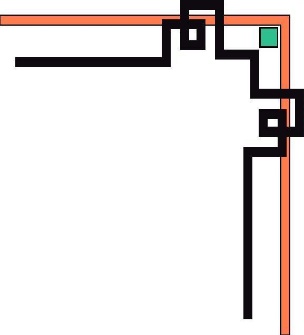
**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**



**PHÂN HIỆU TẠI TP. HỒ CHÍ MINH**

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**

-----🙞🙜🕮🙞🙜-----



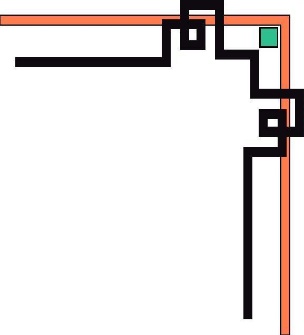
**BÁO CÁO THỰC TẬP TỐT NGHIỆP**

**THỰC TẬP SINH BACK-END DEVELOPER - TRIỂN KHAI VÀ TÌM KIẾM GIẢI PHÁP TỐI ƯU**

|  |  |
| --- | --- |
| Giảng viên hướng dẫn: | **ThS. TRẦN PHONG NHÃ** |
| Sinh viên thực hiện: | **HỒ VĨNH TÍN** |
| Lớp: | **CQ.62.CNTT** |
| Khóa: | **62** |

Tp. Hồ Chí Minh, tháng 3 năm 2025

**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**



**PHÂN HIỆU TẠI TP. HỒ CHÍ MINH**

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**

-----🙞🙜🕮🙞🙜-----



**BÁO CÁO THỰC TẬP TỐT NGHIỆP**

**THỰC TẬP SINH BACK-END DEVELOPER - TRIỂN KHAI VÀ TÌM KIẾM GIẢI PHÁP TỐI ƯU**

|  |  |
| --- | --- |
| Giảng viên hướng dẫn: | **ThS. TRẦN PHONG NHÃ** |
| Sinh viên thực hiện: | **HỒ VĨNH TÍN** |
| Lớp: | **CQ.62.CNTT** |
| Khóa: | **62** |

Tp. Hồ Chí Minh, tháng 3 năm 2025

|  |  |
| --- | --- |
| **TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI** | **CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM** |
| **PHÂN HIỆU TẠI THÀNH PHỐ HỒ CHÍ MINH** | **Độc lập – Tự do – Hạnh phúc** |

**NHIỆM VỤ THỰC TẬP TỐT NGHIỆP**BỘ MÔN: **CÔNG NGHỆ THÔNG TIN**-------\*\*\*-------

|  |  |
| --- | --- |
| **MSSV: 6251071104** | **Họ và tên SV: Hồ Vĩnh Tín** |
| **Khóa: 62** | **Lớp: CQ.62.CNTT** |

1. **Tên đề tài thực tập tốt nghiệp**

Thực tập sinh Back-end developer – Triển khai và tìm kiếm giải pháp tối ưu.

1. **Nhiệm vụ thực tập tốt nghiệp**

* Tìm hiểu, triển khai các giải pháp tối ưu cho hệ thống.
* Xây dựng các chức năng theo yêu cầu.
* Giám sát, triển khai ứng dụng trên máy chủ.

Ngày bắt đầu thực tập tốt nghiệp: Ngày 1 tháng 8 năm 2024.

Ngày hoàn thành báo cáo thực tập tốt nghiệp: Ngày 20 tháng 2 năm 2025.

Tên công ty: CÔNG TY CỔ PHẦN COLIV HOME.

Địa chỉ: 76 đường số 1 KDC Cityland P10 Gò Vấp, Thành phố Hồ Chí Minh.

1. **Giảng viên hướng dẫn**

Họ tên: ThS. Trần Phong Nhã.

Đơn vị công tác: Bộ môn Công nghệ thông tin Phân hiệu Trường Đại học Giao thông Vận tải Phân hiệu tại TP. Hồ Chí Minh.

Điện thoại: 0906761014 Email: tpnha@utc2.edu.vn

|  |  |
| --- | --- |
| **Ngày…tháng 03 năm 2025**  **Trưởng BM Công nghệ Thông tin** | **Đã giao nhiệm vụ TKTN**  **Giáo viên hướng dẫn** |
| **ThS. Trần Phong Nhã** | **ThS. Trần Phong Nhã** |

Đã nhận nhiệm vụ TKTN

|  |  |
| --- | --- |
| Sinh viên: Hồ Vĩnh Tín | Ký tên: Tín |
| Điện thoại: 0833402853 | Email: 6251071104@st.utc2.edu.vn |

# LỜI CẢM ƠN

Em xin gửi lời cảm ơn chân thành và sâu sắc đến Công ty cổ phần COLIV Home, nơi đã tạo điều kiện thuận lợi cho em thực tập và học hỏi trong suốt thời gian qua. Thực tập tại công ty là một cơ hội quý báu giúp em tiếp xúc với môi trường làm việc thực tế, áp dụng những kiến thức đã học vào thực tiễn, đồng thời nâng cao các kỹ năng lập trình và làm việc nhóm.

Đặc biệt, em xin gửi lời tri ân sâu sắc đến thầy Trần Phong Nhã, người đã luôn tận tình chỉ bảo, hướng dẫn em từ những ngày đầu bỡ ngỡ cho đến khi em hoàn thành kỳ thực tập. Anh/chị không chỉ giúp em hiểu rõ hơn về các công nghệ, công cụ phát triển phần mềm mà còn truyền đạt những kinh nghiệm thực tế quý báu, giúp em rèn luyện tư duy logic, khả năng giải quyết vấn đề và phong cách làm việc chuyên nghiệp.

Bên cạnh đó, em cũng xin gửi lời cảm ơn chân thành đến các anh/chị, những người đã luôn sẵn sàng hỗ trợ, chia sẻ kiến thức và kinh nghiệm làm việc thực tế. Nhờ sự giúp đỡ tận tình của các anh/chị, em đã học hỏi thêm nhiều kỹ năng quan trọng, từ lập trình, kiểm thử phần mềm, đến việc làm việc nhóm và xử lý tình huống trong quá trình phát triển sản phẩm.

Không chỉ vậy, em cũng xin gửi lời cảm ơn đến quý thầy cô, những người đã truyền đạt cho em nền tảng kiến thức vững chắc, tạo cơ hội để em được thực tập và áp dụng những gì đã học vào thực tế. Những bài học quý giá từ giảng đường đã giúp em tự tin hơn trong quá trình thực tập và làm việc tại công ty.

Cuối cùng, em xin bày tỏ lòng biết ơn sâu sắc đến gia đình và bạn bè, những người đã luôn động viên, khuyến khích và ủng hộ em trong suốt quá trình học tập và thực tập. Sự động viên và hỗ trợ từ mọi người là nguồn động lực to lớn giúp em hoàn thành kỳ thực tập một cách tốt nhất.

Một lần nữa, em xin chân thành cảm ơn tất cả mọi người đã giúp đỡ em trong suốt quá trình thực tập. Em hy vọng rằng những kinh nghiệm và bài học quý báu có được từ đợt thực tập này sẽ là hành trang quan trọng giúp em phát triển sự nghiệp trong tương lai.

# NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

*TP. Hồ Chí Minh, ngày ...... tháng 3 năm 2025*

**Giảng viên hướng dẫn**

*(Ký và ghi họ và tên)*

**MỤC LỤC**

[LỜI CẢM ƠN iii](#_Toc193961940)

[NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN iv](#_Toc193961941)

[DANH MỤC HÌNH ẢNH VÀ BẢNG BIỂU vii](#_Toc193961942)

[CHƯƠNG 1: TỔNG QUAN VỀ CÔNG TY 1](#_Toc193961943)

[1.1. Giới thiệu chung 1](#_Toc193961944)

[1.2. Cấu trúc nội dung báo cáo Thực tập tốt nghiệp 1](#_Toc193961945)

[CHƯƠNG 2: QUÁ TRÌNH THỰC TẬP 3](#_Toc193961946)

[2.1. Cơ cấu công ty 3](#_Toc193961947)

[2.2. Thời gian thực tập 3](#_Toc193961948)

[2.3. Nhiệm vụ 4](#_Toc193961949)

[CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG 7](#_Toc193961950)

[3.1. Yêu cầu mong muốn của công ty 7](#_Toc193961951)

[3.2. Phân tích yêu cầu 7](#_Toc193961952)

[3.3. Công nghệ sử dụng 7](#_Toc193961953)

[3.3.1. Các công cụ sử dụng 7](#_Toc193961954)

[3.3.2. ASP Net Core API 8](#_Toc193961955)

[3.3.3. SQL server 10](#_Toc193961956)

[3.3.4. NestJS 11](#_Toc193961957)

[3.3.5. MongoDB 15](#_Toc193961958)

[3.3.6. Nginx 23](#_Toc193961959)

[3.3.7. Redis 28](#_Toc193961960)

[3.3.8. MinIO 31](#_Toc193961961)

[3.3.9. Docker 35](#_Toc193961962)

[3.3.10. Angular 39](#_Toc193961963)

[3.3.11. Linux 44](#_Toc193961964)

[3.4. Các chức năng chính 45](#_Toc193961965)

[3.5. Các nội dung học tập, kết quả đạt được tại công ty 47](#_Toc193961966)

[3.5.1. Môi trường thực tế 47](#_Toc193961967)

[3.5.2. Về nguyên tắc làm việc 47](#_Toc193961968)

[CHƯƠNG 4: XÂY DỰNG ỨNG DỤNG 49](#_Toc193961969)

[4.1. Giao diện ứng dụng quản lý nhà 49](#_Toc193961970)

[4.2. Giao diện quản lý hợp đồng 50](#_Toc193961971)

[4.3. Giao diện quản lý tài chính 51](#_Toc193961972)

[4.4. Giao diện ứng dụng môi giới 52](#_Toc193961973)

[CHƯƠNG 5: KẾT LUẬN 54](#_Toc193961974)

[5.1. Kết luận 54](#_Toc193961975)

[5.2. Đóng góp, nhận xét từ phía công ty 54](#_Toc193961976)

[5.2.1. Đóng góp 54](#_Toc193961977)

[5.2.2. Nhận xét từ phía công ty 54](#_Toc193961978)

[5.3. Hướng phát triển của bản thân 55](#_Toc193961979)

[TÀI LIỆU THAM KHẨO 56](#_Toc193961980)

# DANH MỤC HÌNH ẢNH VÀ BẢNG BIỂU

[Hình 3.1 ASP Net Core API 8](#_Toc193962553)

[Hình 3.2 SQL server 10](#_Toc193962554)

[Hình 3.3 NestJS 11](#_Toc193962555)

[Hình 3.4 Kiến trúc NestJS 13](#_Toc193962556)

[Hình 3*.*5 MongoDB 15](#_Toc193962557)

[Hình 3.6 Đặc điểm của MongoDB 16](#_Toc193962558)

[Hình 3.7 Nguyên lý hoạt động 17](#_Toc193962559)

[Hình 3*.*8 Kiến trúc MongoDB 18](#_Toc193962560)

[Hình 3.9 Real-time payment với MongoDB 21](#_Toc193962561)

[Hình 3.10 IoT của MongoDB 21](#_Toc193962562)

[Hình 3.11 Ưu điểm MongoDB 23](#_Toc193962563)

[Hình 3.12 Nginx 24](#_Toc193962564)

[Hình 3.13 Kiến trúc của Nginx 25](#_Toc193962565)

[Hình 3.14 Nguyên lý hoạt động 26](#_Toc193962566)

[Hình 3.15 Redis 28](#_Toc193962567)

[Hình 3.16 Ứng dụng của Redis 30](#_Toc193962568)

[Hình 3.17 MinIO 32](#_Toc193962569)

[Hình 3.18 Ứng dụng của MinIO 34](#_Toc193962570)

[Hình 3.19 Docker 35](#_Toc193962571)

[Hình 3.20 Kiến trúc Docker 36](#_Toc193962572)

[Hình 3.21 Angular 39](#_Toc193962573)

[Hình 3.22 Kiến trúc, thành phần của Angular 40](#_Toc193962574)

[Hình 3.23 Linux 44](#_Toc193962575)

[Hình 3.24 Các chứng năng chính của ứng dụng 46](#_Toc193962576)

[Hình 4.1 Dashboard 49](#_Toc193962577)

[Hình 4.2 Danh sách nhà 49](#_Toc193962578)

[Hình 4.3 Danh sách phòng của một tòa nhà 50](#_Toc193962579)

[Hình 4.4 Danh sách hợp đồng (cọc và thuê) 50](#_Toc193962580)

[Hình 4.5 Cập nhật cọc 51](#_Toc193962581)

[Hình 4.6 Thống kê dòng tiền 51](#_Toc193962582)

[Hình 4.7 Thu tiền thuê tháng đầu 52](#_Toc193962583)

[Hình 4.8 Chốt cọc 52](#_Toc193962584)

[Hình 4.9 Danh sách căn hộ 53](#_Toc193962585)

[Bảng 2.1 Mốc thời gian thực tập 6](#_Toc193538916)

# CHƯƠNG 1: TỔNG QUAN VỀ CÔNG TY

* 1. **Giới thiệu chung**
* CÔNG TY CỔ PHẦN COLIV HOME, tên quốc tế là COLIV HOME JOINT STOCK COMPANY, được cấp giấy chứng nhận đăng ký kinh doanh vào ngày 17 tháng 1 năm 2025. Trụ sở chính của công ty đặt tại số 76 Đường số 1, Khu dân cư Cityland Park Hills, Phường 10, Quận Gò Vấp, Thành phố Hồ Chí Minh. Người đại diện pháp luật là ông Dương Đức Phong.
* Ngành nghề kinh doanh chính của COLIV HOME bao gồm tư vấn, môi giới, đấu giá bất động sản và đấu giá quyền sử dụng đất. Cụ thể, công ty cung cấp dịch vụ tư vấn bất động sản, môi giới bất động sản và quản lý bất động sản trên cơ sở phí hoặc hợp đồng (trừ tư vấn mang tính pháp lý).
* Ngoài ra, công ty còn hoạt động trong các lĩnh vực khác như in ấn, xây dựng nhà ở và nhà không để ở, phá dỡ, chuẩn bị mặt bằng, lắp đặt hệ thống điện, cấp thoát nước, điều hòa không khí, hoàn thiện công trình xây dựng, bán buôn nông sản, thực phẩm, đồ uống, dịch vụ lưu trú ngắn ngày, nhà hàng và các dịch vụ ăn uống, quảng cáo, nghiên cứu thị trường, giáo dục và nhiều lĩnh vực khác.
* Với mục tiêu trở thành một trong những công ty hàng đầu trong lĩnh vực bất động sản và dịch vụ đi kèm, COLIV HOME đặt trọng tâm vào:
* Phát triển mô hình co-living (sống chung) hiện đại: Hướng đến đối tượng khách hàng trẻ, nhân viên văn phòng, người nước ngoài có nhu cầu thuê căn hộ theo mô hình chia sẻ tiện ích.
* Mở rộng mạng lưới bất động sản: Hợp tác với nhiều chủ đầu tư và cá nhân để cung cấp đa dạng sản phẩm nhà ở, văn phòng, khách sạn.
* Đẩy mạnh ứng dụng công nghệ: Tích hợp nền tảng số vào quản lý bất động sản, giúp khách hàng dễ dàng tìm kiếm và giao dịch.
* Nâng cao chất lượng dịch vụ: Cung cấp các giải pháp tư vấn, môi giới chuyên nghiệp, hỗ trợ khách hàng trong quá trình giao dịch bất động sản.
  1. **Cấu trúc nội dung báo cáo Thực tập tốt nghiệp**

Chương 1: Tổng quan về công ty.

