựng một game mobile online phù hợp nhất đề tài và hướng phát triển

**Phần 3: Phân tích yêu cầu**

Chương này tập trung khảo sát và phân tích, thiết kế hệ thống game server online.

Tập trung phân tích các chức năng của hệ thống cần phải có, quy trình hoạt động của các chức năng cụ thể như thế nào, đảm bảo đúng yêu cầu của đề tài.

**Phần 4: Thiết kế hệ thống**

Đưa ra mô hình kiến trúc hệ thống, thiết kế server, thiết kế cơ sở dữ liệu, thiết kế Client và các module chức năng cụ thể của trò chơi.

**Phần 5: Thiết kế giao diện**

Trình bày thiết kế, các màn hình, xử lý trên các màn hình của trò chơi

**Phần 6: Môi trường lập trình và triển khai ứng dụng**

Chi tiết môi trường lập trình cũng như triển khai ở môi trường thực tế của ứng dụng.

**Phần 7: Kết luận và hướng phát triển**

Trình bày kết quả đặt được trong quá trình thực hiện đồ án, các thử nghiệm với hệ thống.

Nhìn lại những gì đã làm được trong khóa luận tốt nghiệp và những gì còn chưa làm được. Đưa ra những ưu, nhược điểm và phương hướng phát triển đồ án trong tương lai.

MỞ ĐẦU

Game online trong những năm qua đã trở thành món ăn tinh thần không thể thiếu của không nhưng giới trẻ, mà còn rất nhiều thành phần trong xã hội. Nó cũng đóng góp một lượng thu nhập không hề nhỏ cho những đất nước có ngành công nghiệp game phát triển (8 tỷ USD – năm 2012 là con số mà ngành công nghiệp game Hàn Quốc đem lại cho quốc gia này - trong đó phần lớn là game online theo TechinAsia) . Cùng với sự phát triển như vũ bão của các thiết bị cầm tay, game online đã không còn gì xa lạ với các nền tảng mobile như Android, IOS hay Window phone …

Với sở thích làm game và cũng như chuẩn bị một lượng kiến thức trước khi ra trường. Nhóm đã lựa chọn đề tài **Game mobile online.** Sau một thời gian nghiên cứu và so sánh cách nền tảng, thì nhóm đã quyết định sử dụng bộ công cụ, engine hỗ trợ mới gồm có: Server sử dụng Netty framework và MongoDB là một NoSql Database. Client sử dụng Haxe và OpenFL. Với những nền tảng và công nghệ trên, nhóm đã cho ra một sản phẩm **Game mobile online** với tên gọi **Tetris Online** đáp ứng được những yêu cầu cơ bản của một game online trên nền mobile.

# MỞ ĐẦU

## Tổng quan về Game Mobile

### Khái niệm Game Mobile

Game mobile là từ dùng để chỉ những trò chơi giải trí được dùng trên các thiết bị di động, điện thoại thông minh, PDA, máy tính bảng, máy nghe nhạc đa phương tiện (MP4) hoặc máy tính xách tay. Điều này không bao gồm các trò chơi dành riêng cho các dòng máy, hệ thống máy chơi game cầm tay như Nitendo DS hoặc PlayStation Portable.

### Tiềm năng thị trường Game Online

Theo kết quả nghiên cứu thị trường cuối năm 2013 của công ty nghiên cứu thị trường Nielsen thì có đến 64% người dân Mỹ có điện thoại thông minh để chơi game, con số này ở Trung Quốc lên tới 70%. Game đã trở thành hạng mục được sử dụng phổ biến nhất trong các loại phần mềm dùng cho điện thoại thông minh.

### Thị trường Game Việt Nam

Việt Nam hiện là là thị trường game lớn nhất khu vực Đông Nam Á. Năm 2013, doanh thu đạt 237 triệu USD, đồng thời đứng thứ 6 khu vực châu Á.

**Game Mobile**

Chưa bao giờ game mobile ở Việt Nam lại phát triển mạnh như vậy sau cú hích Flappy bird. Năm 2014, doanh thu của thị trường ước đạt 6.000 tỷ đồng, trong đó game mobile chiểm khoảng 1.000 tỷ đồng – một con số không hề nhỏ.

Do chi phí để sản xuất game cài đặt trên PC khá cao nên các nhà phát triển game trong nước đang hướng đến sản xuất game chạy trên nền tảng iOS hoặc Android của smartphone. Cả nước hiện có khoảng 40 doanh nghiệp, studio tham gia sản xuất và phát hành game.

Các studio game của Việt Nam tập trung chủ yếu TP.HCM, Hà Nội, Đà Nẵng, có thể kế đến như: Colobox, Pine, Canvas Games Studio, Banana Studio..., có những studio game chỉ làm outsoursing (thuê ngoài) cho nhà sản xuất ngoại nhưng một số lại kiêm luôn vai trò sản xuất.

Nhìn vào bức tranh chung của ngành công nghiệp game thế giới, một nghiên cứu của Newzoo cho thấy, thị phần game mobile toàn cầu đang có xu hướng gia tăng. Cụ thể, trong năm 2013, game mobile chiếm 17,4% doanh thu (tương ứng 12,3 tỷ USD) và năm 2016, tỷ lệ này tăng 27,8% (khoảng 23 tỷ USD).

Riêng tại Việt Nam, từ thị trường không có gì, game chạy trên nền tảng iOS và Android của thiết bị smartphone ước tính đạt doanh thu 2 triệu USD/tháng trong năm 2012; 25,5 triệu USD vào năm 2013, nay dự báo đạt 35 triệu USD và đến năm 2016 là 52 triệu USD.

Thị trường mobile game của Việt Nam có tiềm năng để phát triển vì Việt Nam được đánh giá là thị trường có số người sử dụng internet lẫn sở hữu smartphone tăng nhanh chóng. Theo đó, Việt Nam đứng thứ hai thế giới về tốc độ tăng trưởng thiết bị smartphone với 17 triệu máy.

**Sự ra đời của Game online trên mobile là một bước ngoặt**

Với những quy định quản lý ngày càng chặt chẽ đối với game online trên máy tính, game online trên di điện thoại di động được coi là một hướng đi mới không ít tiềm năng với những nhà kinh doanh game. Để thu hút khách hang, hầu hết các nhà cung cấp đều miễn phí đăng ký cho người chơi. Trong quá trình chơi, người dung thường mua các vật phẩm để nâng cấp vũ khí, sức chiến đấu hoặc tính năng cho nhân vật của mình và phải trả chi phí dịch vụ dữ liệu cho nhà mạng.

Hiện nay, các dòng điện thoại cao cấp, smartphone, Iphone đã đang dần dần chiếm được thị phần người dùng di động. Sự ra đời của mạng 3G là điều kiện vô cùng thuận lợi cho sự phát triển của Game Online Mobile.Game Online Mobile ko chỉ chơi được trên một loại dòng máy, nó có khả năng hỗ trợ trên các nền tảng khác nhau: Java, Android, Iphone, thậm chí là HTML5 trên PC.

Game online cũng giải quyết được 2 bài toán mà Game offline đang mắc phải, đó là phân chia doanh thu và tình trạng crack game. Với cơ chế quản lý toàn bộ tài khoản người dùng, game play trên server, việc crack game đã gần như được giải quyết hoàn toàn.

## Bài toán đặt ra

Qua phân tích, đánh giá những tiềm năng, ưu nhược điểm của thị trường game Việt Nam cũng như thế giới. Thì nhóm đặt ra mục tiêu của khóa luận là xây dựng một Game Mobile Online.

Đặc điểm của một Game Mobile Online là do các giới hạn về mặt phần cứng, thiết bị cũng như đường truyền thường là Wifi, 3G thiếu ổn định, nên Game sẽ không quá lớn, nặng, nên dễ chơi hơn, hướng tới nhiều đối tượng hơn.

## Phạm vi đề tài

Nhóm phát triển Game Online cho nền tảng Android, bao gồm các tính năng như Chơi đơn, chơi nhiều người, hệ thống mua bán kỹ năng, bảng xếp hạng …

## Đối tượng đề tài

Đối tượng mà nhóm hướng đến là những người thích chơi game đơn thuần giải trí, giết thời gian nên game play sẽ đơn giản, quen thuộc với mọi người.

## Yêu cầu đề tài

Game đảm bảo game play đơn giản, phù hợp với nhiều đối tượng.

Game phải có sự cân bằng khi chơi chế độ nhiều người.

Giao diện trực quan, thân thiện.

Có một số cải thiện so với game play nguyên thủy.

# NGHIÊN CỨU NỀN TẢNG CÔNG NGHỆ

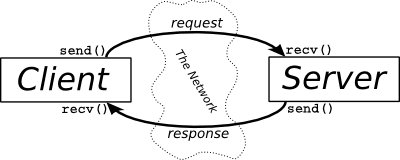
## Mô hình Client - Server

Các ứng dụng mạng thường hoạt động theo mô hình client/server như thư điện tử, truyền nhận tập tin, game trên mạng … Mô hình này gồm có một chương trình đóng vai trò là client và một chương trình đóng vai trò là server. Hai chương trình này sẽ giao tiếp với nhau thông qua mạng. Chương trình server đóng vai trò cung cấp dịch vụ. Chương trình này luôn luôn lắng nghe các yêu cầu từ phía Client, rồi tính toán và đáp trả kết quả tương ứng. Chương trình client cần một dịch vụ và gửi yêu cầu dịch vụ tới chương trình server và đợi đáp trả từ server. Như vậy, quá trình trao đổi dữ liệu giữa client/server bao gồm:

- Truyền một yêu cầu từ chương trình client tới chương trình server

- Yêu cầu được server xử lý

- Truyền đáp ứng cho client



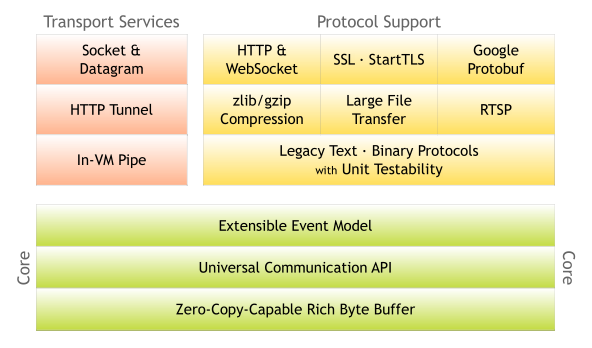
Hình 2.1: Mô hình Client – Server

Mô hình truyền tin này thực hiện truyền hai thông điệp qua lại giữa client và server một cách đồng bộ hóa. Chương trình server nhận được thông điệp từ client thì nó phát ra yêu cầu client chuyển sang trạng thái chờ (tạm dừng) cho tới khi client nhận được thông điệp đáp ứng do server gửi về. Mô hình client/server thường được cài đặt dựa trên các thao tác cơ bản là gửi (send) và nhận (receive).