Chương 2: Quá trình thực tập.

Chương 3: Phân tích thiết kế hệ thống.

Chương 4: Xây dựng ứng dụng.

Chương 5: Kết luận.

# CHƯƠNG 2: QUÁ TRÌNH THỰC TẬP

* 1. **Cơ cấu công ty**

Trong quá trình thực tập, em được công ty giới thiệu về lĩnh vực hoạt động, các dịch vụ chuyển đổi số mà Coliv Home cung cấp. Các module chính như:

* Phần mềm quản lý nhà cho các chủ nhà: Thông thường mỗi chủ nhà có rất nhiều nhà, mỗi tòa nhà có khá nhiều phòng và phải quản lý thông qua sổ sách ghi chép khá bất tiện. Phần mềm quản lý này giúp chủ nhà số hóa thông tin trên phần mềm, quản lý dễ dàng thông qua giao diện đơn giản, dễ sử dụng. Phần mềm này còn giúp chủ nhà quản lý hồ sơ của nhân viên, quản lý tòa nhà (mỗi tòa nhà có nhiều quản lý và nhân viên), các nhân viên này sẽ được phụ trách cho một hoặc nhiều tòa nhà nào đó, công việc này cũng được triển khai trên phần mềm.
* Phần quan trọng hơn là quản lý hợp đồng cọc và thuê để theo dõi, quản lý các hợp đồng, xem thông tin, giá thuê của hợp đồng.
* Phần mềm hỗ trợ tìm kiếm nhà cho môi giới: Các chủ nhà họ có rất nhiều tòa nhà, một trong số đó chủ nhà có thể thêm vào các chính sách dành cho môi giới, ví dụ như hoa hồng khi môi giới giúp chủ nhà tìm kiếm được khách hàng thuê nhà cho họ. Phần mềm này hỗ trợ tìm kiếm môi giới tìm kiếm các căn hộ, tìm kiếm khách hàng phù hợp cho căn hộ đó.
* Trang quản trị dành cho Admin: Dùng để quản lý, cung cấp tài khoản chủ nhà, cập nhật một số thông tin mà chủ nhà không có quyền chỉnh sửa, quản tất cả hình ảnh của toàn bộ ứng dụng.
* Phần mềm khách hàng: Phần mềm này dành cho khách thuê căn hộ. Giúp khách thuê xem được thông tin căn hộ của mình, chi phí dịch vụ, nội thất, xem được chi tiết hợp đồng thuê đã ký kết với chủ nhà, quản lý danh sách thành viên trong cùng căn hộ.
* Quản lý tài chính: tự động hóa việc thanh toán hợp đồng cọc, thanh toán tiền nhà cùng với các phí dịch vụ khác, xem thống kê dòng tiền ra vào, lập phiếu thu chi xem lợi nhuận.
  1. **Thời gian thực tập**

Khoản thời gian thực tập tháng 7 đến tháng 9, em thực tập 3 buổi trong tuần vào thứ 2 4 6, vào lúc 8h30 đến 5h30.

Khoản thời gian thực từ tháng 10 đến tháng 12, em thực tập vào thứ 2, thứ 3, thứ 5 trong tuần với vào lúc 8h30 đến 5h30.

* 1. **Nhiệm vụ**

|  |  |  |
| --- | --- | --- |
| Tuần | Công việc được giao | Mức độ hoàn thành |
| 1🡪2 | Tìm hiểu nghiệp vụ của ứng dụng | Hoàn thành |
| 3🡪4 | Tìm hiểu kiến trúc dự án, công nghệ sử dụng | Hoàn thành |
| 5 | Tìm kiếm giải pháp lưu trữ hình ảnh | Hoàn thành |
| 6 | Chức năng quản lý tòa nhà:   * Thêm phòng * Thêm toàn nhà * Chỉnh sửa tòa nhà * Thêm ảnh nhà * Xóa ảnh ảnh nhà * Xuất nhập file excel danh sách phòng | Hoàn thành |
| 7 | Chức năng quản lý phòng:   * Thêm phòng * Cập nhật phòng * Xóa phòng | Hoàn thành |
| 8 | Chức năng quản lý phòng:   * Thêm ảnh phòng * Xóa ảnh phòng | Hoàn thành |
| 9 | Sử dụng Docker để triển khai Back-end và Front-end lên VPS máy chủ Linux | Hoàn thành |
| 10 | Kết hợp NGINX deploy ứng dụng với tên miền | Hoàn thành |
| 11 | Bổ sung chứng chỉ TLS vào tên miền | Hoàn thành |
| 12 | Các chức năng quản lý hợp đồng cọc:   * Thêm hợp đồng cọc * Cập nhật * Hủy * Xuất file PDF | Hoàn thành |
|  | Chức năng quản lý hợp đồng thuê:   * Thêm hợp đồng cọc * Cập nhật * Hủy hợp đồng thuê * Thêm khách hàng * Xuất file PDF * Chuyển phòng |  |
| 13🡪16 | Nghiên cứu NestJS và MongoDB, phân tích lại database | Hoàn thành |
| 17 | Chuyển đổi hệ thống sang sử dụng NestJS và MongoDB | Hoàn thành |
| 18 |
| 19 | Tích hợp Redis  Sử dụng dịch vụ quét thẻ căn cước của FPT | Hoàn thành |
| 20 | Tài khoản môi giới:   * Đăng nhập * Đăng ký * Quên mật khẩu | Hoàn thành |
| 21 | Chức năng web môi giới:   * Hỗ trợ tìm phòng theo bộ lọc * Xem chi tiết căn hộ | Hoàn thành |
| 22 | Chức năng web môi giới:   * Tạo lịch hẹn xem phòng * Xem danh sách lịch hẹn | Hoàn thành |
| 23 | Chức năng web môi giới:   * Tạo hợp đồng cọc * Xem danh sách hợp đồng cọc * Xem tổng quan tòa nhà | Hoàn thành |
| 24 | Admin:   * Thêm tài khoản chủ nhà * Khóa tài khoản chủ nhà * Quản lý thông tin ngân hàng cho mỗi tòa nhà | Hoàn thành |

Bảng 2.1 Mốc thời gian thực tập

# CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

* 1. **Yêu cầu mong muốn của công ty**
* Yêu cầu về kỹ thuật:
* Thành thạo hoặc có kiến thức cơ bản về C#, TypeScript.
* Có khả năng làm việc với ASP Net Core API, NestJS.
* Sử dụng tốt Git để quản lý mã nguồn và làm việc nhóm.
* Hiểu về cơ sở dữ liệu MongoDB, SQL Server và có thể truy vấn dữ liệu.
* Tìm hiểu, nghiên cứu nhanh các dịch vụ, công nghệ khác.
* Yêu cầu về tư duy và kỹ năng làm việc:
* Tư duy logic và giải quyết vấn đề: Có khả năng phân tích và tìm ra giải pháp tối ưu khi gặp lỗi.
* Tính chủ động: Chủ động tìm hiểu tài liệu, đề xuất giải pháp và cải thiện công việc.
* Làm việc nhóm tốt: Biết cách giao tiếp, phối hợp và hỗ trợ đồng đội trong dự án.
* Tuân thủ deadline: Hoàn thành nhiệm vụ đúng thời gian và báo cáo tiến độ thường xuyên.
  1. **Phân tích yêu cầu**
* Phân tích tính khả thi:
* Kiểm tra tính khả thi của từng chức năng dựa trên công nghệ sẵn có.
* Xác định những rủi ro như tải trọng hệ thống, lỗi bảo mật, khả năng mở rộng.
* Đề xuất các phương án để tối ưu hiệu suất và giảm thiểu lỗi.
* Phân chia công việc và lập kế hoạch phát triển:
* Công việc được chia thành nhiều giai đoạn.
* Mỗi thành viên trong nhóm phụ trách một phần nhỏ như frontend, backend, database, kiểm thử.
* Đặt ra các mốc thời gian để hoàn thành từng nhiệm vụ và kiểm tra kết quả.
  1. **Công nghệ sử dụng**
     1. **Các công cụ sử dụng**
* Visual Studio.
* SQL server.
* SQL Server Management Studio.
* Visual studio code.
* MongoDB Compass.
* Redis Insight.
* Hoppscotch.
* MobaXterm.
* Lark.
  + 1. **ASP Net Core API**



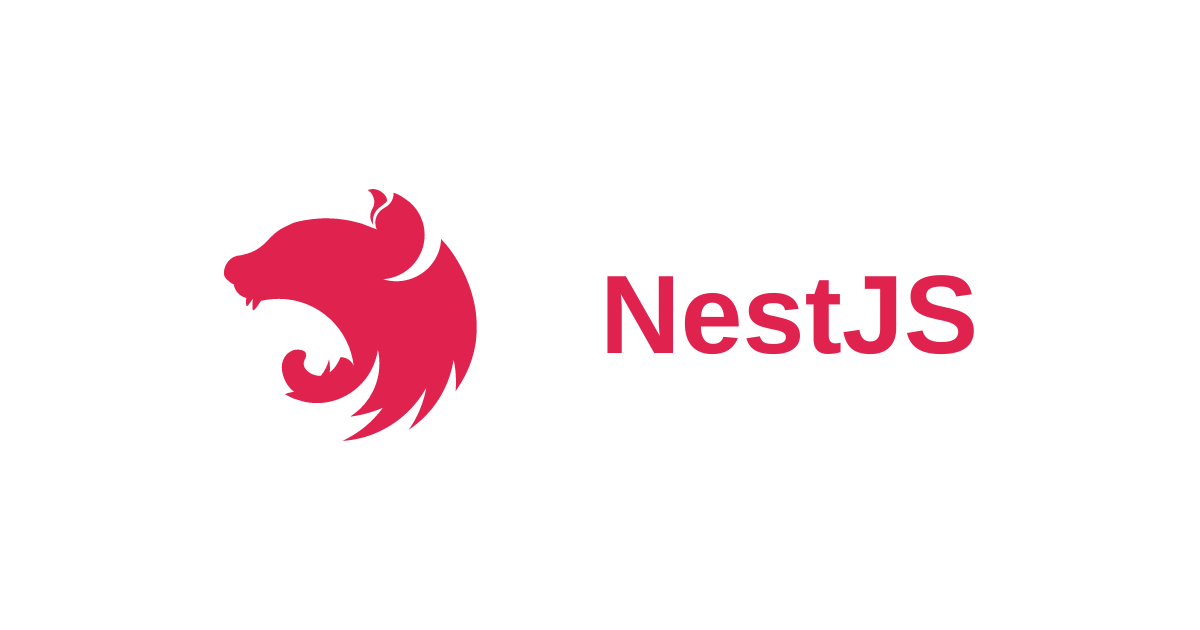
Hình 3.1 ASP Net Core API

* ASP.NET Core API là một khuôn khổ web hiệu suất cao do Microsoft phát triển để xây dựng các ứng dụng kết nối internet dựa trên đám mây. Nó là một phần của hệ sinh thái ASP.NET Core, cho phép các nhà phát triển tạo các ứng dụng và dịch vụ web có thể chạy trên nhiều nền tảng. Kiến trúc mô-đun và khả năng đa nền tảng của nó khiến nó trở thành lựa chọn phổ biến để tạo các dịch vụ và API web có khả năng mở rộng và bảo trì. Khuôn khổ này nhấn mạnh vào hiệu suất và năng suất của nhà phát triển, sử dụng các thành phần phần mềm trung gian để xử lý hiệu quả các yêu cầu và phản hồi. Nó cũng bao gồm các công cụ bảo mật như OAuth và xác thực đa yếu tố để bảo vệ dữ liệu. Bất chấp những thách thức của nó, ASP.NET Core API nổi bật về tính linh hoạt và hiệu suất, khiến nó trở thành khuôn khổ hàng đầu cho các nhà phát triển xây dựng các dịch vụ web chất lượng cao trong bối cảnh kỹ thuật số.
* Tính năng:
* Kiến trúc mô-đun: Thiết kế mô-đun của ASP.NET Core cho phép các nhà phát triển chỉ thêm các thư viện và chức năng cần thiết, tạo ra các ứng dụng hiệu quả hơn. Khung này sử dụng các gói NuGet, cho phép thêm và xóa chức năng khi cần, giảm sự phình to và tăng tốc độ ứng dụng.
* Khả năng tương thích đa nền tảng: Một trong những tính năng nổi bật của ASP.NET Core là khả năng đa nền tảng. Không giống như phiên bản trước, chỉ giới hạn ở Windows, ASP.NET Core cho phép các nhà phát triển xây dựng và triển khai ứng dụng trên nhiều hệ điều hành, bao gồm Windows, Linux và macOS. Tính linh hoạt này cho phép nhiều môi trường phát triển và tùy chọn lưu trữ hơn, khiến nó trở thành lựa chọn hấp dẫn cho các nhà phát triển tìm kiếm sự linh hoạt trong các dự án của họ.
* Middleware Pipelining: Kiến trúc của khung dựa trên các thành phần phần mềm trung gian hoạt động cùng nhau để tạo thành một đường ống yêu cầu-phản hồi. Thiết kế này hiệu quả và cho phép tích hợp logic tùy chỉnh ở nhiều giai đoạn khác nhau của quá trình xử lý yêu cầu, tăng cường tính linh hoạt của các ứng dụng web.
* Các tính năng bảo mật nâng cao: ASP.NET Core nhấn mạnh vào bảo mật, với các cải tiến tích hợp hỗ trợ OAuth3, xác thực đa yếu tố và nguyên tắc Zero Trust. Các tính năng bảo mật này rất quan trọng để bảo vệ các ứng dụng, đặc biệt là các ứng dụng liên quan đến dữ liệu nhạy cảm hoặc yêu cầu xác thực người dùng an toàn.
* Kiến trúc:
* Domain Layer: Lớp trong cùng này chứa logic kinh doanh cốt lõi và các thực thể, hoàn toàn độc lập với các công nghệ bên ngoài. Nó bao gồm các lớp mô hình kinh doanh, tổng hợp, thông số kỹ thuật và dịch vụ miền, đảm bảo rằng ứng dụng có thể phát triển mà không bị ràng buộc với các công nghệ cụ thể.
* Application Layer: Lớp này quản lý các trường hợp sử dụng và hoạt động như một trung gian giữa Lớp miền và các đầu vào bên ngoài. Nó phối hợp các hoạt động và sắp xếp các tương tác, sử dụng logic kinh doanh được xác định trong Lớp miền.
* Infrastructure Layer: Chịu trách nhiệm xử lý các mối quan tâm bên ngoài, chẳng hạn như lưu trữ dữ liệu và API bên ngoài, lớp này triển khai các mẫu kho lưu trữ để tách logic truy cập dữ liệu khỏi ứng dụng cốt lõi. Nó chứa các triển khai cho các giao diện được xác định trong Lõi ứng dụng.
* Presentation Layer: Lớp ngoài cùng này tương tác với người dùng hoặc các hệ thống khác thông qua giao diện người dùng hoặc API. Nó quản lý các yêu cầu và phản hồi HTTP, đảm bảo trải nghiệm liền mạch trong khi vẫn giữ cho logic kinh doanh cốt lõi được tách biệt khỏi các thay đổi giao diện người dùng.
* Ưu điểm và thách thức: Việc triển khai Clean Architecture có thể mang lại những lợi thế đáng kể, bao gồm khả năng bảo trì mã được cải thiện, thử nghiệm dễ dàng hơn và khả năng thích ứng với sự thay đổi được nâng cao. Tuy nhiên, nó cũng có thể gây ra sự phức tạp, đặc biệt là trong các dự án nhỏ hơn, nơi mà chi phí chung của nhiều lớp có thể không được biện minh. Việc cân nhắc cẩn thận về quy mô, độ phức tạp và chuyên môn của nhóm dự án là điều cần thiết khi quyết định tính phù hợp của Clean Architecture.
  + 1. **SQL server**



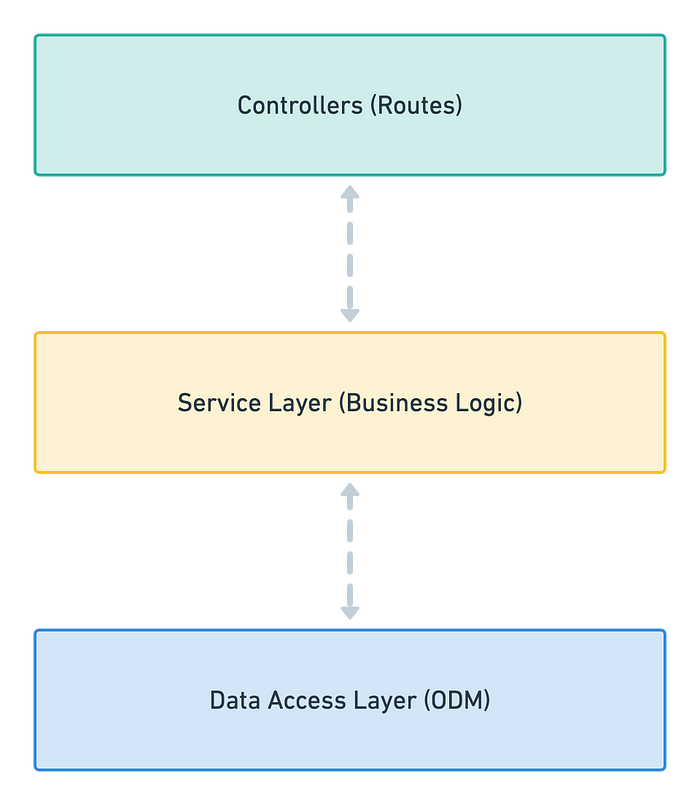
Hình 3.2 SQL server

* SQL Server là một hệ thống quản lý cơ sở dữ liệu quan hệ (RDBMS) do Microsoft phát triển, nổi tiếng với kiến ​​trúc mạnh mẽ và khả năng quản lý và lưu trữ hiệu quả các tập dữ liệu lớn. SQL Server đã phát triển thành một nền tảng đa năng, hỗ trợ nhiều ứng dụng khác nhau, từ các giải pháp quy mô nhỏ đến các hệ thống doanh nghiệp lớn.
* Được thiết kế để lưu trữ và quản lý hiệu quả lượng dữ liệu lớn. Hệ thống này hoạt động theo các nguyên tắc của thuộc tính giao dịch, thường được gọi là ACID, viết tắt của Atomicity, Consistency, Isolation và Durability.
* Hệ thống cho phép các truy vấn phức tạp, hỗ trợ Ngôn ngữ định nghĩa dữ liệu (DDL), Ngôn ngữ thao tác dữ liệu (DML), Ngôn ngữ kiểm soát dữ liệu (DCL) và Ngôn ngữ kiểm soát giao dịch (TCL).
* Kiến trúc:
* Các thành phần cốt lõi: Công cụ cơ sở dữ liệu của kiến ​​trúc SQL Server quản lý việc lưu trữ, truy xuất và thao tác dữ liệu trong DBMS, trong đó kho dữ liệu là thành phần quan trọng để lưu trữ dữ liệu có cấu trúc lớn.
* Dịch vụ phân tích: SQL Server Analysis Services (SSAS) là một thành phần quan trọng của ngăn xếp Business Intelligence (BI), cho phép phân tích nâng cao, xử lý các tập dữ liệu lớn và hỗ trợ các khả năng truy vấn phức tạp.
* Dịch vụ tích hợp: SSIS là một thành phần quan trọng của kiến ​​trúc SQL Server, cho phép tích hợp dữ liệu liền mạch từ nhiều nguồn khác nhau, đảm bảo làm sạch và chuyển đổi dữ liệu trước khi đưa vào cơ sở dữ liệu phân tích.
* Mô hình hóa chiều: Mô hình hóa theo chiều là một tính năng chính của kiến ​​trúc SQL Server, đặc biệt là trong Power BI, tối ưu hóa hiệu suất và xử lý dữ liệu bằng cách kết hợp các bảng dữ liệu hẹp với các chiều liên quan.
  + 1. **NestJS**



Hình 3.3 NestJS

* NestJS là một framework Node.js mã nguồn mở, được thiết kế để phát triển các ứng dụng phía server hiệu quả và dễ bảo trì. Sử dụng ngôn ngữ TypeScript, NestJS kết hợp các nguyên tắc lập trình hướng đối tượng (OOP), lập trình hàm (FP) và lập trình phản ứng hàm (FRP) để tạo ra một môi trường phát triển mạnh mẽ.
* Lấy cảm hứng từ kiến trúc của Angular, NestJS áp dụng mô hình module hóa, giúp tổ chức mã nguồn rõ ràng và linh hoạt. Mỗi ứng dụng NestJS bao gồm ít nhất một module gốc và có thể có nhiều module con, đại diện cho các phần chức năng cụ thể của ứng dụng.
* Các tính năng:
* Kiến trúc module hóa: Ứng dụng trong NestJS được tổ chức thành các module, giúp quản lý và mở rộng dễ dàng. Mỗi module tập trung vào một chức năng cụ thể, tạo nên cấu trúc rõ ràng cho dự án.
* Sử dụng TypeScript: NestJS được xây dựng trên TypeScript, mang lại lợi ích của kiểu tĩnh và các tính năng hiện đại, giúp mã nguồn rõ ràng và dễ bảo trì.
* Tích hợp với các thư viện và framework khác: NestJS hỗ trợ tích hợp dễ dàng với các thư viện như TypeORM, Mongoose và các framework HTTP server như Express hoặc Fastify, giúp tận dụng các công cụ và middleware hiện có.
* Dependency Injection: Triển khai hệ thống dependency injection mạnh mẽ để quản lý các phụ thuộc.
* Hỗ trợ đa dạng giao thức và công nghệ: NestJS cung cấp các module tích hợp sẵn cho WebSockets, GraphQL, Microservices và hỗ trợ xây dựng các ứng dụng REST API, giúp phát triển các ứng dụng thời gian thực và phân tán một cách thuận lợi.
* Command-Line Interface (CLI) mạnh mẽ: NestJS đi kèm với CLI hỗ trợ tạo và quản lý dự án, giúp tự động hóa các tác vụ như tạo module, controller, service, tăng tốc quá trình phát triển.
* Hệ thống Middleware, Guards, Interceptors và Filters: NestJS cung cấp các thành phần này để xử lý yêu cầu và phản hồi, quản lý xác thực, ủy quyền và xử lý lỗi một cách hiệu quả.
* Microservices : Hỗ trợ tích hợp để xây dựng kiến ​​trúc microservices.
* Mô hình kiến trúc: NestJS áp dụng mô hình kiến trúc module hóa, lấy cảm hứng từ Angular, giúp tổ chức mã nguồn rõ ràng và dễ quản lý. Mỗi ứng dụng NestJS bao gồm ít nhất một module gốc (root module) và có thể có nhiều module con, mỗi module đại diện cho một phần chức năng cụ thể của ứng dụng.



Hình 3.4 Kiến trúc NestJS

* Module: Là đơn vị cơ bản để tổ chức mã nguồn, chứa các thành phần như controllers, providers và các module khác. Module giúp chia ứng dụng thành các phần độc lập, dễ bảo trì và mở rộng.
* Controller: Xử lý các yêu cầu từ client và trả về phản hồi tương ứng. Controllers định nghĩa các route và liên kết chúng với các phương thức xử lý cụ thể.
* Provider: Cung cấp các dịch vụ, như services, repositories, giúp quản lý logic nghiệp vụ và tương tác với dữ liệu. Providers được quản lý thông qua cơ chế Dependency Injection của NestJS.
* Middleware: Các hàm trung gian được thực thi trước khi yêu cầu đến controller, thường dùng cho các tác vụ như xác thực, logging hoặc xử lý lỗi.
* Filter: Xử lý các ngoại lệ (exceptions) phát sinh trong quá trình xử lý yêu cầu, cho phép tùy chỉnh phản hồi lỗi gửi về client.
* Guard: Xác định xem yêu cầu có được phép thực hiện hay không, thường được sử dụng cho việc kiểm tra quyền truy cập và xác thực người dùng.
* Interceptor: Can thiệp vào quá trình xử lý yêu cầu và phản hồi, cho phép thay đổi dữ liệu hoặc thực hiện các tác vụ bổ sung trước hoặc sau khi controller xử lý.
* Với cấu trúc module hóa và các thành phần trên, NestJS cung cấp một kiến trúc linh hoạt và mạnh mẽ, giúp phát triển các ứng dụng server-side hiệu quả và dễ dàng mở rộng.
* Ứng dụng: NestJS là một framework mạnh mẽ dựa trên Node.js, được thiết kế để phát triển các ứng dụng phía server hiệu quả và linh hoạt. Dưới đây là một số ứng dụng phổ biến của NestJS:
* Xây dựng RESTful API: NestJS cung cấp cấu trúc module hóa và hỗ trợ mạnh mẽ cho việc tạo các RESTful API, giúp phát triển các dịch vụ web dễ dàng và hiệu quả
* Phát triển ứng dụng MVC: Với kiến trúc lấy cảm hứng từ Angular, NestJS hỗ trợ mô hình MVC, cho phép xây dựng các ứng dụng web với giao diện người dùng và logic nghiệp vụ được tách biệt rõ ràng.
* Microservices: NestJS hỗ trợ phát triển các ứng dụng microservices, giúp chia nhỏ hệ thống thành các dịch vụ độc lập, dễ quản lý và mở rộng.
* WebSockets: NestJS tích hợp sẵn hỗ trợ cho WebSockets, cho phép xây dựng các ứng dụng thời gian thực như chat trực tuyến, thông báo tức thì và các ứng dụng yêu cầu cập nhật liên tục.
* GraphQL: NestJS cung cấp module tích hợp cho GraphQL, giúp tạo các API linh hoạt và hiệu quả, đáp ứng nhu cầu truy vấn dữ liệu phức tạp từ phía client.
* Tác vụ định kỳ (CRON Jobs): NestJS hỗ trợ thiết lập và quản lý các tác vụ định kỳ, giúp thực hiện các công việc theo lịch trình như sao lưu dữ liệu, gửi email nhắc nhở và các tác vụ bảo trì hệ thống.