## Netty Framework



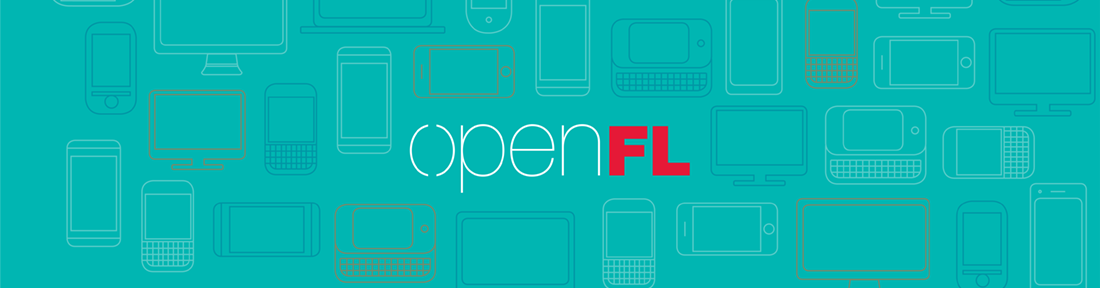
Hình 2.2: Netty

Netty là một NIO (Non – blocking IO) client – server framework dành cho việc phát triển ứng dụng mạng như hệ thống client – server sử dụng ngôn ngữ Java. Netty framework giúp đơn giản hóa và giảm thiểu thời gian của quá trình phát triển một ứng dụng client – server, bên cạnh đó nâng cao hiệu suất, vận hành và dễ dàng bảo trì. Ngoài ra, Netty cũng hỗ trợ các giao thứ HTTP, WebSocket, hay có thể tương thích với Google Protocol Buffers, hỗ trợ chuẩn bảo mật SSL/TSL …

Hình 2.3: Kiến trúc Netty

Xem thêm trong phần phụ lục.

## OpenFL



Hình 2.4: OpenFL

OpenFL là một framework mã nguồn mở và đa nền tảng dùng để phát triển các ứng dụng và video game. OpenFL sử dụng ngôn ngữ Haxe và được publish ra Flash movis, hay các ứng dụng độc lập trên trên các nền như Windows, Mac OS, iOS, Android, HTML 5 …

OpenFL được thiết kế tương tự như Adobe Flash Player, và cung cấp đầy đủ các API tương tự như vậy. Tệp SWF tạo ra bởi Adobe Flash Professinal cũng có thể được mở bẳng OpenFL.

Mốt số game nổi tiếng sử dụng OpenFL như: “Papers, Please”, “Rymdkapsel” hay “Evoland classic”.

Xem thêm trong phần phụ lục.

## Haxe



Hình 2.5: Haxe

Haxe là ngôn ngữ, bộ công cụ mã nguồn mở cấp cao đa nền tảng, và có thể biên dịch ra ứng dụng hoặc ra thẳng ngôn ngữ khác từ chỉ một code nguồn. Code viết bởi Haxe có thể dịch qua Flash Application, Javascript, Java, C#, C++, Python, Php, Apache CGI và Node.js.

Haxe chứa những tập lệnh hỗ trợ đa nền tảng như kiểu số, chữ, mảng, nhị phân hay một số kiểu định dạng file phổ biến. Haxe cũng chứa một số API đặc biệt dùng cho một số nền tảng khác. Haxe cũng là một phần của OpenFL framework – cho phép tạo ra các trò chơi đa nền tảng.

Haxe có nghĩa là “Has an X inside”.

Xem thêm trong phần phụ lục.

## Blocking và Non – Blocking

### Blocking

Theo cách truyền thống (thread-based) thì hãy tưởng tượng một ngân hàng đang áp dụng mô hình phục vụ: Phục vụ hoàn toàn một yêu cầu rồi mới chuyển sang yêu cầu khác. Trong đó nhân viên trong ngân hàng. Tương ứng mỗi nhân viên là một Thread. Và mỗi một yêu cầu tương ứng là 1 request đến server. Bạn yêu cầu là muốn gửi tiền vào ngân hàng. Bạn sẽ phải điền 1 số form như tên người gửi, số tài khoản của họ, số tiền cần gửi. v...v. Trong thời gian bạn điền thông tin cần rút tiền vào tờ khai. Cô nhân viên phải chờ bạn. Bạn đã "khóa" cô ấy không cho cô ấy phục vụ các khách hàng khác vì lúc đó cô ấy đang rảnh vì phải đợi bạn. Hành động đợi ở đây phần lớn là hành động vào/ra, truy suất file, hoặc đợi kết quả truy vấn SQL trong Webservice. Đó là cơ chế **Blocking**. Theo cách này. Nếu ngân hàng đang quá tải vì có quá nhiều người chờ được phục vụ. Thì ngân hàng chỉ còn một cách duy nhất là. Tăng thêm số nhân viên phục vụ lên. Ở trường hợp này trong ví dụ chúng ta là tăng số server phục vụ lên.

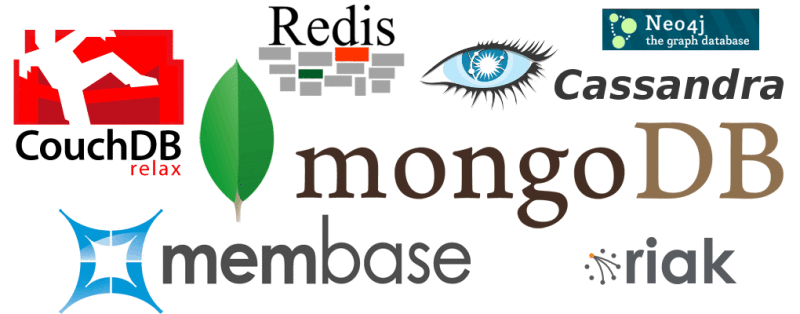
Vấn đề xảy ra ở đây ra. Khi tăng số lượng nhân viên lên để đáp ứng nhu cầu phục vụ khách hàng thì ngân hàng phải tăng chi phí (tiền để trả lương, mặt bằng văn phòng...) (tương ứng với việc tăng phần cứng máy chủ lên để đáp ứng). Như thế sẽ gây ra các vấn đề lãng phi, không tận dụng được nguồn nhân lực và chi phí.

### Non – blocking

Ở một ngân hàng khác, họ lại áp dụng theo một phương thức mới (eventdriven): Tận dụng mọi khả năng của tất cả các nhân viên khi họ rảnh và nhân viên khi có yêu cầu mới phục vụ.Tức là khi bạn có một yêu cầu muốn gửi tiền như ở trên. Cô nhân viên chỉ cần đưa bạn bút và giấy để bạn điền vào và bảo bạn hãy ngồi ở ghế chờ để điền xong thông tin gửi tiền. Trong khi đó cô ấy có thể phục vụ những vị khách tiếp theo. Ở đây bạn đã không "khóa" cô ấy lại. Và cô nhân viên tranh thủ lúc đợi bạn điền các thông tin. Cô ấy có thể làm việc khác. Thành ra ở đây không có hành động đợi vô nghĩa ở đây. Khi bạn điền xong thông tin, bạn có thể trở lại gặp cô ấy báo là đã hoàn thành. Cô ấy sẽ tiếp tục phục vụ bạn để bạn hoàn thành việc của mình. Đây là cơ chế **Non-Blocking.**

Bạn có thể thấy, theo mô hình áp dụng của ngân hàng này (event-driven), họ sẽ tận dụng được khoảng thời gian rỗi của nhân viên. Khiến cho việc một nhân viên có thể phục vụ nhiều khách hàng hơn so với ngân hàng dùng mô hình theo cách cũ ở trên. Nếu có quá tải. Bạn chắc chắn vẫn phải thêm nhân viên để đáp ứng kịp thời. Nhưng chắc chắn sẽ thêm ít nhân viên hơn. Tiết kiệm được rất nhiều tài nguyên. (Đây cũng là mô hình được áp dụng phổ biến ở các ngân hàng).

## NoSQL Database



Hình 2.6: NoSQL Database

NoSQL, viết tắt của non-relational, hoặc theo cách hiểu khác thì có nghĩa là Not only SQL (không chỉ là SQL). NoSQL đặc biệt nhấn mạnh đến mô hình lưu trữ cặp giá trị - khóa và hệ thống lưu trữ phân tán.

Hệ CSDL này có thể lưu trữ, xử lý từ lượng rất nhỏ đến hàng petabytes dữ liệu với khả năng chịu tải, chịu lỗi cao nhưng chỉ đòi hỏi về tài nguyên phần cứng thấp. NoSQL thiết kế đơn giản, nhẹ, gọn hơn so với RDBMs, thiết kế đặc biệt tối ưu về hiệu suất, tác vụ đọc-ghi, ít đòi hỏi về phần cứng mạnh và đồng nhất, dễ dàng thêm bớt các node không ảnh hưởng tới toàn hệ thống, ...

Xem thêm trong phần phụ lục.

## NoSQL và RDBM

Các RDBMs hiện tại đã bộc lộ những yếu kém như việc đánh chỉ mục một lượng lớn dữ liệu, phân trang, hoặc phân phối luồng dữ liệu media (phim, ảnh, nhạc, ...). Cơ sở dữ liệu quan hệ được thiết kế cho những mô hình dữ liệu nhỏ thường xuyên đọc viết trong khi các Social Network Services lại có một lượng dữ liệu cực lớn và cập nhật liên tục do số lượng người dùng quá nhiều ở một thời điểm. Thiết kế trên Distributed NoSQL giảm thiểu tối đa các phép tính toán, I/O liên quan kết hợp với batch processing đủ đảm bảo được yêu cầu xử lý dữ liệu của các mạng dịch vụ dữ liệu cộng đồng này. Facebook, Amazon là những ví dụ điểm hình.

Về cơ bản, các thiết kế của NoSQL lựa chọn mô hình lưu trữ tập dữ liệu theo cặp giá trị key-value. Khái niệm node được sử dụng trong quản lý dữ liệu phân tán. Với các hệ thống phân tán, việc lưu trữ có chấp nhận trùng lặp dữ liệu. Một request truy vấn tới data có thể gửi tới nhiều máy cùng lúc, khi một máy nào nó bị chết cũng không ảnh hưởng nhiều tới toàn bộ hệ thống. Để đảm bảo tính real time trong các hệ thống xử lý lượng lớn, thông thường người ta sẽ tách biệt database ra làm 2 hoặc nhiều database. Một database nhỏ đảm bảo vào ra liên tục, khi đạt tới ngưỡng thời gian hoặc dung lượng, database nhỏ sẽ được gộp (merge) vào database lớn có thiết kế tối ưu cho phép đọc (read operation). Mô hình đó cho phép tăng cường hiệu suất I/O - một trong những nguyên nhân chính khiến hiệu năng trở nên kém đi.

Xem thêm trong phần phụ lục.

|  |  |  |
| --- | --- | --- |
| **Tính năng** | CSDL quan hệ | NoSQL |
| Hiệu suất | Kém hơn  Có quan hệ giữa các bảng | Cực cao  Bỏ qua các ràng buộc quan hệ |
| Khả năng đọc – ghi | Kém do thiết kế đảm bảo sự vào – ra liên tục của dữ liệu | Tốt với mô hình sử lý theo lô (batch process) và tối ưu việc đọc ghi |
| Thay đổi số node trong hệ thống | Phải tắt cả hệ thống.Việc thay đổi số node phức tạp Không cần phải tắt cả hệ thống | Việc thay đổi số node thường khá đơn giản, không ảnh hưởng đến hệ thống |
| Phần cứng | Đòi hỏi cao về phần cứng | Đòi hỏi thấp hơn về giá trị và tính đồng nhất của phần cứng |

Bảng 2.1: So sánh CSDL quan hệ và NoSQL

## MongoDB



Hình 2.7: MongoDB

Đây là một cơ sở dữ liệu NoSQL thuộc loại cơ sở dữ liệu hướng văn bản mã nguồn mở, có hiệu năng, tính sẵn có và tính mở rộng cao được cung cấp và hổ trợ bởi 10gen. MongoDB được 10gen phát triển từ năm 2007, đến năm 2009, MongoDB trở thành một sản phẩm mã nguồn mở có giấy phép AGPL. MongoDB được viết bằng ngôn ngữ C++. MongoDB lưu trữ dữ liệu dạng BSON. Không giống như các cơ sở dữ liệu quan hệ lưu dữ cấu trúc dữ liệu theo các bảng, MongoDB lưu trữ cấu trúc dữ liệu thành văn bản dựa JSON với mô hình động (gọi là BSON) khiến cho việc tích hợp dữ liệu cho các ứng dụng trở nên dễ dàng và nhanh hơn.Với mục tiêu là kết hợp các điểm mạnh của mô hình khóa – giá trị (nhanh mà tính mở rộng cao) với mô hình dữ liệu quan hệ (giàu chức năng).

Mục tiêu chính của Mongo là giữ lại các thuộc tính thân thiện của SQL. Do đó các câu truy vấn khá giống với SQL nên MongoDB khá thích hợp cho các lập trình viên đã quen với ngôn ngữ truy vấn SQL. MongoDB có một khối lượng tính năng lớn và hiệu năng cao. Với các loại dữ liệu phong phú, nhiều truy vấn và việc giảm thời gian phát triển trong việc mô hình hóa các đối tượng.

MongoDB được sử dụng tốt nhất với nhu cầu cần truy vấn động, cần tốc độ nhanh cho một cơ sở dữ liệu lớn vì MongoDB ngoài tốc độ đọc nhanh ra thì tốc độ ghi của nó rất nhanh.

MongoDB hỗ trợ việc tìm theo trường, khoảng kết quả tìm và tìm theo cú pháp. Các truy vấn có thể trả về các trường được qui định trong văn bản và cũng có thể bao gồm các hàm Javascript mà người dùng chưa định nghĩa. Cũng giống như các cơ sở dữ liệu quan hệ, bất cứ một trường nào trong MongoDB đều được đánh chỉ mục. MongoDB còn có hổ trợ theo mô hình chủ - tớ (master - slave), mảnh, vùng dữ liệu (Sharding).

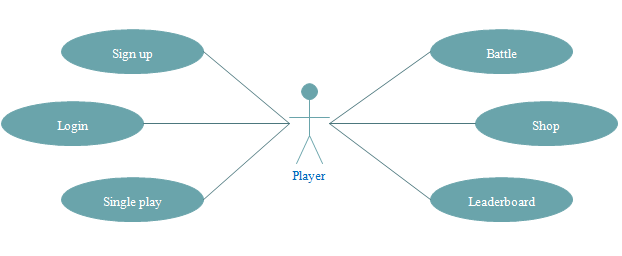
MongoDB sử dụng một quá trình xử lý để xử lý các yêu cầu về dữ liệu, quản lý định dạng dữ liệu, thực hiện các hoạt động quản lý bên dưới là mongod, đây là trình xử lý chính. Trong việc mở rộng theo chiều ngang sử dụng mô hình mảnh lưu trữ, MongoDB cung cấp dịch vụ xử lý các truy vấn từ tầng ứng dụng, xác định vị trí dữ liệu trong cụm các node phân mảnh được gọi là mongos.

Xem thêm trong phần phụ lục.

# PHÂN TÍCH YÊU CẦU

## Sơ đồ đặc tả hệ thống

### Sơ đồ Use case



Sơ đồ 3.1: Sơ đồ Use case

### Danh sách actor

|  |  |  |
| --- | --- | --- |
| STT | Tên Actor | Ý nghĩa/Ghi chú |
| 1 | Player | Người chơi |

Bảng 3.1: Bảng Actor

### Danh sách use case

|  |  |  |
| --- | --- | --- |
| STT | Tên Usecase | Ý nghĩa |
| 1 | Sign up | Đăng ký người chơi mới với server |
| 2 | Sign in | Đăng nhập vào trò chơi |
| 3 | Single play | Chế độ chơi đơn |
| 4 | Battle | Chế độ chơi với player khác |
| 5 | Shop | Mua các kỹ năng |
| 6 | Leaderboard | Bảng xếp hạng |

Bảng 3.2: Bảng danh sách use case

### Đặc tả Use case

#### Use case Sign up

|  |
| --- |
| Tóm tắt:   * Use case cho phép người dùng mới đăng ký, cung cấp các thông tin cần thiết cho việc tạo 1 tài khoản mới trên server. |
| Dòng sự kiện:   * Người chơi khởi động trò chơi * Hệ thống hiển thị màn hình Sign up * Người dùng cung cấp thông tin cần thiết và nhấn nút Sign up * Hệ thống đăng nhập vào trò chơi |
| Dòng sự kiện phụ:   * Không có |
| Tiền điều kiện:   * Người chơi khởi động trò chơi lần đầu tiên * Có kết nối internet |
| Hậu điều kiện:   * Trò chơi sẵn sàng cho những tác vụ tiếp theo |

Bảng 3.3: Use case Sign up

#### Use case Login

|  |
| --- |
| Tóm tắt:   * Use case login vào trò chơi, load dữ liệu cần thiết về thông tin của người chơi được lưu trên server. |
| Dòng sự kiện:   * Người chơi khởi động trò chơi * Hệ thống hiển thị màn hình Sign in * Người dùng cung cấp thông tin cần thiết và nhấn nút Sign in * Hệ thống đăng nhập vào trò chơi |
| Dòng sự kiện phụ:   * Không có |
| Tiền điều kiện:   * Người chơi đã có tài khoản trên server * Có kết nối internet |
| Hậu điều kiện:   * Trò chơi sẵn sàng cho những tác vụ tiếp theo |

Bảng 3.4: Use case Sign in

#### Use case Single play

|  |
| --- |
| Tóm tắt:   * Cho phép người chơi chơi ở chế độ đấu đơn, kể cả khi không có kết nối internet |
| Dòng sự kiện:   * Người chơi khởi động trò chơi * Vào menu chính * Người chơi chọn chế độ Single play * Trò chơi hiển thị chế độ đấu đơn |
| Dòng sự kiện phụ:   * Không có |
| Tiền điều kiện:   * Không có |
| Hậu điều kiện:   * Trò chơi sẵn sàng cho những tác vụ tiếp theo khi màn chơi kết thúc |

Bảng 3.5: Use case Single play

#### Use case Battle

|  |
| --- |
| Tóm tắt:   * Cho phép người chơi chơi ở chế độ đấu với các người chơi khác trên mạng. |
| Dòng sự kiện:   * Người chơi đăng nhập hoặc đăng ký mới * Vào menu chính * Người chơi chọn chế độ Battle * Trò chơi hiển thị chế độ Battle |
| Dòng sự kiện phụ:   * Không có |
| Tiền điều kiện:   * Có kết nối internet * Đã đăng ký mới hoặc đăng nhập vào trò chơi |
| Hậu điều kiện:   * Trò chơi sẵn sàng cho những tác vụ tiếp theo khi màn chơi kết thúc |

Bảng 3.6: Use case Battle

#### Use case Shop

|  |
| --- |
| Tóm tắt:   * Cho phép người chơi mua sắm các kỹ năng có thể sử dụng trong game |
| Dòng sự kiện:   * Người chơi đăng nhập hoặc đăng ký mới * Vào menu chính * Người chơi chọn Shop * Trò chơi hiển thị màn hình Shop |
| Dòng sự kiện phụ:   * Không có |
| Tiền điều kiện:   * Có kết nối internet * Đã đăng ký mới hoặc đăng nhập vào trò chơi |
| Hậu điều kiện:   * Trò chơi sẵn sàng cho những tác vụ tiếp theo |