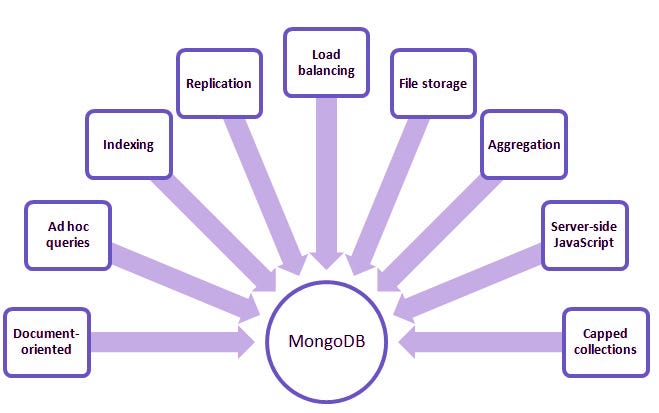
Với khả năng mở rộng và linh hoạt, NestJS được ứng dụng rộng rãi trong việc phát triển các hệ thống backend hiện đại, đáp ứng đa dạng nhu cầu từ các dự án nhỏ đến các ứng dụng doanh nghiệp quy mô lớn.

* Ưu điểm:
* Tính mô-đun : Cung cấp mã có tổ chức và mô-đun, giúp ứng dụng dễ quản lý và mở rộng hơn.
* TypeScript : Cung cấp tất cả các lợi ích của TypeScript, bao gồm tính năng an toàn kiểu và các tính năng JavaScript hiện đại.
* Dependency Injection : Đơn giản hóa việc quản lý các phụ thuộc và thúc đẩy sự kết nối lỏng lẻo, tăng tính linh hoạt và khả năng kiểm thử của ứng dụng.
* Có thể mở rộng : Dễ dàng tích hợp với nhiều thư viện và khuôn khổ khác nhau, NestJS hỗ trợ tích hợp dễ dàng với các thư viện như TypeORM, Mongoose và các framework HTTP server như Express hoặc Fastify, giúp tận dụng các công cụ và middleware hiện có.
* Hỗ trợ đa dạng giao thức và công nghệ: NestJS cung cấp các module tích hợp sẵn cho WebSockets, GraphQL, Microservices và hỗ trợ xây dựng các ứng dụng REST API, giúp phát triển các ứng dụng thời gian thực và phân tán một cách thuận lợi.
* Cộng đồng và hệ sinh thái : Cộng đồng hỗ trợ mạnh mẽ và hệ sinh thái phong phú các mô-đun và công cụ, NestJS có cộng đồng phát triển tích cực và tài liệu chi tiết, giúp lập trình viên dễ dàng tiếp cận và giải quyết các vấn đề trong quá trình phát triển.
* Nhược điểm:
* Đường cong học tập : Các tính năng và khái niệm nâng cao của khung có thể là thách thức đối với người mới bắt đầu.
* Cấu trúc có ý kiến : Mặc dù có lợi cho nhiều người, nhưng cấu trúc có ý kiến ​​có thể gây hạn chế cho một số nhà phát triển.
* Yêu cầu về TypeScript : Các nhà phát triển không quen thuộc với TypeScript có thể cần đầu tư thêm thời gian để học nó.
  + 1. **MongoDB**



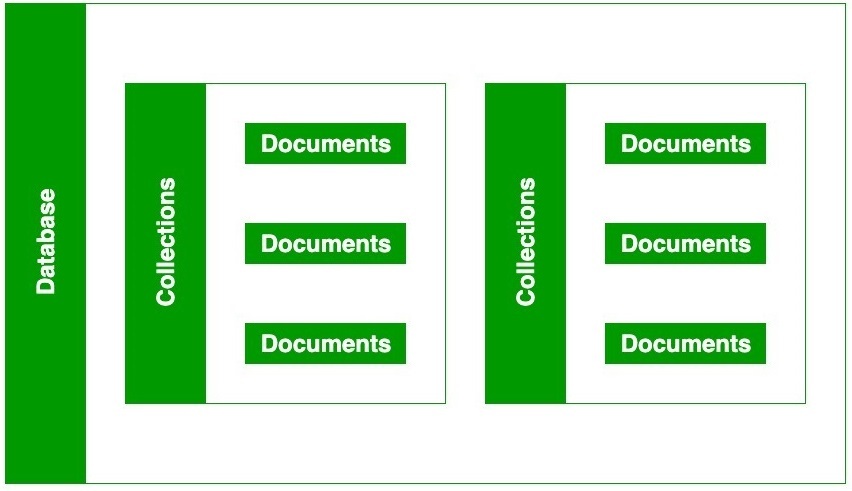
Hình 3*.*5 MongoDB

* MongoDB là một hệ quản trị cơ sở dữ liệu NoSQL mã nguồn mở, được thiết kế quản lý dữ liệu dưới dạng tài liệu JSON linh hoạt, giúp lập trình viên dễ dàng thao tác và quản lý dữ liệu nhờ cấu trúc trực quan và linh hoạt của JSON. Khác với các cơ sở dữ liệu quan hệ truyền thống sử dụng bảng và hàng, MongoDB sử dụng các collection và document, cho phép lưu trữ dữ liệu có cấu trúc linh hoạt và đa dạng.
* MongoDB là giải pháp mạnh mẽ và linh hoạt để xử lý nhu cầu dữ liệu hiện đại. Là cơ sở dữ liệu NoSQL hàng đầu, MongoDB cung cấp thiết kế lược đồ động, cho phép các nhà phát triển lưu trữ và quản lý dữ liệu theo cách phù hợp hoàn toàn với các yêu cầu ứng dụng hiện đại.
* Không giống như các cơ sở dữ liệu quan hệ truyền thống, kiến ​​trúc hướng tài liệu của MongoDB cho phép tính linh hoạt và khả năng mở rộng cao hơn, khiến nó trở thành lựa chọn ưu tiên cho các doanh nghiệp và nhà phát triển muốn xử lý khối lượng lớn dữ liệu phi cấu trúc hoặc bán cấu trúc.
* Các đặc điểm chính:



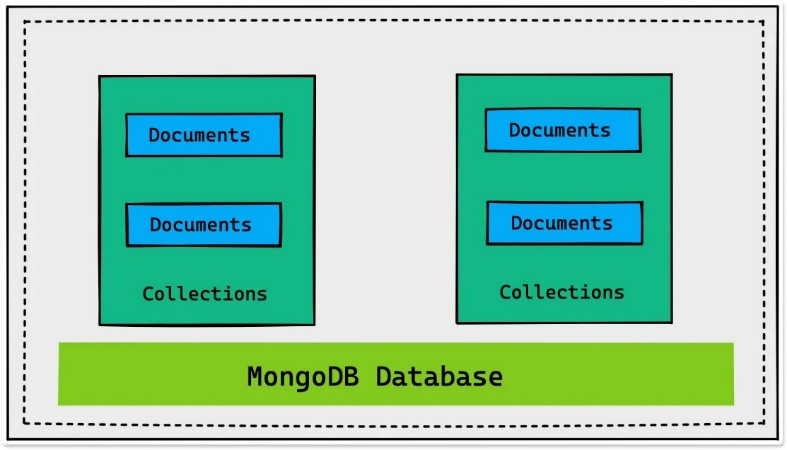
Hình 3.6 Đặc điểm của MongoDB

* Hướng tài liệu**:** Khi lưu trữ thực tế, MongoDB chuyển đổi các tài liệu này sang định dạng BSON (Binary JSON). BSON là một định dạng nhị phân được tối ưu hóa, hỗ trợ nhiều kiểu dữ liệu hơn JSON và cho phép truy xuất nhanh hơn.
* Schema linh hoạt: Không yêu cầu cấu trúc cố định cho dữ liệu, cho phép dễ dàng thay đổi và mở rộng cấu trúc khi cần thiết.
* Khả năng mở rộng: Hỗ trợ sharding và replication, giúp phân phối và sao lưu dữ liệu trên nhiều máy chủ, đảm bảo hiệu suất và độ tin cậy cao.
* Hiệu suất cao: Tối ưu cho việc xử lý các tập dữ liệu lớn và phức tạp, phù hợp với các ứng dụng yêu cầu tốc độ truy cập nhanh.
* Linh hoạt và dễ sử dụng:
* Mô hình dữ liệu linh hoạt: MongoDB lưu trữ dữ liệu dưới dạng tài liệu JSON, cho phép dễ dàng thay đổi cấu trúc dữ liệu mà không cần sửa đổi toàn bộ cơ sở dữ liệu.
* Phát triển nhanh chóng: Với lược đồ động và các công cụ mạnh mẽ, MongoDB hỗ trợ quá trình phát triển và triển khai ứng dụng nhanh hơn.
* Khả năng mở rộng và hiệu suất cao:
* Mở rộng ngang (sharding): MongoDB cho phép phân chia dữ liệu trên nhiều máy chủ, giúp xử lý khối lượng dữ liệu lớn một cách hiệu quả.
* Hiệu suất truy vấn nhanh**:** Sử dụng cơ chế lập chỉ mục và lưu trữ dữ liệu trong bộ nhớ, MongoDB cung cấp tốc độ truy xuất dữ liệu nhanh chóng.
* Tính sẵn sàng và độ tin cậy cao:
* Sao chép dữ liệu (replication): Tính năng này đảm bảo dữ liệu được sao lưu trên nhiều máy chủ, tăng cường khả năng chịu lỗi và sẵn sàng của hệ thống.
* Hỗ trợ đa dạng và tích hợp tốt:
* Hỗ trợ nhiều ngôn ngữ lập trình: MongoDB cung cấp driver cho nhiều ngôn ngữ, giúp dễ dàng tích hợp vào các ứng dụng hiện có.
* Tích hợp với các công cụ phát triển: Nền tảng này tương thích tốt với nhiều framework và công cụ quản lý dữ liệu phổ biến, giúp tối ưu hóa quy trình phát triển.
* Nguyên lý hoạt động:



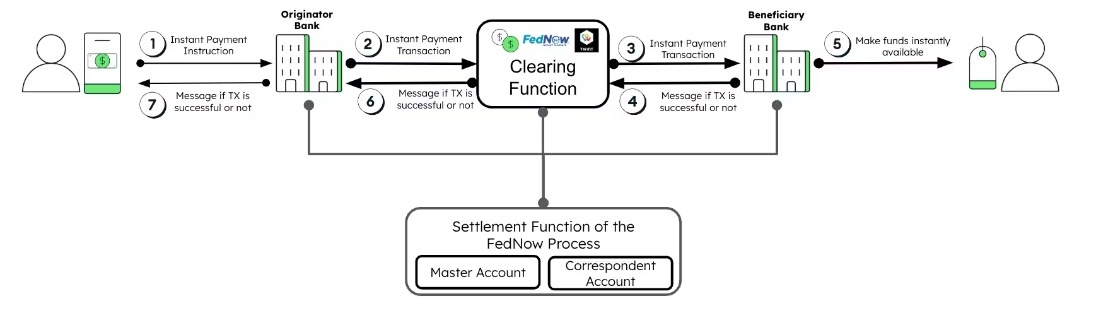
Hình 3.7 Nguyên lý hoạt động

* MongoDB là một máy chủ cơ sở dữ liệu và dữ liệu được lưu trữ trong các cơ sở dữ liệu này. Hay nói cách khác, môi trường MongoDB cung cấp cho một máy chủ mà có thể khởi động và sau đó tạo nhiều cơ sở dữ liệu trên đó bằng MongoDB.
* Cơ sở dữ liệu MongoDB chứa các bộ sưu tập giống như cơ sở dữ liệu MYSQL chứa các bảng. Được phép tạo nhiều cơ sở dữ liệu và nhiều bộ sưu tập.
* Bây giờ bên trong bộ sưu tập, chúng ta có các tài liệu. Các tài liệu này chứa dữ liệu mà chúng ta muốn lưu trữ trong cơ sở dữ liệu MongoDB và một bộ sưu tập duy nhất có thể chứa nhiều tài liệu và không có lược đồ có nghĩa là không cần thiết phải có một tài liệu giống với tài liệu khác.
* Các tài liệu được tạo ra bằng cách sử dụng các trường. Các trường là các cặp khóa-giá trị trong các tài liệu, giống như các cột trong cơ sở dữ liệu quan hệ. Giá trị của các trường có thể là bất kỳ kiểu dữ liệu BSON nào như double, string, boolean, v.v.
* Dữ liệu được lưu trữ trong MongoDB ở định dạng tài liệu BSON. Ở đây, BSON là viết tắt của Binary representation of JSON documents. Hay nói cách khác, ở phần phụ trợ, máy chủ MongoDB chuyển đổi dữ liệu JSON thành dạng nhị phân được gọi là BSON và BSON này được lưu trữ và truy vấn hiệu quả hơn.
* Trong các tài liệu MongoDB được phép lưu trữ dữ liệu lồng nhau. Việc lồng nhau dữ liệu này cho phép tạo các mối quan hệ phức tạp giữa dữ liệu và lưu trữ chúng trong cùng một tài liệu, giúp việc làm việc và truy xuất dữ liệu cực kỳ hiệu quả so với SQL. Trong SQL, cần viết các phép nối phức tạp để lấy dữ liệu từ bảng 1 và bảng 2. Kích thước tối đa của tài liệu BSON là 16MB.
* Cấu trúc cơ sở dữ liệu của MongoDB:



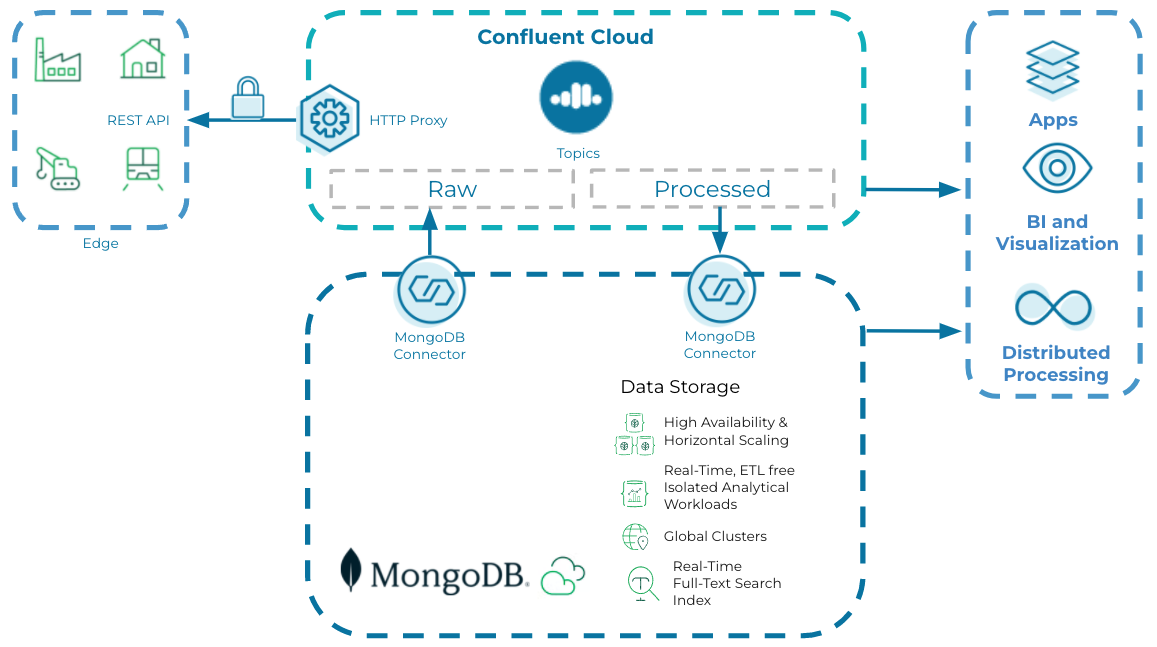
Hình 3*.*8 Kiến trúc MongoDB

* Database: Nói một cách đơn giản, nó có thể được gọi là nơi lưu trữ vật lý cho dữ liệu. Mỗi cơ sở dữ liệu có một tập hợp tệp riêng trên hệ thống tệp với nhiều cơ sở dữ liệu hiện có trên một máy chủ MongoDB.
* Collection: Một nhóm database documents có thể được gọi là collection. RDBMS tương đương với một bộ sưu tập là một bảng. Toàn bộ collection tồn tại trong một cơ sở dữ liệu duy nhất. Không có lược đồ khi nói đến bộ sưu tập. Bên trong collection, các tài liệu khác nhau có thể có các trường khác nhau, nhưng hầu hết các document trong collection đều có cùng mục đích hoặc phục vụ cùng một mục tiêu cuối cùng.
* Document: Một tập hợp các cặp key-value có thể được chỉ định làm document. Documents được liên kết với các lược đồ động – dynamic schemas. Lợi ích của việc có các dynamic schemas là một document trong một collection không nhất thiết phải có cùng cấu trúc hoặc các trường. Ngoài ra, các common fields trong collection document có thể có nhiều loại dữ liệu khác nhau.
* Các tính năng chính:
* Schema-less Database: Đây là tính năng tuyệt vời do MongoDB cung cấp. Một cơ sở dữ liệu Schema-less có nghĩa là một bộ sưu tập có thể chứa nhiều loại tài liệu khác nhau trong đó . Hay nói cách khác, trong cơ sở dữ liệu MongoDB, một bộ sưu tập duy nhất có thể chứa nhiều tài liệu và các tài liệu này có thể bao gồm số lượng trường, nội dung và kích thước khác nhau. Không nhất thiết một tài liệu phải giống với một tài liệu khác như trong cơ sở dữ liệu quan hệ . Nhờ tính năng tuyệt vời này, MongoDB cung cấp tính linh hoạt tuyệt vời cho cơ sở dữ liệu.
* Document Oriented: Trong MongoDB, tất cả dữ liệu được lưu trữ trong documents thay vì table như trong RDBMS . Trong các documents này, dữ liệu được lưu trữ trong các trường (cặp khóa-giá trị) thay vì hàng và cột, giúp dữ liệu linh hoạt hơn nhiều so với RDBMS. Và mỗi document chứa id đối tượng duy nhất của nó.
* Lập chỉ mục: Trong cơ sở dữ liệu MongoDB, mọi trường trong các tài liệu đều được lập chỉ mục bằng các chỉ mục chính và phụ, điều này giúp việc lấy hoặc tìm kiếm dữ liệu từ nhóm dữ liệu dễ dàng hơn và mất ít thời gian hơn. Nếu dữ liệu không được lập chỉ mục, thì cơ sở dữ liệu sẽ tìm kiếm từng tài liệu bằng truy vấn đã chỉ định, điều này mất nhiều thời gian và không hiệu quả.
* Khả năng mở rộng: MongoDB cung cấp khả năng mở rộng theo chiều ngang với sự trợ giúp của sharding. Sharding có nghĩa là phân phối dữ liệu trên nhiều máy chủ, ở đây một lượng lớn dữ liệu được phân vùng thành các khối dữ liệu bằng cách sử dụng khóa shard và các khối dữ liệu này được phân phối đều trên các shard nằm trên nhiều máy chủ vật lý. Nó cũng sẽ thêm các máy mới vào cơ sở dữ liệu đang chạy.
* Sao chép: MongoDB cung cấp tính khả dụng cao và dự phòng với sự trợ giúp của tính năng sao chép, nó tạo ra nhiều bản sao dữ liệu và gửi các bản sao này đến một máy chủ khác nhau để nếu một máy chủ bị lỗi, dữ liệu sẽ được lấy từ máy chủ khác.
* Tổng hợp: Cho phép thực hiện các hoạt động trên dữ liệu được nhóm và nhận được một kết quả duy nhất hoặc kết quả được tính toán . Tương tự như mệnh đề GROUPBY của SQL. Nó cung cấp ba tổng hợp khác nhau, tức là đường ống tổng hợp, hàm map-reduce và các phương pháp tổng hợp mục đích đơn
* Hiệu suất cao: Hiệu suất của MongoDB rất cao và tính bền vững của dữ liệu khi so sánh với các cơ sở dữ liệu khác do các tính năng như khả năng mở rộng, lập chỉ mục, sao chép, v.v.
* MongoDB là một cơ sở dữ liệu NoSQL phổ biến được biết đến với tính linh hoạt, khả năng mở rộng và hiệu suất. Nó được sử dụng rộng rãi trong nhiều ứng dụng khác nhau trên nhiều ngành công nghiệp khác nhau. Sau đây là một số cách sử dụng phổ biến của MongoDB:
* Hệ thống quản lý nội dung (CMS) : Sơ đồ linh hoạt và khả năng truy vấn mạnh mẽ của MongoDB khiến nó trở thành lựa chọn lý tưởng cho các hệ thống quản lý nội dung. Nó có thể xử lý hiệu quả nhiều loại nội dung và cấu trúc khác nhau, cho phép các giải pháp quản lý nội dung năng động và có thể mở rộng.
* Nền tảng thương mại điện tử : Nền tảng thương mại điện tử được hưởng lợi từ khả năng lưu trữ và truy xuất nhanh khối lượng lớn dữ liệu sản phẩm của MongoDB. Sơ đồ linh hoạt của nó hỗ trợ danh mục sản phẩm động, hồ sơ người dùng, giỏ hàng và lịch sử giao dịch.



Hình 3.9 Real-time payment với MongoDB

* Phân tích thời gian thực : MongoDB rất phù hợp cho các ứng dụng phân tích thời gian thực do khả năng thu thập và truy vấn dữ liệu hiệu suất cao. Nó có thể xử lý khối lượng lớn dữ liệu theo thời gian thực, lý tưởng cho việc giám sát, phát hiện gian lận và đưa ra các khuyến nghị được cá nhân hóa.
* Internet vạn vật (IoT) : Các ứng dụng IoT tạo ra lượng dữ liệu khổng lồ từ các cảm biến và thiết bị. Khả năng mở rộng và mô hình dữ liệu linh hoạt của MongoDB cho phép lưu trữ và xử lý dữ liệu này một cách hiệu quả, cho phép phân tích và ra quyết định theo thời gian thực cho các hệ thống IoT.



Hình 3.10 IoT của MongoDB

* Ứng dụng chơi game : Ứng dụng chơi game tạo ra các cấu trúc dữ liệu phức tạp, chẳng hạn như hồ sơ người chơi, điểm số, thành tích và trạng thái trò chơi. Mô hình dựa trên tài liệu của MongoDB cho phép lưu trữ và truy xuất dữ liệu này hiệu quả, hỗ trợ trải nghiệm chơi game hiệu suất cao.
* Quản lý và phân tích nhật ký : Các tổ chức sử dụng MongoDB để lưu trữ và phân tích dữ liệu nhật ký từ nhiều nguồn khác nhau. Khả năng xử lý khối lượng lớn dữ liệu phi cấu trúc của MongoDB làm cho nó trở nên lý tưởng để ghi nhật ký, giám sát và khắc phục sự cố cho các ứng dụng và cơ sở hạ tầng.
* Quản lý quan hệ khách hàng (CRM) : Hệ thống CRM sử dụng MongoDB để quản lý dữ liệu khách hàng, tương tác và kênh bán hàng. Khả năng xử lý các mối quan hệ phức tạp và dữ liệu phi cấu trúc cho phép các chiến lược tương tác với khách hàng hiệu quả và được cá nhân hóa hơn.
* Mạng xã hội : Các ứng dụng mạng xã hội yêu cầu một cơ sở dữ liệu có thể xử lý các mối quan hệ phức tạp, nội dung do người dùng tạo và tương tác thời gian thực. Tính linh hoạt và khả năng mở rộng của MongoDB khiến nó trở thành lựa chọn tuyệt vời để xây dựng mạng xã hội và nền tảng cộng đồng.
* Ứng dụng dữ liệu lớn : MongoDB được sử dụng trong các ứng dụng dữ liệu lớn vì khả năng lưu trữ và xử lý khối lượng lớn các loại dữ liệu đa dạng. Nó tích hợp tốt với các công nghệ dữ liệu lớn như Hadoop và Spark, cho phép phân tích và xử lý dữ liệu nâng cao.
* Hệ thống chăm sóc sức khỏe : Các ứng dụng chăm sóc sức khỏe sử dụng MongoDB để quản lý hồ sơ bệnh nhân, dữ liệu lâm sàng và hình ảnh y tế. Sơ đồ linh hoạt của nó cho phép lưu trữ hiệu quả dữ liệu chăm sóc sức khỏe phức tạp, hỗ trợ chăm sóc bệnh nhân và phân tích dữ liệu tốt hơn.
* Ưu điểm:



Hình 3.11 Ưu điểm MongoDB

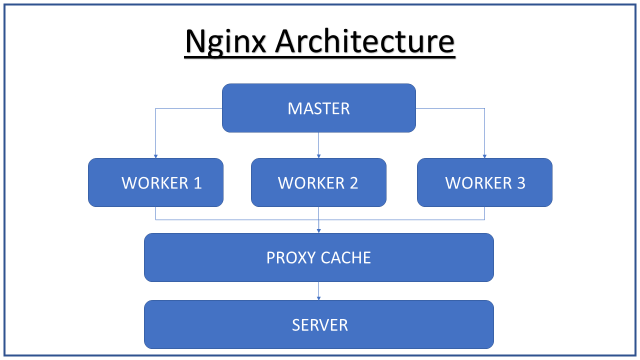
* Đây là cơ sở dữ liệu NoSQL không có lược đồ, không cần thiết kế lược đồ của cơ sở dữ liệu khi làm việc với MongoDB.
* Không hỗ trợ thao tác nối.
* Nó cung cấp tính linh hoạt cao cho các trường trong tài liệu.
* Nó chứa dữ liệu không đồng nhất.
* Nó cung cấp hiệu suất cao, tính khả dụng và khả năng mở rộng.
* Nó hỗ trợ Địa không gian một cách hiệu quả.
* Đây là cơ sở dữ liệu hướng tài liệu và dữ liệu được lưu trữ trong các tài liệu BSON.
* Nó cũng hỗ trợ chuyển đổi ACID nhiều tài liệu (chuỗi từ MongoDB 4.0).
* Nó không yêu cầu bất kỳ lệnh SQL nào.
* Nó dễ dàng tích hợp với Big Data Hadoop.
* Nhược điểm:
* Nó sử dụng bộ nhớ lớn để lưu trữ dữ liệu.
* Không được phép lưu trữ dữ liệu trong tài liệu có dung lượng quá 16MB.
* Việc lồng dữ liệu trong BSON cũng bị hạn chế, không được phép lồng dữ liệu quá 100 cấp.
  + 1. **Nginx**