Bảng 3.7: Use case Shop

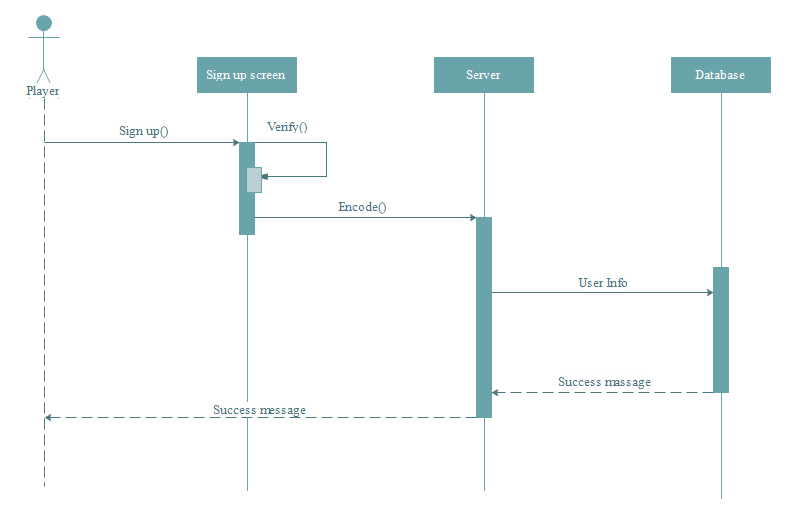
#### Use case Leaderboard

|  |
| --- |
| Tóm tắt:   * Cho phép người chơi có thể xem bảng xếp hạng các top player trong tuần, tháng ... |
| Dòng sự kiện:   * Người chơi đăng nhập * Vào menu chính * Người chơi chọn nút Leaderboard |
| Dòng sự kiện phụ:   * Không có |
| Tiền điều kiện:   * Có kết nối internet * Đã đăng nhập vào trò chơi |
| Hậu điều kiện:   * Trò chơi sẵn sàng cho những tác vụ tiếp theo khi màn chơi kết thúc |

Bảng 3.8: Use case Leaderboard

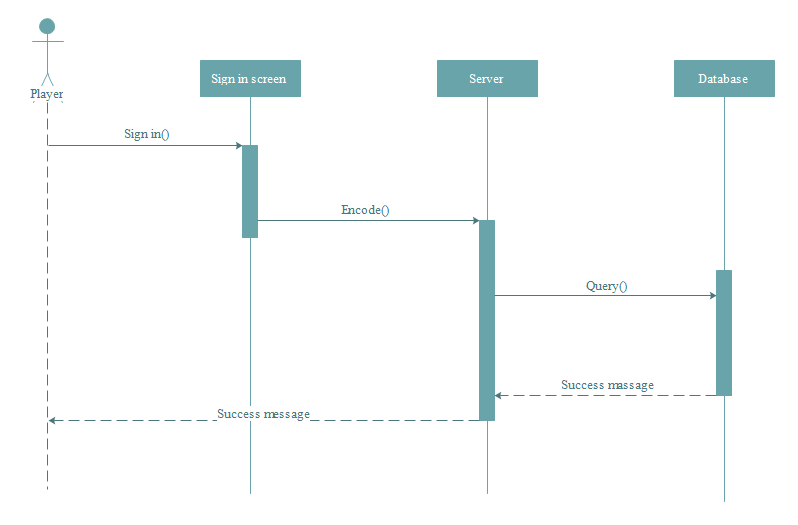
## Hiện thực hóa – sơ đồ tuần tự

### Use case Sign up



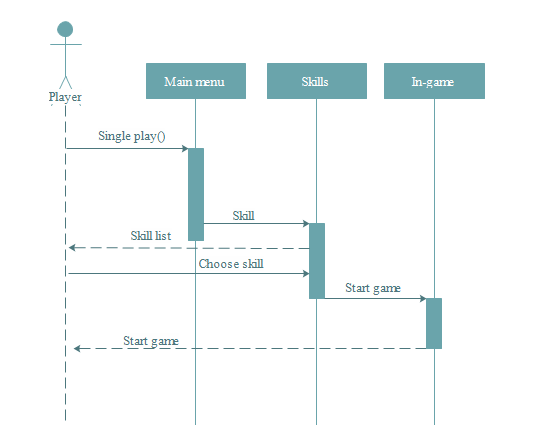
Sơ đồ 3.2: Sơ đồ tuần tự Sign up

### Use case Login



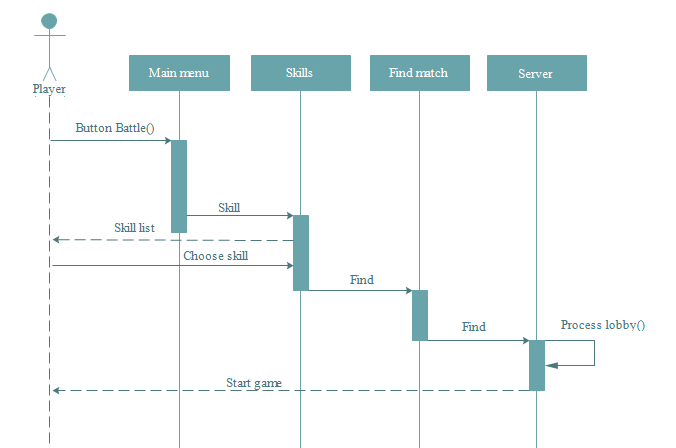
Sơ đồ 3.: Sơ đồ tuần tự Login

### Use case Single play



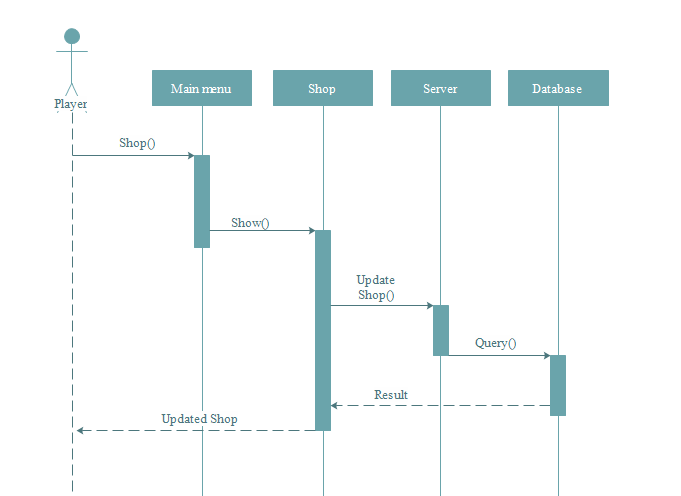
Sơ đồ 3.: Sơ đồ tuần tự Single play

### Use case Battle



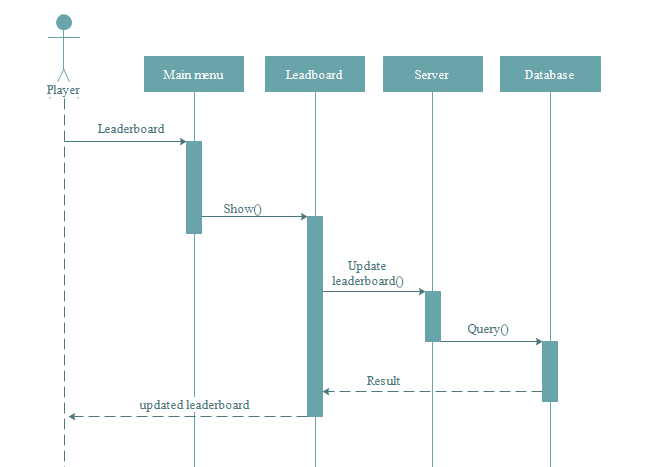
Sơ đồ 3.: Sơ đồ tuần tự Battle

### Use case Shop



Sơ đồ 3.: Sơ đồ tuần tự Shop

### Use case Leaderboard



Sơ đồ 3.: Sơ đồ tuần tự Leaderboard

# THIẾT KẾ HỆ THỐNG

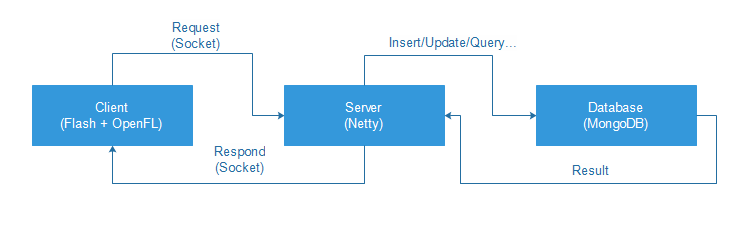
## Thiết kế hệ thống

### Hệ thống chức năng của trò chơi



Sơ đồ 4.: Tổng quan chức năng trò chơi

### Kiến trúc tổng quát của trò chơi



Sơ đồ 4.: Kiến trúc tổng quát của trò chơi

## Thiết kế Database

Với đặc thù một game nói chung, dữ liệu của các player chỉ là các thông tin về player đó, ngoài ra còn có thông tin về các trạng thái, chỉ số của game. Tuy nhiên, tất cả những dữ liệu đó đều không có quan hệ với nhau.

Với những đặc điểm trên nhóm đã chọn MongoDB, một NoSQL database để lưu dữ liệu. Trong MongoDB, dữ liệu được lưu dưới dạng một document – tương đương với một record trong RDBMs, các document sẽ tập hợp lại tạo thành một collection – tương đương với một table.

Một document lưu thông tin của player sẽ có dạng như sau:

|  |  |  |
| --- | --- | --- |
| STT | Tên key | Ý nghĩa |
| 1 | ID | ID của player |
| 2 | Name | Tên player |
| 3 | Gold | Tiền của player |
| 4 | Elo | Chỉ số Elo của player |
| 6 | Exp | Điểm kinh nghiệm của player |

Bảng 4.: Document lưu trữ player

Ngoài ra, còn có các document lưu lại giá trị của các vật phẩm có dạng:

|  |  |  |
| --- | --- | --- |
| STT | Tên key | Ý nghĩa |
| 1 | IDSkill | ID của kỹ năng/vật phẩm |
| 2 | NameSkill | Tên kỹ năng/vật phẩm |
| 3 | Cost | Giá của kỹ năng/vật phẩm |

Bảng 4.: Document lưu trữ kỹ năng/vật phẩm

## Thiết kế Server

### Gói Server

Định nghĩa, khởi tạo các giá trị cho server. Xử lý các kết nối đến và đi, các request từ phía client.

|  |  |  |
| --- | --- | --- |
| STT | Tên Class | Ý nghĩa |
| 1 | SocketServer | Thiết lập các config setting của hệ thống, khởi tạo server |
| 2 | SocketServerHandler | Xứ lý việc nhận các kết nối, request từ client gửi lên |

Bảng 4.: Gói server

### Gói Connection

Quản lý toàn bộ các kết nối đến server. Xử lý các luồng dữ liệu khi hệ thống hoạt động.

|  |  |  |
| --- | --- | --- |
| STT | Tên Class | Ý nghĩa |
| 1 | ConnectionManager | Quản lý tất cả các connection, player và điều hướng user, data |
| 2 | Lobby | Quản lý toàn bộ các xử lý liên quan đến 1 phòng chơi |

Bảng 4.: Gói Connection

### Gói Database

Tất các các thao tác, yêu cầu liên quan đến database.

|  |  |  |
| --- | --- | --- |
| STT | Tên Class | Ý nghĩa |
| 1 | MongoDBConnection | Quản lý tất các xử lý, thao tác, kết nối liên quan đến database |

Bảng 4.: Gói Database

### Gói Player

Định nghĩa user, quản lý logic game, xử lý request …

|  |  |  |
| --- | --- | --- |
| STT | Tên Class | Ý nghĩa |
| 1 | Player | Xử lý toàn bộ những logic, hoạt động liên quan đến những request mà client gửi lên |
| 2 | PlayerInformation | Thông tin của 1 player |
| 3 | Command | Các lệnh có thể nhận được từ client gửi lên server |

Bảng 4.: Gói Connection

### Gói Misc

|  |  |  |
| --- | --- | --- |
| STT | Tên Class | Ý nghĩa |
| 1 | Misc | Chứa 1 số hàm thường xuyên sử dụng |

Bảng 4.: Gói Connection

## Thiết kế Client

### Gói Core.display

Chứa các class có chức năng hiển thị và quản lý các đối tượng được vẽ ra màn hình. Trong đó có các gói nhỏ như: popup, scene, dialog, layer, note, screen

|  |  |  |
| --- | --- | --- |
| STT | Tên Class | Ý nghĩa |
| 1 | ScreenBase | Lớp screen cơ bản, dùng để kiểm tra và xữ lý các response từ sever |
| 2 | ScreenView | Lớp Screen View cơ bản, Chứa , quản lý, xữ lý và vẽ các đối tượng (Sprite) lên màn hình |
| 3 | ScreenID | Khai báo cac Scene, Dialog, popup của game. |
| 4 | NoteObject | Hiển thị 1 thông báo lên màn hình |

Bảng 4.: Gói Core.display

### Gói resource

Khái báo, quản lý các tài nguyên của game, Sound, sprite, animate, bitmap

|  |  |  |
| --- | --- | --- |
| STT | Tên Class | Ý nghĩa |
| 1 | Defines | Khai báo các ID của tài nguyên |
| 2 | ResourcePath | Đường dẫn của tài nguyên |
| 3 | ResourceManager | Khai bào và Quản lý tài nguyên |

Bảng 4.: Gói resouce

### Gói Const

Quản lý data của game

|  |  |  |
| --- | --- | --- |
| STT | Tên Class | Ý nghĩa |
| 1 | ExploringCache | Ghi và đọc cache để lưu thông tin của user |
| 2 | HighScoreInfo | Data high score của boss |
| 3 | SkillInfo | Data của skill (Tên, mode, giá,…) |
| 4 | Const | Đọc, ghi, khởi tạo 1 số data |

Bảng 4.: Gói Const

### Gói Data

Trong game sẻ tạo các đối tượng static, dùng để quản lý, xữ lý các data đang được sử dụng

|  |  |  |
| --- | --- | --- |
| STT | Tên Class | Ý nghĩa |
| 1 | UserInfo | Chứa thông tin của người chơi |
| 2 | DTSkill | Data skill của người chơi |
| 3 | DTGamePlay | Chứa các data dùng để xữ lý màn hình game play |
| 4 | DTPVP | Chứa thông tin người chơi và đối thủ khi đang thi đấu game |

Bảng 4.: Gói Data

### GameObject

Chứa các đối tượng của game

**Brick:** Các đối tượng liên quan tới viên gạch (Gạch, khối gạch, Loại gạch, Hướng gạch, Trạng thái)

|  |  |  |
| --- | --- | --- |
| STT | Tên Class | Ý nghĩa |
| 1 | Brick | Viên gạch (Loại gạch) |
| 2 | Block | Khối gạch (Gồm 4 viên gạch) (Loại gạch, Hướng gạch) |
| 3 | I, LL, LR, O, T, ZL, ZR | Các loại khối gạch |

Bảng 4.: Gói GameObject

**GamePlay:** Các đối tượng của màn hình game play (Màn chơi, HudLeft, HudRight, Score, Time, Background, gameControll, header, OneTouch)

|  |  |  |
| --- | --- | --- |
| STT | Tên Class | Ý nghĩa |
| 1 | Board | Màn chơi (10x22) dùng để hiển thị các viên gạch |
| 2 | HudLeft | Hiển thị các viên gạch đang giữ |
| 3 | HudRight | Hiển thị các viên gạch đang đợi |
| 4 | Time | Hiển thị và xữ lý thời gian |
| 5 | GameControler | Xữ lý game (Di chuyển, giữ) |

Bảng 4.: Gói Gameplay

**Skill:** Hiển thị skill, thông tin, chose skill

|  |  |  |
| --- | --- | --- |
| STT | Tên Class | Ý nghĩa |
| 1 | SkillDisplay | Hiển thị hình ảnh của skill |
| 2 | SkillInfo | Hiển thị thông tin của skill (Giá, Tên) |

Bảng 4.: Gói Skill

### Popup

Chứa các đối tượng popup của game

|  |  |  |
| --- | --- | --- |
| STT | Tên Class | Ý nghĩa |
| 1 | LoginPopup | Đăng ký khi lần đầu chơi |
| 2 | StaPvEPopup | Thông báo khi đánh xong 1 trận single |
| 3 | StaPvPPopup | Thông báo khi đánh xong 1 trận battle |

Bảng 4.: Gói Popup

### Scene

Chứa các màn hình của game

|  |  |  |
| --- | --- | --- |
| STT | Tên Class | Ý nghĩa |
| 1 | Battle | Màn hình của game battle |
| 2 | Battle Room | Màn hình tìm kiếm trận đấu |
| 3 | Chose Skill | Màn hình chose skill |
| 4 | High Score | Màn hình High score |
| 5 | Home | Màn hình chính |
| 6 | Loading | Màn hình loading |
| 7 | Single play | Màn hình của game signle |
| 8 | hudTop | Hiện thị thông tin của user |

Bảng 4.: Gói Scene

## Thiết kế gói tin

Gói tin truyền dữ liệu giữa client và server sử dụng ChannelBuffer của Netty có cấu trúc như sau:

**Phần HeaderPackage:**

* Length: kiểu short, độ dài của gói tin truyền đi
* Cmd: tức command, là lệnh mà server hay client phải xử lý

**Phần Data:**

Tùy thuộc và mỗi kiểu dữ liệu gửi đi sẽ có phần data khác nhau, vd một gói tin đăng nhập sẽ có phần data với chỉ thông tin đó là Id của người chơi đó được gửi lên:

* Id: kiểu string, id của người chơi muốn đăng nhập

# THIẾT KẾ GIAO DIỆN

## Danh sách các màn hình.

|  |  |  |
| --- | --- | --- |
| STT | Tên màn hình | Ý nghĩa |
| 1 | Đăng nhập/ Đăng ký | Màn hình sử dụng để đăng ký, đăng nhập vào game |
| 2 | Menu chính | Chứa tất các các danh phục của game như single play, battle … |
| 3 | Vật phẩm/kỹ năng | Người chơi có thể lựa chọn, mua thêm kỹ năng trang bị ở màn hình này |
| 4 | Tìm kiếm đối thủ | Tìm kiếm đối thủ ở chế độ chơi multiplayer |
| 5 | Màn hình in-game | Giao diện chính khi đang chơi game |
| 6 | Màn hình kết thúc game | Thông báo cho người chơi biết các thông số cần thiết sau game vừa chơi xong |
| 7 | Bảng xếp hạng | Xếp hạng người chơi có điểm số cao nhất |

Bảng 5.: Danh sách các màn hình

## Chi tiết các màn hình

### Đăng nhập/ Đăng ký

#### Hình ảnh



Hình 5.1: Màn hình Đăng nhập/ Đăng ký.

#### Mô tả

* Chức năng: giúp người chơi đăng ký mới hoặc đăng nhập vào trò chơi.
* Các thành phần xử lý trên màn hình:
  + Nút đăng nhập:

### Menu chính

#### Hình ảnh



Hình 5.2: Màn hình Menu chính.

#### Mô tả

* Chức năng: giúp người chơi lựa chọn các danh mục chính trong trò chơi.
* Các thành phần xử lý trên màn hình:
  + Nút “Single player”: vào chế độ chơi đơn
  + Nút “Battle”: vào chế độ chơi multiplayer
  + Nút “”: …

### Vật phẩm/kỹ năng

#### Hình ảnh



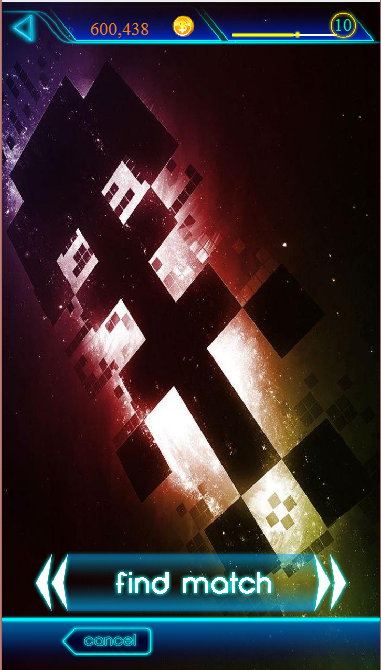
Hình 5.3: Màn hình Vật phẩm/kỹ năng.

#### Mô tả

* Chức năng: giúp người chơi lựa chọn các kỹ năng, hoặc mua mới các kỹ năng cần thiết khi tham gia trò chơi.
* Các thành phần xử lý trên màn hình:
  + Nút “Back”: quay lại màn hình chính
  + Nút “Play”: vào trò chơi
  + Các nút “Skill 1”, “Skill 2”, “Skill 3” : lựa chọn skill muốn thay đổi hoặc mua mới
  + Các nút mua kỹ năng: giúp người chơi mua kỹ năng mong muốn.
  + Nút “Back”: quay lại màn hình trước đó.