Hình 3.12 Nginx

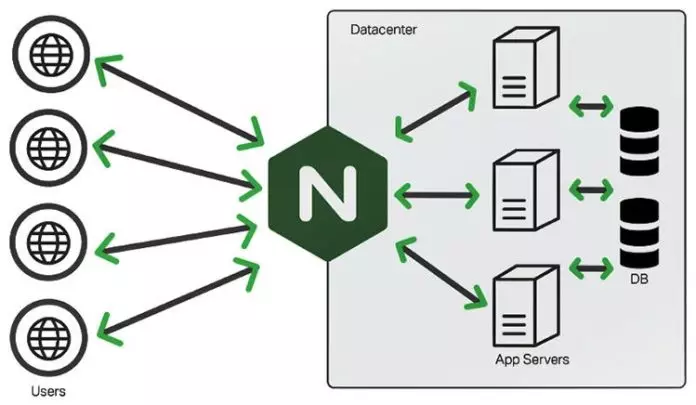
* NGINX là một phần mềm máy chủ web mã nguồn mở mạnh mẽ, được phát triển bởi Igor Sysoev và phát hành lần đầu vào năm 2004. Nó được thiết kế để xử lý đồng thời hàng nghìn kết nối với hiệu suất cao và sử dụng tài nguyên thấp. Ngoài chức năng máy chủ web, NGINX còn đóng vai trò như một reverse proxy, cân bằng tải (load balancer), bộ nhớ đệm HTTP (HTTP caching), và proxy cho các giao thức email như IMAP, POP3 và SMTP. Với kiến trúc hướng sự kiện và bất đồng bộ, NGINX đã trở thành lựa chọn phổ biến cho nhiều doanh nghiệp và tổ chức trên toàn thế giới.
* Nginx là một máy chủ web chuyên dụng đã giải quyết được các vấn đề về hiệu quả và cung cấp cho chúng ta giải pháp tối ưu để xử lý hàng nghìn yêu cầu cùng lúc.
* Máy chủ web cho proxy ngược, lưu trữ đệm và cân bằng tải: Proxy ngược chấp nhận yêu cầu từ máy khách, chuyển tiếp yêu cầu đến máy chủ có thể thực hiện yêu cầu đó và trả về phản hồi từ máy chủ cho máy khách.
* Caching là một kỹ thuật lưu trữ một bản sao của một tài nguyên nhất định và phục vụ lại khi được yêu cầu. Khi bộ nhớ đệm web có một tài nguyên được yêu cầu trong kho lưu trữ của nó, nó sẽ chặn yêu cầu và trả về bản sao của nó thay vì tải xuống lại từ máy chủ gốc.
* Bộ cân bằng tải phân phối các yêu cầu của máy khách đến một nhóm máy chủ, trong đó nó có thể xử lý các yêu cầu đồng thời mà không gây quá tải cho một máy chủ cụ thể.
* Các tính năng khác của Nginx như sau:
* Nó cung cấp khả năng của máy chủ HTTP.
* Được thiết kế để mang lại sự ổn định và hiệu suất tối đa.
* Hoạt động như một máy chủ proxy cho email (IMAP, POP3 và SMTP).
* Nó sử dụng kiến ​​trúc theo sự kiện và không phân luồng để cung cấp ít tính toán CPU hơn cho mỗi yêu cầu được phục vụ.
* Nó cung cấp khả năng mở rộng.
* Giảm thời gian chờ đợi của khách hàng.
* Có thể thực hiện nâng cấp trong khi Nginx đang lưu trữ trang web mà không bị ngừng hoạt động.
* Kiến trúc:

Nginx sử dụng kiến ​​trúc Master-Slave, trong đó chúng ta có một master định tuyến lại yêu cầu của chúng ta đến bất kỳ worker nào bên dưới nó bằng cách phân phối tải trên máy chủ, sau đó bộ đệm Proxy được tìm kiếm để phản hồi nhanh hơn, nếu không thì sau khi không thực hiện được, trang web sẽ được tải từ chính bộ nhớ. Một hình ảnh minh họa sẽ giúp hiểu rõ hơn về cấu trúc này.



Hình 3.13 Kiến trúc của Nginx

* Các thiết lập chính cần thiết cho Nginx được lưu trữ trong một tệp có tên **nginx.conf** . Tệp này chủ yếu được cấu trúc thành nhiều ngữ cảnh. Một số thiết lập cấu hình cho Nginx:
* worker\_processes: Một tiến trình worker về cơ bản là số lượng worker mà master sẽ kiểm soát. Nói chính xác hơn, đây là một tiến trình đơn luồng. Vì vậy, nếu có bộ xử lý đa lõi như 8 lõi, có thể thiết lập 8 worker\_processes để tận dụng tối đa băng thông đĩa và giúp nó thực hiện công việc đòi hỏi nhiều CPU.
* worker\_connections: Đây là số lượng kết nối đồng thời với các máy khách khác nhau. Mặc định là 512 hoặc 1024. Vì vậy, giả sử có 8 worker\_processes và 512 worker\_connections, số lượng máy khách tối đa có thể phục vụ là worker\_processes \* worker\_connections, tức là 8\*512 = 4096 kết nối.
* access\_log & error\_log: Theo mặc định, máy chủ Nginx ghi các sự kiện của nó vào 2 loại nhật ký như đã đề cập. Các hoạt động của tất cả khách truy cập, yêu cầu dữ liệu từ máy chủ được ghi vào access\_log, có thể tìm thấy những tệp nào đã được truy cập và cách Nginx phản hồi chúng, IP của máy khách và nhiều thông tin khác. Nhưng nếu Nginx gặp bất kỳ sự cố và trục trặc nào, thì nó sẽ ghi lại tất cả thông tin này vào nhật ký lỗi để gỡ lỗi thêm.
* gzip: Nếu muốn tận dụng tối đa hiệu suất có sẵn, nó sẽ nén tất cả dữ liệu phản hồi. Tuy nhiên, nó sử dụng nhiều tài nguyên CPU, có thể làm chậm Nginx nếu sử dụng tùy chọn này cho tất cả các mục, vì vậy chỉ nên sử dụng tùy chọn này cho các mục lớn.
* Nguyên lý hoạt động:



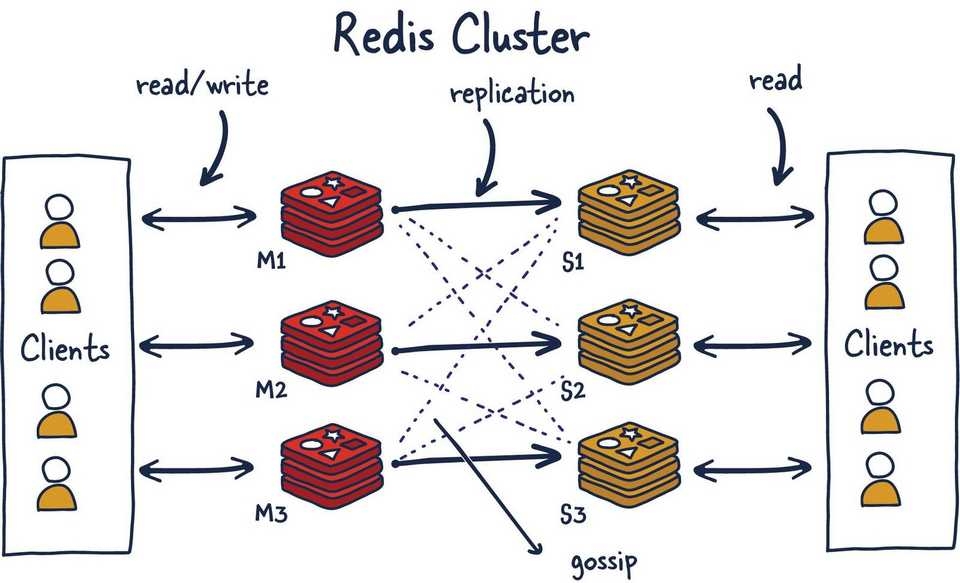
Hình 3.14 Nguyên lý hoạt động

* NGINX hoạt động dựa trên kiến trúc bất đồng bộ và hướng sự kiện, cho phép xử lý đồng thời hàng nghìn kết nối mà không tiêu tốn nhiều tài nguyên. Thay vì tạo một luồng (thread) riêng cho mỗi yêu cầu, NGINX sử dụng một tiến trình chính (master process) để quản lý nhiều tiến trình con (worker processes). Mỗi tiến trình con này có thể xử lý nhiều kết nối cùng lúc thông qua các sự kiện, giúp tăng hiệu suất và khả năng mở rộng của máy chủ.
* Cụ thể, khi một yêu cầu đến, tiến trình con sẽ nhận và xử lý yêu cầu đó mà không cần chờ đợi các yêu cầu khác hoàn thành. Điều này đạt được nhờ vào mô hình xử lý sự kiện không đồng bộ, nơi các kết nối được quản lý thông qua các sự kiện và callback, giúp tránh tình trạng chờ đợi và tối ưu hóa việc sử dụng tài nguyên.
* Ứng dụng:
* Một máy chủ web. Đây là phổ biến nhất vì hiệu suất và khả năng mở rộng của nó.
* Một máy chủ proxy ngược. NGINX thực hiện điều này bằng cách hướng yêu cầu của máy khách đến máy chủ back-end thích hợp.
* Cân bằng tải. Nó tự động phân phối tải lưu lượng mạng mà không cần cấu hình thủ công.
* Cổng API. Điều này rất hữu ích cho việc định tuyến yêu cầu, xác thực và xử lý ngoại lệ.
* Tường lửa cho các ứng dụng web. Điều này bảo vệ ứng dụng bằng cách lọc các yêu cầu mạng đến và đi trên máy chủ.
* Một bộ nhớ đệm. NGINX hoạt động như một bộ nhớ đệm để giúp lưu trữ dữ liệu cho các yêu cầu trong tương lai.
* Bảo vệ chống lại các cuộc tấn công từ chối dịch vụ phân tán (DDoS).
* K8s. Chúng tự động hóa việc triển khai, mở rộng quy mô và quản lý các ứng dụng trong bộ chứa.
* Một proxy sidecar. Điều này định tuyến lưu lượng truy cập đến và đi từ container mà nó chạy cùng nhau.
* Ưu điểm:
* Giảm thời gian chờ đợi để tải một trang web. Không phải lo lắng về độ trễ cao trên trang web của mình, do đó mang lại trải nghiệm người dùng tốt.
* Tăng tốc hiệu suất bằng cách định tuyến lưu lượng truy cập đến máy chủ web theo cách tăng tốc độ tổng thể. Tính năng này cung cấp trải nghiệm duyệt web tốt cho người dùng.
* Hoạt động như một cân bằng tải không tốn kém và mạnh mẽ.
* Cung cấp khả năng mở rộng và khả năng xử lý các yêu cầu đồng thời.
* Cho phép nâng cấp nhanh chóng mà không cần thời gian chết.
* Nhược điểm:
* Cân bằng tải còn hạn chế so với các máy chủ khác.
* Cơ chế lưu trữ, cung cấp các lệnh cần phải cải thiện.
* Việc điều hướng giữa trang chủ Nginx và bộ phận hỗ trợ khách hàng thực hiện rất khó.
  + 1. **Redis**



Hình 3.15 Redis

* Redis là một cấu trúc dữ liệu trong bộ nhớ (in-memory) được sử dụng để truy cập dữ liệu nhanh hơn. Nó được sử dụng để lưu trữ dữ liệu cần được truy cập thường xuyên và nhanh chóng. Nó hỗ trợ nhiều cấu trúc dữ liệu như chuỗi (strings), danh sách (lists), tập hợp (sets), bảng băm (hashes) và tập hợp có thứ tự (sorted sets). Redis thường được sử dụng như một cơ sở dữ liệu, bộ nhớ cache hoặc message broker.
* Các tính năng chính:
* Tốc độ cao: Với việc lưu trữ dữ liệu chủ yếu trong bộ nhớ, Redis cung cấp khả năng truy xuất và ghi dữ liệu nhanh chóng, phù hợp cho các ứng dụng yêu cầu hiệu suất cao.
* Hỗ trợ nhiều cấu trúc dữ liệu: Redis không chỉ hỗ trợ lưu trữ dữ liệu dạng key-value mà còn cung cấp các cấu trúc dữ liệu phức tạp như danh sách, tập hợp và bảng băm, giúp xử lý linh hoạt hơn trong các tình huống khác nhau.
* Khả năng mở rộng: Redis hỗ trợ phân tán dữ liệu và có thể mở rộng dễ dàng để đáp ứng nhu cầu của các ứng dụng quy mô lớn.
* Tính bền vững: Mặc dù hoạt động chủ yếu trong bộ nhớ, Redis cung cấp các cơ chế để lưu trữ dữ liệu trên đĩa, đảm bảo tính bền vững và khả năng phục hồi sau sự cố
* Cách hoạt động:
* Lưu trữ dữ liệu trong bộ nhớ: Redis lưu trữ tất cả dữ liệu trực tiếp trong bộ nhớ RAM, giúp truy xuất và ghi dữ liệu với tốc độ rất nhanh. Điều này làm cho Redis trở thành lựa chọn lý tưởng cho các ứng dụng yêu cầu hiệu suất cao và độ trễ thấp.
* Hỗ trợ nhiều cấu trúc dữ liệu: Redis không chỉ hỗ trợ lưu trữ dữ liệu dạng key-value mà còn cung cấp các cấu trúc dữ liệu phức tạp như danh sách, tập hợp và bảng băm, giúp xử lý linh hoạt hơn trong các tình huống khác nhau.
* Khả năng mở rộng và phân tán: Redis hỗ trợ phân tán dữ liệu và có thể mở rộng dễ dàng để đáp ứng nhu cầu của các ứng dụng quy mô lớn. Điều này cho phép Redis xử lý hàng triệu yêu cầu mỗi giây và đảm bảo tính sẵn sàng cao cho các ứng dụng.
* Tính bền vững: Mặc dù hoạt động chủ yếu trong bộ nhớ, Redis cung cấp các cơ chế để lưu trữ dữ liệu trên đĩa, đảm bảo tính bền vững và khả năng phục hồi sau sự cố. Redis hỗ trợ hai phương pháp lưu trữ dữ liệu trên đĩa gồm RDB (Redis Database Backup) và AOF (Append Only File).
* Tính năng sao chép và phân mảnh: Redis hỗ trợ sao chép dữ liệu giữa các máy chủ (replication) và phân mảnh dữ liệu (sharding), giúp tăng cường khả năng chịu lỗi và mở rộng quy mô hệ thống. Điều này cho phép Redis duy trì tính nhất quán và sẵn sàng cao cho các ứng dụng phân tán.
* Ứng dụng của Redis:



Hình 3.16 Ứng dụng của Redis

* Bộ nhớ cache: Redis thường được sử dụng để lưu trữ dữ liệu tạm thời, giúp giảm tải cho cơ sở dữ liệu chính và tăng tốc độ truy xuất dữ liệu.
* Hàng đợi tin nhắn: Với khả năng xử lý các thao tác trên danh sách và tập hợp, Redis là lựa chọn phổ biến cho việc quản lý hàng đợi tin nhắn trong các hệ thống phân tán.
* Quản lý phiên người dùng: Redis được sử dụng để lưu trữ thông tin phiên người dùng, giúp quản lý trạng thái người dùng trong các ứng dụng web và di động.
* Ưu điểm:
* Hiệu suất cao: Redis vượt trội về mặt hiệu suất nhờ bản chất trong bộ nhớ. Nó có thể cung cấp các hoạt động đọc và ghi cực nhanh, phù hợp với các tình huống mà độ trễ thấp là rất quan trọng.
* API đơn giản và dễ sử dụng: Redis có API đơn giản bao gồm các lệnh đơn giản và trực quan, giúp các nhà phát triển dễ dàng sử dụng và tích hợp vào ứng dụng của họ.
* Cấu trúc dữ liệu: Redis hỗ trợ nhiều cấu trúc dữ liệu, bao gồm chuỗi, danh sách, tập hợp, băm, v.v. Tính linh hoạt này cho phép các nhà phát triển mô hình hóa dữ liệu của họ hiệu quả hơn, chọn đúng cấu trúc dữ liệu cho nhiệm vụ đang thực hiện.
* Hoạt động nguyên tử: Redis hỗ trợ các hoạt động nguyên tử trên các cấu trúc dữ liệu này, rất phù hợp với các tình huống đòi hỏi tính nhất quán và độ tin cậy trong các hoạt động nhiều bước.
* Tùy chọn duy trì: Mặc dù Redis là cơ sở dữ liệu trong bộ nhớ, nhưng nó cung cấp các tùy chọn lưu trữ như ảnh chụp nhanh và tệp chỉ thêm. Điều này cho phép người dùng định cấu hình mức độ bền cần thiết cho trường hợp sử dụng cụ thể của họ.
* Sao chép và tính khả dụng cao: Redis hỗ trợ sao chép master-slave, cho phép tạo bản sao của máy chủ master. Điều này cung cấp khả năng sẵn sàng cao và khả năng chịu lỗi trong trường hợp nút master bị lỗi.
* Hạn chế:
* Độ phức tạp của cơ chế duy trì: Redis là cơ sở dữ liệu trong bộ nhớ và mặc dù hỗ trợ tính bền bỉ, các cơ chế để đạt được tính bền bỉ này (chẳng hạn như ảnh chụp nhanh và tệp chỉ thêm) có thể phức tạp và có thể ảnh hưởng đến hiệu suất.
* Khả năng truy vấn hạn chế: Redis không phải là cơ sở dữ liệu quan hệ đầy đủ và thiếu khả năng truy vấn phức tạp của cơ sở dữ liệu truyền thống. Nó chủ yếu hoạt động trên các cặp khóa-giá trị và cung cấp các cấu trúc dữ liệu cơ bản như chuỗi, danh sách, tập hợp và băm.
* Sử dụng bộ nhớ: Vì Redis lưu trữ tất cả dữ liệu trong bộ nhớ nên lượng dữ liệu mà nó có thể xử lý bị giới hạn bởi bộ nhớ hệ thống khả dụng. Các tập dữ liệu lớn có thể yêu cầu tài nguyên bộ nhớ đáng kể, đây có thể là một hạn chế tiềm ẩn.
* Bản chất luồng đơn: Redis theo truyền thống sử dụng kiến ​​trúc vòng lặp sự kiện luồng đơn. Mặc dù thiết kế này đơn giản hóa một số khía cạnh của hệ thống, nhưng nó có thể hạn chế hiệu suất trên các hệ thống đa lõi. Tuy nhiên, các phiên bản gần đây của Redis đã giới thiệu đa luồng ở một số phần để giải quyết hạn chế này.
* Không có tính năng bảo mật tích hợp: Redis ban đầu thiếu các tính năng bảo mật tích hợp và được khuyến nghị chạy trong môi trường đáng tin cậy. Mặc dù các phiên bản mới hơn có cơ chế xác thực nhưng việc cấu hình và quản lý các tính năng bảo mật này đúng cách vẫn rất quan trọng.
  + 1. **MinIO**



Hình 3.17 MinIO

* MinIO là một hệ thống lưu trữ đối tượng mã nguồn mở, tương thích với API của Amazon S3, được thiết kế để lưu trữ và quản lý dữ liệu phi cấu trúc như hình ảnh, video, tệp tin nhật ký, bản sao lưu và hình ảnh container. Với khả năng hỗ trợ kích thước đối tượng lên đến 50TB, MinIO phù hợp cho việc lưu trữ và quản lý các tệp tin lớn.
* MinIO hoạt động theo mô hình client-server, cho phép các ứng dụng web hoặc di động gửi yêu cầu đến máy chủ MinIO để thao tác với dữ liệu. Nó hỗ trợ các tính năng như phân tán dữ liệu, chịu lỗi, cân bằng tải và mở rộng hệ thống dễ dàng. Để bảo vệ dữ liệu, MinIO sử dụng kỹ thuật Erasure Coding, chia nhỏ và lưu trữ dữ liệu trên nhiều node, giúp tái tạo dữ liệu khi gặp sự cố. Ngoài ra, MinIO hỗ trợ mã hóa dữ liệu và kết nối bảo mật TLS/SSL, đảm bảo an toàn trong việc truy xuất và truyền tải dữ liệu.
* Người dùng có thể quản lý hệ thống MinIO qua giao diện web đơn giản hoặc công cụ dòng lệnh, dễ dàng thực hiện các tác vụ như tạo bucket, tải lên tệp và quản lý đối tượng. Với tính năng tương thích, hiệu suất cao và dễ sử dụng, MinIO là lựa chọn lý tưởng cho các ứng dụng lưu trữ dữ liệu lớn, hệ thống đa phương tiện, sao lưu và các giải pháp dữ liệu lớn khác.
* Các tính năng:
* Tương thích với Amazon S3: MinIO hoàn toàn tương thích với API của Amazon S3, cho phép tích hợp dễ dàng với các ứng dụng và công cụ đã được phát triển cho S3.
* Hiệu suất cao và khả năng mở rộng: MinIO được thiết kế để xử lý lượng lớn dữ liệu với hiệu suất cao và khả năng mở rộng linh hoạt, phù hợp cho các ứng dụng yêu cầu tốc độ truy xuất dữ liệu nhanh chóng.
* Bảo mật mạnh mẽ: MinIO hỗ trợ mã hóa dữ liệu cả khi lưu trữ và khi truyền tải, đảm bảo an toàn cho dữ liệu.
* Quản lý dữ liệu linh hoạt: MinIO cung cấp các công cụ và API để quản lý dữ liệu hiệu quả, bao gồm khả năng tạo bucket, tải lên và tải xuống tệp, cũng như hỗ trợ các thao tác như sao chép, di chuyển và xóa đối tượng.
* Hỗ trợ đa nền tảng: MinIO có thể triển khai trên nhiều hệ điều hành và môi trường khác nhau, bao gồm Linux, Windows, macOS, Docker và Kubernetes, giúp dễ dàng tích hợp vào hạ tầng hiện có.
* Quản lý vòng đời dữ liệu: MinIO hỗ trợ các chính sách quản lý vòng đời dữ liệu, cho phép tự động hóa các tác vụ như chuyển dữ liệu giữa các lớp lưu trữ, xóa dữ liệu cũ và sao lưu dữ liệu.
* Tính sẵn sàng cao và chịu lỗi: MinIO hỗ trợ cấu trúc phân tán, cho phép lưu trữ dữ liệu trên nhiều nút và đảm bảo tính sẵn sàng cao, giúp hệ thống chịu lỗi và phục hồi nhanh chóng khi gặp sự cố.
* Ưu điểm:
* Chi phí thấp: Là phần mềm mã nguồn mở, MINIO giúp giảm chi phí so với các giải pháp lưu trữ đám mây khác.
* Dễ triển khai và quản lý: MINIO có cấu hình đơn giản và dễ triển khai trên nhiều hệ thống khác nhau.
* Khả năng tích hợp cao: MINIO có thể tích hợp vào các ứng dụng và dịch vụ hiện tại nhờ vào API S3 tương thích.
* Cách hoạt động:
* Tổ chức dữ liệu bằng Buckets:

MinIO sử dụng khái niệm "bucket" để tổ chức dữ liệu. Mỗi bucket có thể chứa một số lượng đối tượng (objects) không giới hạn. Mặc dù MinIO hỗ trợ cấu trúc thư mục lồng nhau thông qua tính năng "prefixing", nhưng thực chất, MinIO không sử dụng hệ thống tệp truyền thống với thư mục và tệp con.

* Lưu trữ đối tượng:

Dữ liệu được lưu trữ dưới dạng các đối tượng, mỗi đối tượng có thể có kích thước lên đến 50TB. Điều này cho phép MinIO xử lý hiệu quả các tệp lớn mà không gặp phải hạn chế về kích thước tệp như trong hệ thống tệp truyền thống.

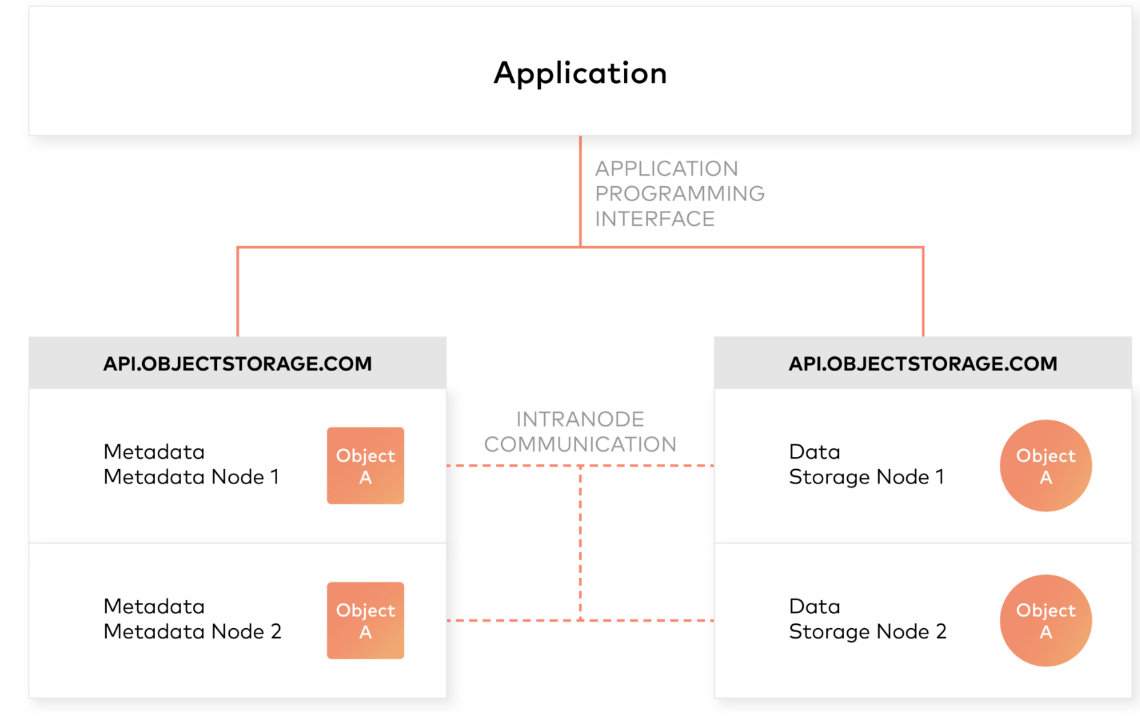
* Truy cập và quản lý dữ liệu:

MinIO cung cấp giao diện API tương thích với S3, cho phép các ứng dụng và dịch vụ tương tác với dữ liệu thông qua các lệnh HTTP như PUT, GET, DELETE. Điều này giúp tích hợp MinIO vào các ứng dụng hiện có một cách dễ dàng.

* Bảo mật và quản lý truy cập:

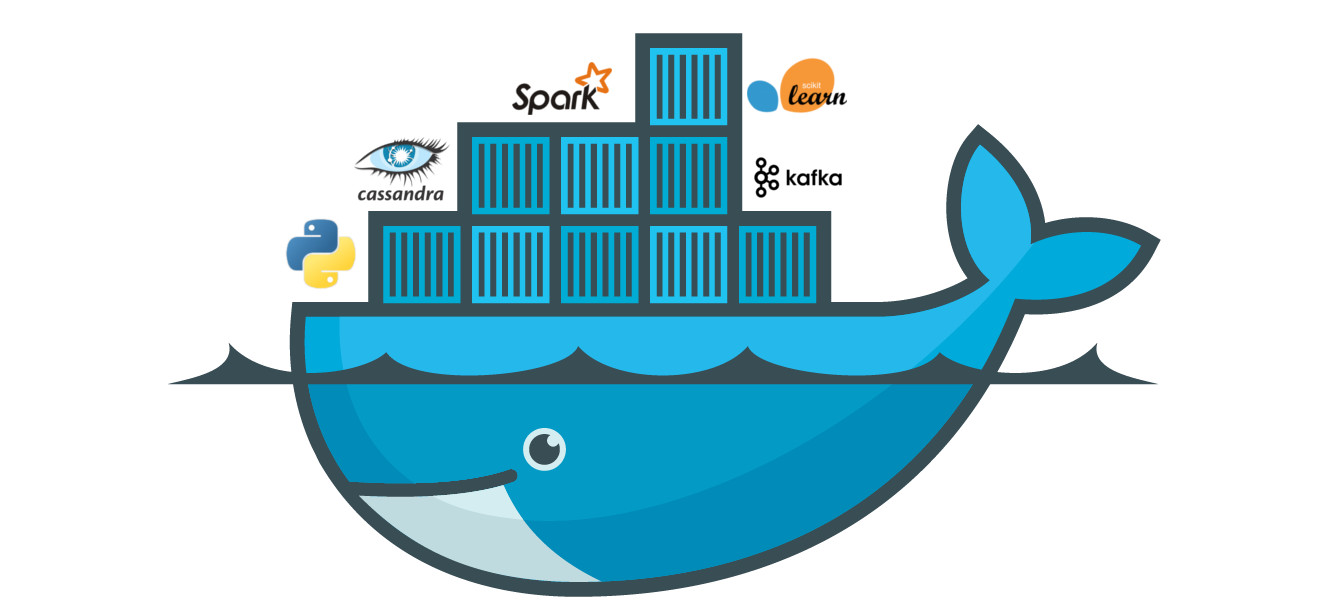
MinIO hỗ trợ các cơ chế bảo mật như mã hóa dữ liệu khi lưu trữ và truyền tải, cũng như quản lý quyền truy cập chi tiết để đảm bảo an toàn cho dữ liệu.