### Tìm kiếm đối thủ

#### Hình ảnh



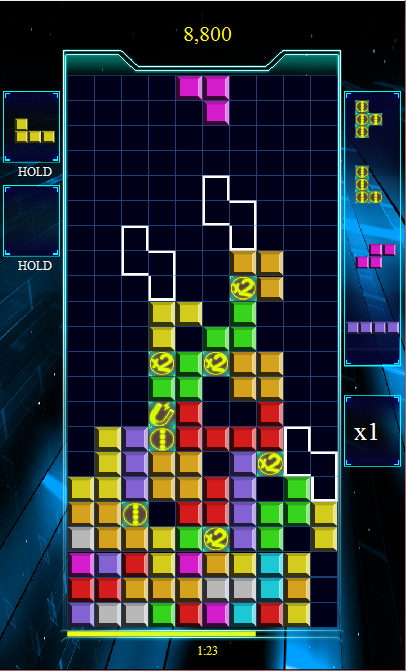
Hình 5.4: Màn hình tìm kiếm đối thủ.

#### Mô tả

* Chức năng: giúp người chơi tìm kiếm đối thủ trong chế độ multiplayer của trò chơi.
* Các thành phần xử lý trên màn hình:
  + Nút “Find match”: tìm kiếm người chơi
  + Nút “Cancel”: hủy bỏ tìm kiếm
  + Nút “Back”: quay lại màn hình trước đó.

### Màn hình in-game

#### Hình ảnh



Hình 5.5: Màn hình in-game.

#### Mô tả

### Màn hình kết thúc game

#### Hình ảnh



Hình 5.6: Màn hình kết thúc game.

#### Mô tả

* + Nút “Continue”: đóng màn hình kết thúc game.
  + Nút “Back”: quay lại màn hình trước đó.

### Bảng xếp hạng

#### Hình ảnh



Hình 5.7: Màn hình bảng xếp hạng.

#### Mô tả

* + Nút “Play”: Vào chế độ battle
  + Nút “Back”: quay lại màn hình trước đó.

# MÔI TRƯỜNG LẬP TRÌNH VÀ TRIỂN KHAI ỨNG DỤNG

## Môi trường lập trình

* Eclipse Luna 4.4.1
* Netty 3.2
* Haxe 3.1.3
* Wikitude SDK 3.2.1
* Wikitude Phonegap Plugin

## Triển khai ứng dụng

### Yêu cầu ứng dụng

|  |  |
| --- | --- |
| Android | Android 2.3, SDK=9 (nên 4.0 trở lên)  GPS and / or network positioning  Accelerometer  High resolution devices (hdpi)  Rear-facing camera  OpenGL 2.0  Devices with a capable CPU (hỗ trợ armv7a and NEON) |

Bảng 6.1: Yêu cầu ứng dụng

### Triển khai

* Moto G: Ứng dụng chạy bình thường.
* Oppo Finder : Ứng dụng chạy bình thường.

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## Kết luận

Trong thời gian thực hiện đề tài “**Xây dựng Game Mobile Online – Tetris Online”.** Nhóm đã tìm hiểu và biết đến những kiến trúc, nền tảng, engine thông dụng mà ngành công nghiệp game hiện nay (nhất là tại Việt Nam) đang sử dụng. Qua đó, nhóm đã lựa chọn ngôn ngữ Haxe với framework OpenFL ở phía client và lập trình Java Socket với bộ framework Netty ở phía server.

Framework trên có thể không mạnh mẽ được bằng một số framework khác nhưng bù lại, nó có một số điểm mạnh như sau:

**Về phía Server:**

* Netty là một bộ framework, thư viện khá dễ sử dụng, và đặc biệt phù hợp cho những dự án lớn, đòi hỏi tốc độ xử lý cao
* Tính linh hoạt cao, tương thích với nhiều giao thức truyền nhận dữ liệu
* Khả năng bảo mật tốt, hỗ trợ SSL/TLS và StartTLS
* Miễn phí, mã nguồn mở, cộng đồng sử dụng đông đảo.

**Về phía Client:**

* OpenFL và Haxe là bộ framework khá ổn định, có một cộng đồng khá lớn trên thế giới
* Khả năng build ra sản phẩm cũng như source code của Haxe rất phong phú về đầu ra, trong khi chỉ cần một source code có thể build ra nhiều ngôn ngữ khác nhau.
* Khả năng, tốc độ compile nhanh hơn nhiều so với bộ công cụ của Adobe
* Miễn phí, mã nguồn mở.

Mặc dù thời gian làm khóa luận không thực sự được nhiều như mong muốn nhưng nhóm cuối cùng cũng đã cho ra sản phẩm là game mobile online với lối chơi đơn giản, gần gũi nhưng không kém phần hấp dẫn với chế độ Battle cùng với nhiều kỹ năng, vật phẩm mới lạ.

Cùng với gameplay thì đồ họa của trò chơi cũng là một điểm cộng. Nhóm đã bỏ công sức khá lớn cho việc đầu tư vào đồ họa. Trò chơi có đồ họa bắt mắt, và khá chuyên nghiệp, ngoài ra, việc bố trí các nút, luồng hoạt động của trò chơi dễ sử dụng và làm quen cũng là điểm rất tích cực mà nhóm đạt được.

Tuy có gắng nhưng cũng không tránh khỏi một số thiếu sót như chưa có chế độ chơi theo nhiệm vụ. Cốt truyện chưa được thực hiện, một số kỹ năng trong game chưa thực sự cân bằng, hay bảng xếp hạng còn đơn giản, chưa được chi tiết.

## Hướng phát triển

Trong tương lai, nhóm định hướng phát triển với các mục tiêu sau:

* Phát triển hệ thống nhiệm vụ đơn của trò chơi
* Hệ thống lại các kỹ năng, vật phẩm in-game nhằm mục tiêu cân bằng giữa các người chơi.
* Thương mại hóa sản phẩm bằng việc bán kỹ năng, vật phẩm trong game

TÀI LIỆU THAM KHẢO

Theo chuẩn IEEE

PHỤ LỤC

1. **Netty**

Netty là một NIO (Non – blocking IO) client – server framework dành cho việc phát triển ứng dụng mạng như hệ thống client – server sử dụng ngôn ngữ Java. Netty framework giúp đơn giản hóa và giảm thiểu thời gian của quá trình phát triển một ứng dụng client – server, bên cạnh đó nâng cao hiệu suất, vận hành và dễ dàng bảo trì. Ngoài ra, Netty cũng hỗ trợ các giao thứ HTTP, WebSocket, hay có thể tương thích với Google Protocol Buffers, hỗ trợ chuẩn bảo mật SSL/TSL …

Netty là một bộ thư viện, API dược sử dụng với mục đích nhắm tới là xây dựng ứng dụng hệ thống mạng, và dịch vụ. Điểm khác biệt của Netty là sử dụng chủ yếu API không đồng bộ. Đa số các hàm được sử dụng đều là không đồng bộ, điều đó có nghĩa là không có nghĩa là các giá trị trả về cũng như lời gọi hàm không xảy ra ngay lập tức. Giá trị trả về có thể được trả về trong luồng khác chứ không phải luồng thực hiện lời gọi hàm. Dưới đây là sự khác biệt cơ bản của API thông thường và API không đồng bộ.

**API thông thường**

public int getWidgetCount();

Sau khi thread thực hiện hàm này một khoảng thời gian thì một giá trị int sẽ được trả về.

**API không đồng bộ**

public WidgetCountListener myListener = new WidgetCountListener() {

public void onWidgetCount(int widgetCount) {

...... do your thing with the widget count

}

};

Ở đây, hàm onWidgetCount sẽ không thực hiện ngay lập tức, mà nó sẽ được thực hiện khi lắng nghe được sự kiện WidgetCount trả về. Đây một một trong những khía cạnh quan trọng của một ứng dụng hiệu suất cao. Máy con sẽ không bị “kẹt” những lúc phải chờ đợi response từ server, thay vào đó nó sẽ được thông báo khi có response trả về.

* 1. **ChannelBuffer**

ChannelBuffer chính là lớp giúp truyền dữ liệu trong Netty. ChannelBuffer là một mảng ngẫu nhiên hoặc có thứ tự các byte dữ liệu có thể truy cập được. Nếu ứng dụng chỉ sử dụng truyền dữ liệu kiểu byte thì chúng ta rất đơn giản để sử dụng. Nếu ứng dụng truyền một đối tượng hoặc kiểu dữ liệu đặc biệt thì bắt buộc client và server phải sử dụng các lớp encode và decode để truyền được dữ liệu.

* 1. **Channel**

Channel chính là lớp giúp vận chuyển ChannelBuffer từ client tới server hay ngược lại. Một kết nối từ client tới server đó chính là một channel. Channel có nhiệm vụ chính đó là đọc, ghi, hay kết nối …

Các phương thức trong lớp Channel thuộc hai loại:

* Đồng bộ: là các phương thức thông thường, cung cấp thông tin về channel đó
* Không đồng bộ: các phương thức đọc, ghi, đứt kết nối …

Những phương thức không đồng bộ sẽ được trả về kiểu ChannelFuture, nó sẽ được trả về khi có kết quả. Interface sử dụng để hiện thực hóa sử lý khi nhận sự kiện trả về đó là ChannelFutureListener.

**Cách tạo một Channel:**

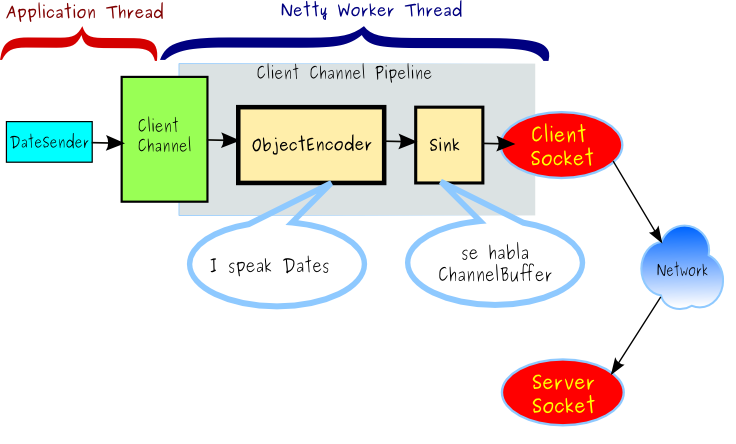
Channel không được tạo trực tiếp mà nó được tao thông qua ChannelFactory. Sẽ có một số kiểu Channel mà Netty hỗ trợ như:

* TCP NIO Channels
* UDP NIO Channels
* TCP OIO Channels
* UDP OIO Channels

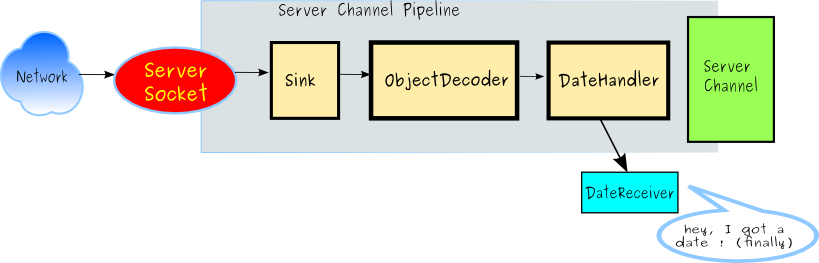
Tuy nhiên trong giới hạn của đề tài, chúng ta sẽ chỉ đề cập đến TCP NIO Channels vì đó là kiểu Channel nhóm sử dụng để hoàn thành đề tài.

Trong TCP NIO Channels sẽ có kiểu thread:

* Boss thread: boss thread Netty sử dụng trong việc tạo, bind socket, sau đó truyền lại worker thread. Trong một server sẽ có duy nhất một boss thread lắng nghe ở mỗi socket. Còn trong client thì sẽ chỉ có duy nhất một boss thread.
* Worker thread: chỉ có ở server. Nó thực hiện tất cả các tác vụ I/O không đồng bộ.
  1. **Cách thức truyền dữ liệu**



Hình phụ lục 1: Xử lý ở Client



Hình phụ lục 2: Xử lý ở Server

Trên đây ta có thể thấy được tổng quát quá trình gửi và nhận dữ liệu xảy ra trong Netty.

Ở client, chúng là có một đối tượng là DateSender cần được gửi. DateSender được truyền vào Channel phía client, client sẽ sử dụng lớp ObjectEncode, chuyển dữ liệu từ kiểu DataSender sang kiểu ChannelBuffer. Dữ liệu này sẽ được chuyển vào ChannelHandler cuối cùng, và được gửi tới server thông qua mạng

Ở phía server thì sẽ ngược lại, dữ liệu được nhận từ ChannelPipeline sẽ được ObjectDecode decode từ ChannelBuffer sang kiểu dữ liệu được gửi đi từ client.

1. **OpenFL và Haxe**
   1. **Haxe là gì**

Haxe là ngôn ngữ mã nguồn mở cấp cao đa nền tảng, và có thể biên dịch ra ứng dụng hoặc ra thẳng ngôn ngữ khác từ chỉ một code nguồn. Code viết bởi Haxe có thể dịch qua Flash Application, Javascript, Java, C#, C++, Python, Php, Apache CGI và Node.js.

Haxe chứa những tập lệnh hỗ trợ đa nền tảng như kiểu số, chữ, mảng, nhị phân hay một số kiểu định dạng file phổ biến. Haxe cũng chứa một số API đặc biệt dùng cho một số nền tảng khác. Haxe cũng là một phần của OpenFL framework – cho phép tạo ra các trò chơi đa nền tảng.

|  |  |  |
| --- | --- | --- |
| Tên | Kiểu output | Nền tảng sử dụng |
| Javascript | Sourcecode | Browser, Desktop, Mobile, Server |
| Neko | Bytecode | Desktop, Server |
| PHP | Sourcecode | Server |
| Python | Sourcecode | Desktop, Server |
| C++ | Sourcecode | Desktop, Mobile, Server |
| Actionscript 3 | Sourcecode | Browser, Desktop, Mobile |
| Flash | Bytecode | Browser, Desktop, Mobile |
| Java | Sourcecode | Desktop, Server |
| C# | Sourcecode | Desktop, Mobile, Server |

Bảng phụ lục 2.: Các ngôn ngữ có thể biên dịch ra được khi dùng Haxe

Nhiều công ty lớn trên thế giới sử dụng Haxe để xây dựng app, game như Nickelodeon, TiVo, Zynga, Prezi.

* 1. **Hello world với Haxe**

Chúng ta có chương trình “Hello world” sử dụng Haxe như sau:

class HelloWorld

{

static public function main():Void

{

trace("Hello World");

}

}

Chương trình sẽ in chữ “Hello World”.

* 1. **OpenFL**

OpenFL là một framework mã nguồn mở và đa nền tảng dùng để phát triển các ứng dụng và video game. OpenFL sử dụng ngôn ngữ Haxe và được publish ra Flash movis, hay các ứng dụng độc lập trên trên các nền như Windows, Mac OS, iOS, Android, HTML 5 …

OpenFL được thiết kế tương tự như Adobe Flash Player, và cung cấp đầy đủ các API tương tự như vậy. Tệp SWF tạo ra bởi Adobe Flash Professinal cũng có thể được mở bẳng OpenFL.

* 1. **Tạo project với OpenFL**
     1. **Cài đặt OpenFL**

Đầu tiên chúng ta cần cài đặt Haxe và Neko. OpenFL sử dụng Haxe trong quá trình build, còn Neko sử dụng trong command-lines. Chỉ cần download đúng phiên bản hệ điều hành và cài đặt.

Sau đó chạy lệnh:

haxelib install openfl

haxelib run openfl setup

* + 1. **Lựa chọn Code Editor**

Sau khi đã cài dặt Haxe và Neko, chúng ta cần một IDE. Việc sử dụng một IDE tốt sẽ giúp chúng ta có thể lập trình một cách nhanh chóng và thuận tiện nhất.

IDE đề nghị của OpenFL đó chính là FlashDevelop.



Hình phụ lục 3: FlashDevelop

FlashDevelop là bộ IDE mạnh mẽ, giúp người dùng dễ dàng làm quen. Trong OpenFL có chứa sắn file project là FlashDevelop.

* + 1. **Tạo project mới**

Để tạo một project mới ta dùng lệnh sau:

openfl create project HelloWorld

OpenFL sẽ tự động sinh ra một project rỗng, có thể tùy ý sử dụng.

1. **NoSQL Database**

NoSQL, viết tắt của non-relational, hoặc theo cách hiểu khác thì có nghĩa là Not only SQL (không chỉ là SQL). NoSQL đặc biệt nhấn mạnh đến mô hình lưu trữ cặp giá trị - khóa và hệ thống lưu trữ phân tán.

Hệ CSDL này có thể lưu trữ, xử lý từ lượng rất nhỏ đến hàng petabytes dữ liệu với khả năng chịu tải, chịu lỗi cao nhưng chỉ đòi hỏi về tài nguyên phần cứng thấp. NoSQL thiết kế đơn giản, nhẹ, gọn hơn so với RDBMs, thiết kế đặc biệt tối ưu về hiệu suất, tác vụ đọc-ghi, ít đòi hỏi về phần cứng mạnh và đồng nhất, dễ dàng thêm bớt các node không ảnh hưởng tới toàn hệ thống, ...

* 1. **Đặc điểm của NoSQL**

- Phi quan hệ (hay không ràng buộc): Các mối quan hệ giữa các bảng trong cơ sở dữ liệu quan hệ (RDBM) sử dụng mô hình gồm 2 loại khóa: khóa chính và khóa phụ (primary key + foreign key) để ràng buộc dữ liệu nhằm thể hiện tính nhất quán dữ liệu từ các bảng khác nhau. Non-relational là khái niệm không sử dụng các ràng buộc dữ liệu cho nhất quán dữ liệu.

- Lưu trữ phân tán: mô hình lưu trữ phân tán các tập tin hoặc dữ liệu ra nhiều máy khác nhau trong mạng LAN hoặc Internet dưới sự kiểm soát của phần mềm.

- Nhất quán cuối: tính nhất quán của dữ liệu không cần phải đảm bảo ngay tức khắc sau mỗi phép ghi. Một hệ thống phân tán chấp nhận những ảnh hưởng theo phương thức lan truyền và sau một khoảng thời gian (không phải ngay tức khắc), thay đổi sẽ đi đến mọi điểm trong hệ thống để cuối cùng dữ liệu trên hệ thống sẽ trở lại trạng thái nhất quán.

- Triển khai đơn giản, dễ nâng cấp và mở rộng.

- Mô hình dữ liệu và truy vấn linh hoạt. …

* 1. **Phân loại NoSQL**
     1. **Wide Column Store / Column Families**

Hệ cơ sở dữ liệu phân tán cho phép truy xuất ngẫu nhiên/tức thời với khả năng lưu trữ một lượng cực lớn data có cấu trúc. Dữ liệu có thể tồn tại dạng bảng với hàng tỷ bản ghi và mỗi bản ghi có thể chứa hàng triệu cột. Một triển khai từ vài trăm cho tới hàng nghìn commodity hardware dẫn đến khả năng lưu trữ hàng petabytes nhưng vẫn đảm bảo high performance. Dưới đây là một số sản phẩm thông dụng: Hadoop/HBase – Apache, BigTable – Google, Cassandra - Facebook/Apache, Hypertable - Zvents Inc/Baidu, Cloudera, SciDB, Mnesia, Tablets, …

* + 1. **Key-Value Store/Tuple store**

Mô hình lưu trữ dữ liệu dưới cặp giá trị key-value trong đó việc truy xuất, xóa, cập nhật giá trị thực thông qua key tương ứng. Với sự bổ trợ bởi các kỹ thuật BTree, B+Tree, Hash, ... dữ liệu có thể tồn tại trên RAM hoặc đĩa cứng, phân tán hoặc không phân tán. Hầu hết các NoSQL database đều là key-value store.

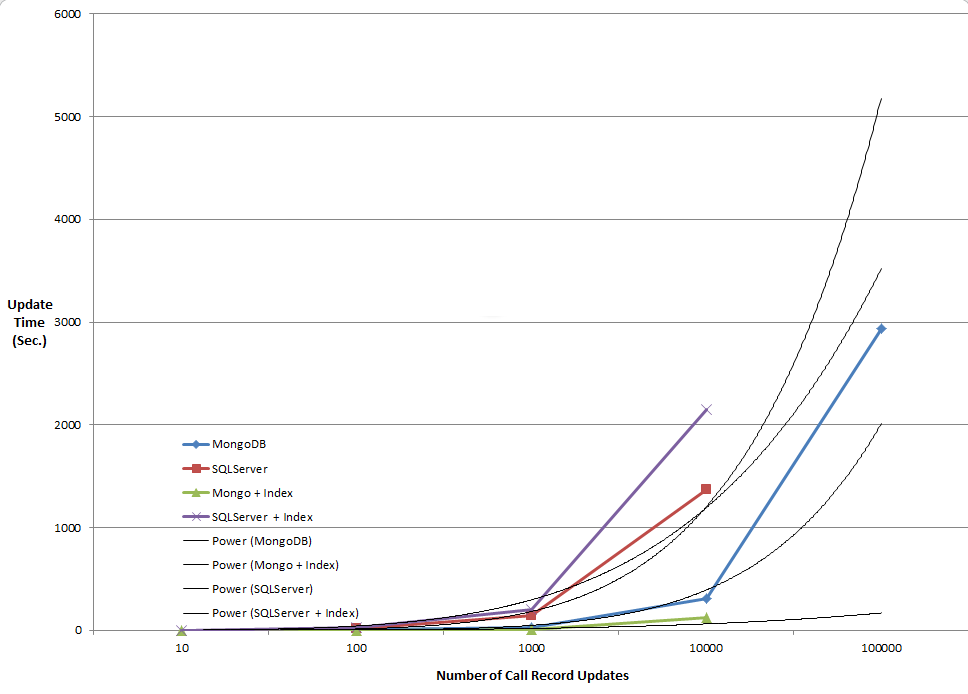
* + 1. **Document store**

Thực chất là các document-oriented database - một thiết kế riêng biệt cho việc lưu trữ document. Các cài đặt có thể là giả lập tương tác trên relational database, object database hay key-value store. Một số sản phẩm tiêu biểu: Apache Jackrabbit, CouchDB, IBM Lotus Notes Storage Format (NSF), MongoDB, Terrastore, ThruDB, OrientDB, RavenDB, ...

* + 1. **Graph Database**

Graph database là một dạng cơ sở dữ liệu được thiết kế riêng cho việc lưu trữ thông tin đồ họa như cạnh, nút, properties. Một số sản phẩm tiêu biểu: Neo4J, Sones, AllegroGraph, Core Data, DEX, FlockDB, InfoGrid, OpenLink Virtuoso, ...

* 1. **Hiệu năng NoSQL**



Hình phụ lục 4: Benchmark của MongoDB và MySQL

* + 1. **Xử lý suy giảm hiệu năng**

NoSQL không dùng các câu lệnh thông dịch SQL và các truy vấn rườm rà khác khiến nó cho ta một kiến trúc tối ưu về tốc độ thực thi (Đọc và ghi và cập nhật dữ liệu).Các câu lệnh SQL khá dễ đọc dễ hiểu.Nhưng nếu dữ liệu các đơn giản, các thủ tục SQL sẽ không cần thiết. Nhưng với các dữ liệu không có một cấu trúc cố định - ví dụ như - bài viết chẳng hạn, có rất nhiều kiểu bài viết, mỗi một kiểu bài viết lại có rất nhiều trường đi riêng biệt dành riêng cho nó.Tin tức thời sự chỉ có 3 trường là Tiêu đề, miêu tả, và phần thân, nhưng với văn bản pháp luật thì ngoài 3 trường trên còn có thêm trường cho phép đính kèm file văn bản pháp luật sửa đổi mới. Việc lưu trữ và truy vấn lấy các bài viết dạng thế này phải dùng khá nhiều câu lệnh join. Càng nhiều câu lệnh Join thì hiệu năng càng giảm.

Không như SQL, NoSQL lưu tất cả các trường tập trung vào trong một phần tử là document, các document này không nhất thiết phải bị "ràng buộc" phải có các trường giống nhau, nên sẽ có các phần tử có nhiều trường hơn các phần tử khác.Các khái niệm khóa ngoại khóa chính sẽ được loại bỏ - vì các document không còn quan hệ với nhau.Các bài viết được phân biệt với nhau bởi trường id.Đến lúc này khi truy vấn bài viết nào đó thì ta chỉ việc truy vấn thẳng đến document đó là có thể lấy chính xác tất cả dữ liệu.

1. **MongoDB**

Đây là một cơ sở dữ liệu NoSQL thuộc loại cơ sở dữ liệu hướng văn bản mã nguồn mở, có hiệu năng, tính sẵn có và tính mở rộng cao được cung cấp và hổ trợ bởi 10gen. MongoDB được 10gen phát triển từ năm 2007, đến năm 2009, MongoDB trở thành một sản phẩm mã nguồn mở có giấy phép AGPL. MongoDB được viết bằng ngôn ngữ C++. MongoDB lưu trữ dữ liệu dạng BSON. Không giống như các cơ sở dữ liệu quan hệ lưu dữ cấu trúc dữ liệu theo các bảng, MongoDB lưu trữ cấu trúc dữ liệu thành văn bản dựa JSON với mô hình động (gọi là BSON) khiến cho việc tích hợp dữ liệu cho các ứng dụng trở nên dễ dàng và nhanh hơn.Với mục tiêu là kết hợp các điểm mạnh của mô hình khóa – giá trị (nhanh mà tính mở rộng cao) với mô hình dữ liệu quan hệ (giàu chức năng).

Mục tiêu chính của Mongo là giữ lại các thuộc tính thân thiện của SQL. Do đó các câu truy vấn khá giống với SQL nên MongoDB khá thích hợp cho các lập trình viên đã quen với ngôn ngữ truy vấn SQL. MongoDB có một khối lượng tính năng lớn và hiệu năng cao. Với các loại dữ liệu phong phú, nhiều truy vấn và việc giảm thời gian phát triển trong việc mô hình hóa các đối tượng.

MongoDB được sử dụng tốt nhất với nhu cầu cần truy vấn động, cần tốc độ nhanh cho một cơ sở dữ liệu lớn vì MongoDB ngoài tốc độ đọc nhanh ra thì tốc độ ghi của nó rất nhanh.

MongoDB hỗ trợ việc tìm theo trường, khoảng kết quả tìm và tìm theo cú pháp. Các truy vấn có thể trả về các trường được qui định trong văn bản và cũng có thể bao gồm các hàm Javascript mà người dùng chưa định nghĩa. Cũng giống như các cơ sở dữ liệu quan hệ, bất cứ một trường nào trong MongoDB đều được đánh chỉ mục. MongoDB còn có hổ trợ theo mô hình chủ - tớ (master - slave), mảnh, vùng dữ liệu (Sharding).

MongoDB sử dụng một quá trình xử lý để xử lý các yêu cầu về dữ liệu, quản lý định dạng dữ liệu, thực hiện các hoạt động quản lý bên dưới là mongod, đây là trình xử lý chính. Trong việc mở rộng theo chiều ngang sử dụng mô hình mảnh lưu trữ, MongoDB cung cấp dịch vụ xử lý các truy vấn từ tầng ứng dụng, xác định vị trí dữ liệu trong cụm các node phân mảnh được gọi là mongos

* 1. **Mô hình dữ liệu tổng quát**

MongoDB thuộc cơ sở dữ liệu hướng văn bản. Phần kiến trúc cơ bản của cơ sở dữ liệu hướng văn bản, mỗi cơ sở dữ liệu trong MongoDB có thể bao gồm một tập các bộ sưu tập, mỗi bộ sưu tập có thể bao gồm một tập các văn bản, mỗi văn bản có một khóa đặc biệt, đó là “\_id”, nó là duy nhất trong bộ sưu tập của văn bản và nó là một chuỗi, các văn bản bao gồm một tập các trường. Mỗi trường là một cặp khóa – giá trị. Các cặp khóa – giá trị trong văn bản được sắp xếp, hai văn bản có cùng các cặp khóa – giá trị nhưng vị trí sắp xếp khác nhau thì hai văn bản đó khác nhau. Các khóa trong một văn bản không được chứa giá trị giống nhau. MongoDB phân biệt chữ cái hoa thường.

Dữ liệu trong mongoDB là một giản đồ linh hoạt. Giản đồ linh hoạt có nghĩa là ta có thể mô hình hóa các văn bản để thể hiện hiện tương đồng với các thực thể hay các đối tượng. Các bộ sư tập không tuân theo một cấu của các văn bản, nghĩa là các văn bản trong một bộ sưu tập không cần phải có cùng cấu trúc của các trường, các trường giống nhau có thể chứa các loại dữ liệu khác nhau.

Mỗi văn bản chỉ cần có các trường liên quan tới các thực thể hoặc đối tượng mà văn bản thể hiện. Thực tế thì hầu hết các văn bản ở trong MongoDB có cấu trúc tương tự nhau.

Khi ta xây dựng, phát triển một mô hình dữ liệu trong MongoDB thì phải xem xét đến các thuộc tính vốn có và mối quan hệ giữa các đối tượng trong ứng dụng, cách mà dữ liệu phát triển và thay đổi theo thời gian, các loại truy vấn mà ứng dụng thực hiện.

* 1. **Mô hình nhúng**

Lưu hai phần của dữ liệu vào trong một văn bản nên hạn chế số lượng các văn bản, làm cho hiệu suất của máy chủ cao hơn.

Sử dụng mô hình nhúng khi: Ta có các mối quan hệ một – một giữa các thực thể, có các mối quan hệ một - nhiều khi mà nhiều đối tượng luôn luôn xuất hiện có hoặc được xem trong bối cảnh văn bản cha.

Lợi ích của mô hình nhúng là tăng hiệu suất hơn cho các thao tác đọc dữ liệu và khả năng yêu cầu, truy vấn dữ liệu trong một cơ sở dữ liệu.

Việc nhúng trong các văn bản tăng dung lượng của văn bản dẫn đến giảm hiệu suất ghi và dữ liệu bị phân mảnh. Ngoài ra, văn bản trong MongoDB có dung lượng nhỏ hơn dung lượng tối đa của một văn bản BSON là 16 Megabytes. Nếu ta muốn lưu một văn bản lớn hơn 16 Megabytes thì ta sử dụng GridFS API.

* 1. **Mô hình tham chiếu**

Lưu trữ tham chiếu giữa hai văn bản để chỉ ra mối quan hệ của dữ liệu được trình bày trong văn bản.

Sử dụng mô hình tham chiếu khi dùng mô hình nhúng cho dữ liệu trùng lặp, nhưng hiệu năng của việc đọc dữ liệu không thể bằng những tác động xấu của sự trùng lặp dữ liệu, khi cần biểu diển mối quan hệ nhiều nhiều giữa các đối tượng và khi cần xây dựng một mô hình lớn của các bộ dữ liệu phân cấp.

Mô hình tham chiếu linh động hơn mô hình nhúng, tuy nhiên, để giải quyết các tham chiếu thì ứng dụng phía máy khách phải theo dõi vấn đề truy vấn hay phải truy vấn nhiều vòng đến máy chủ hơn.