* Ứng dụng thực tế:



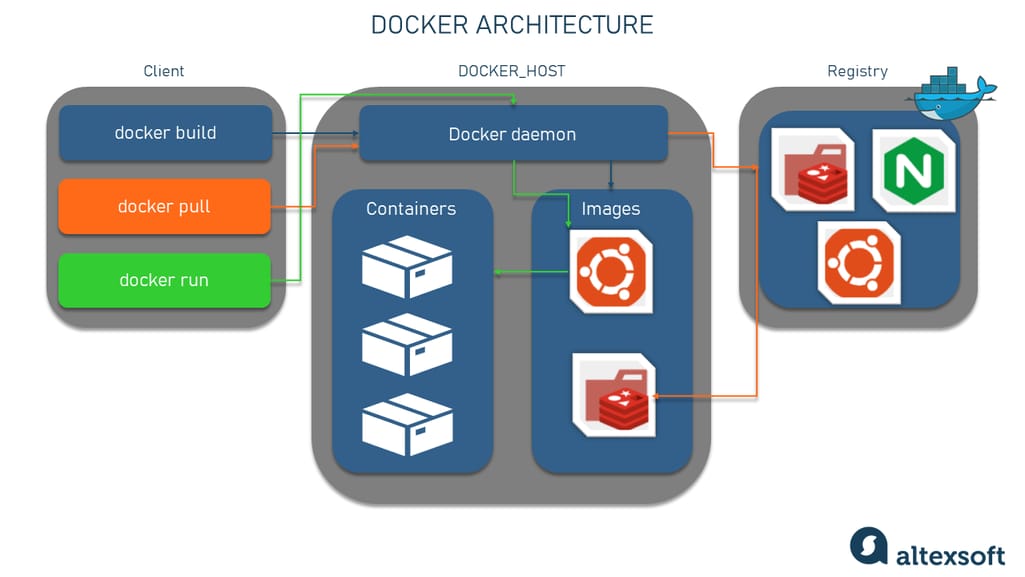
Hình 3.18 Ứng dụng của MinIO

* Lưu trữ ảnh, video: MINIO có thể được sử dụng để lưu trữ các file hình ảnh, video, đặc biệt là đối với các ứng dụng web hoặc di động.
* Lưu trữ sao lưu: MINIO là lựa chọn lý tưởng để lưu trữ các bản sao lưu hệ thống, dữ liệu quan trọng của doanh nghiệp.
* MinIO được sử dụng để lưu trữ và quản lý khối lượng dữ liệu lớn, bao gồm dữ liệu thu thập từ các hệ thống IoT, cảm biến, hoặc thiết bị thông minh. Nó cũng hỗ trợ lưu trữ tập dữ liệu cho các ứng dụng học máy và AI, tích hợp với các hệ thống phân tích dữ liệu như Apache Spark, Hadoop, hoặc Presto.
* MinIO thường được sử dụng làm hệ thống sao lưu dữ liệu cho các tổ chức, bao gồm sao lưu dữ liệu từ các máy chủ, cơ sở dữ liệu, và ứng dụng. Nó hỗ trợ phục hồi dữ liệu sau các sự cố mất mát hoặc hỏng hóc hệ thống và tích hợp với các giải pháp sao lưu như Veeam, Commvault, hoặc Restic.
* MinIO được dùng để tạo ra các giải pháp lưu trữ đám mây nội bộ hoặc tại chỗ, thay thế các dịch vụ đám mây công cộng như Amazon S3. Nó cung cấp nền tảng lưu trữ tập trung cho doanh nghiệp và quyền truy cập dữ liệu an toàn cho các nhóm làm việc phân tán.
  + 1. **Docker**



Hình 3.19 Docker

* Docker là một nền tảng mã nguồn mở tự động hóa việc triển khai, mở rộng và quản lý các ứng dụng trong các container nhẹ, di động. Các container này đóng gói một ứng dụng và các phụ thuộc của nó, đảm bảo hiệu suất nhất quán trên nhiều môi trường khác nhau, từ máy cục bộ của nhà phát triển đến máy chủ sản xuất.
* Bằng cách sử dụng Docker, các nhà phát triển có thể hợp lý hóa vòng đời phát triển, cho phép xây dựng, thử nghiệm và triển khai ứng dụng nhanh chóng. Cách tiếp cận này nâng cao hiệu quả và độ tin cậy, vì các ứng dụng chạy nhất quán bất kể cơ sở hạ tầng cơ bản.
* Docker cung cấp khả năng đóng gói và chạy một ứng dụng trong một cách ly lỏng lẻo môi trường được gọi là container. Sự cô lập và bảo mật cho phép chạy nhiều các vùng chứa đồng thời trên một máy chủ nhất định. Hộp đựng có trọng lượng nhẹ và chứa mọi thứ cần thiết để chạy ứng dụng, vì vậy không cần phải dựa vào những gì được cài đặt trên máy chủ. Chúng ta có thể chia sẻ vùng chứa trong khi làm việc, và đảm bảo rằng mọi người được chia sẻ đều nhận được cùng một vùng chứa hoạt động trong theo cách tương tự.
* Các thành phần chính:



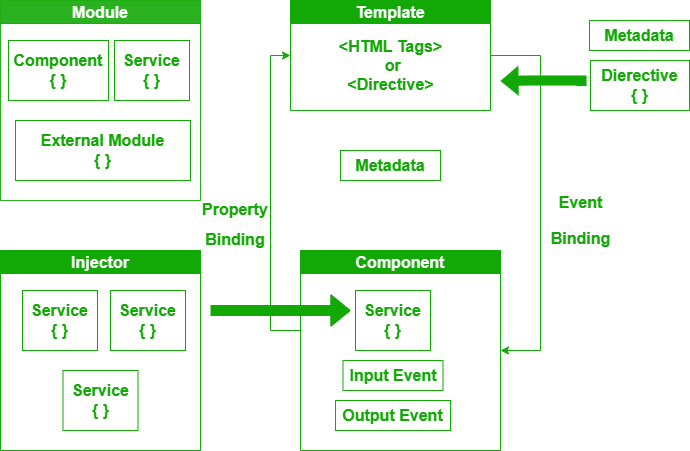
Hình 3.20 Kiến trúc Docker

* Docker Daemon: Thành phần cốt lõi quản lý các container Docker và xử lý các đối tượng container. Thành phần này lắng nghe các yêu cầu được gửi qua API Docker Engine và thực hiện các hành động như xây dựng, chạy và phân phối container.
* Docker Client: Giao diện dòng lệnh (CLI) cho phép người dùng tương tác với Docker Daemon. Giao diện này gửi lệnh đến Daemon và nhận phản hồi, cho phép người dùng quản lý container, hình ảnh và mạng.
* Docker Images: Được sử dụng để xây dựng các container Docker. Image được sử dụng để lưu trữ và vận chuyển các ứng dụng và có thể được lưu trữ trong Docker registry.
* Docker Containers: Môi trường được chuẩn hóa, đóng gói chạy ứng dụng. Các container được quản lý bằng Docker API hoặc CLI và được tạo từ hình ảnh Docker.
* Docker Registries: Kho lưu trữ để lưu trữ hình ảnh Docker. Máy khách Docker kết nối với các registries để tải xuống ("pull") hình ảnh để sử dụng hoặc tải lên ("push") image mà chúng đã xây dựng. Registries công khai chính là Docker Hub, nhưng registries cũng có thể là riêng tư.
* Docker Networks: virtual network được sử dụng để kết nối nhiều container. Chúng cho phép các container giao tiếp với nhau và với hệ thống máy chủ, tạo điều kiện thuận lợi cho giao tiếp giữa các container và cô lập mạng.
* Docker Volumes: Được sử dụng để lưu trữ dữ liệu được tạo ra và sử dụng bởi các container Docker. Các volume được lưu trữ bên ngoài hệ thống tệp container, cho phép dữ liệu lưu trữ ngay cả sau khi container bị xóa hoặc tạo lại.
* Docker Compose: Một công cụ để xác định và chạy các ứng dụng Docker đa container. Công cụ này sử dụng các tệp YAML để cấu hình các dịch vụ của ứng dụng và thực hiện quá trình tạo và khởi động tất cả các container chỉ bằng một lệnh.
* Docker Swarm: Cung cấp chức năng cụm gốc cho các container Docker, biến một nhóm các công cụ Docker thành một công cụ Docker ảo duy nhất. Chế độ Swarm được tích hợp với Docker Engine và cho phép sắp xếp các dịch vụ trên nhiều máy chủ Docker.
* Lợi ích của Docker:
* Tính di động: Các container Docker đóng gói các ứng dụng và các phụ thuộc của chúng, giúp dễ dàng di chuyển các ứng dụng giữa các môi trường khác nhau một cách liền mạch. Cho dù làm việc trên máy trạm cục bộ hay trên máy chủ sản xuất trên đám mây, container Docker sẽ chạy theo cùng một cách, đảm bảo tính nhất quán trong tất cả các bước của vòng đời ứng dụng.
* Tính cách ly: Các container Docker cung cấp mức độ cô lập cao giữa các ứng dụng và các phụ thuộc của chúng. Mỗi container chạy độc lập, nghĩa là nó sẽ không can thiệp vào các container khác trên cùng một máy hoặc máy chủ. Điều này rất quan trọng để đảm bảo tính bảo mật và tính ổn định của các ứng dụng trong môi trường dùng chung.
* Sử dụng tài nguyên hiệu quả: Docker sử dụng ít tài nguyên hơn so với máy ảo truyền thống. Vì container chia sẻ cùng một nhân hệ điều hành cơ bản nên chúng nhẹ hơn nhiều và cần ít dung lượng đĩa và RAM hơn. Điều này cho phép nhiều ứng dụng chạy trên một máy duy nhất, giúp tiết kiệm chi phí phần cứng và dễ dàng quản lý tài nguyên.
* Các trường hợp sử dụng Docker:
* Microservices: Nhờ có docker, các ứng dụng có thể được chia thành các thành phần nhỏ hơn, dễ quản lý hơn. Việc triển khai và phát triển các kiến ​​trúc dựa trên microservices được tạo điều kiện thuận lợi bởi docker.
* Containers: Việc quản lý và tạo các container nhẹ, riêng biệt được hỗ trợ bởi docker. Các container này đóng gói các phụ thuộc của ứng dụng và đảm bảo rằng ứng dụng đó nhất quán trên nhiều môi trường.
* DevOps Adoption: Quá trình phân phối phần mềm được Docker hợp lý hóa vì nó giúp các tổ chức áp dụng các nguyên tắc DevOps. Nó được thực hiện bằng cách cho phép cộng tác giữa các nhóm vận hành và phát triển.
* Software Testing: Docker cho phép nhà phát triển kiểm tra ứng dụng của họ trong các container riêng biệt, có thể tái tạo. Quá trình quản lý và thiết lập môi trường kiểm tra được Docker đơn giản hóa.
* Continuous Deployment: Người ta có thể dễ dàng đạt được việc phân phối và tích hợp liên tục nhờ docker. Điều này là do docker cho phép triển khai ứng dụng tự động và nhanh chóng.
* Legacy App Migration: Việc di chuyển các ứng dụng cũ sang môi trường chứa được hỗ trợ bởi docker. Điều này cải thiện khả năng mở rộng, tính di động và dễ quản lý các ứng dụng cũ.
* Hạn chế của Docker:
* Security: Vì trong hầu hết các trường hợp, các container đều sử dụng một hệ điều hành chung nên có nguy cơ chung là nhiều container sẽ bị xâm phạm cùng lúc nếu hệ thống máy chủ bị tấn công.
* Đường cong học tập khó: Những cảnh báo dành riêng cho hệ điều hành, các bản cập nhật thường xuyên và các sắc thái khác khiến việc thành thạo Docker trở nên khó khăn. Ngoài ra, mặc dù tính năng Docker Extensions có lợi theo nhiều cách, nhưng nó đòi hỏi phải có thêm nhiều cấp độ hiểu biết về các công cụ của bên thứ ba.
* Hỗ trợ giao diện hạn chế: Docker chủ yếu được thiết kế cho các ứng dụng máy chủ không có giao diện người dùng đồ họa. Có thể chạy các ứng dụng dựa trên GUI trong các container nhưng thường cồng kềnh và không đơn giản
  + 1. **Angular**



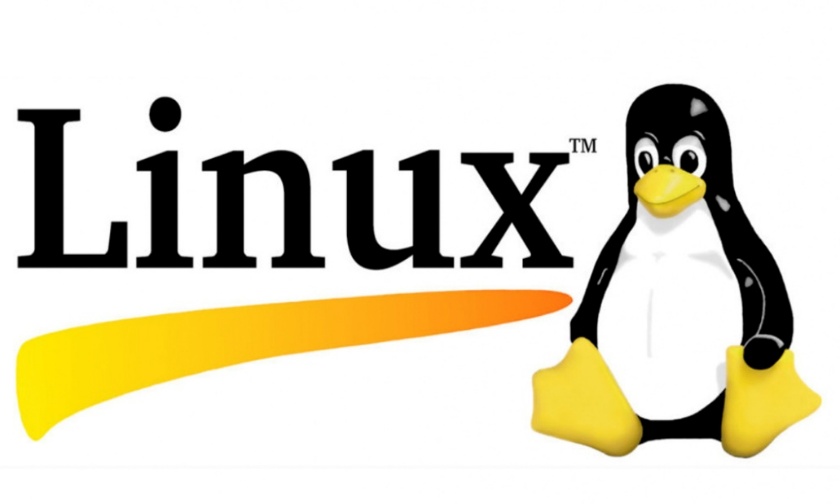
Hình 3.21 Angular

* Angular là một framework JavaScript mã nguồn mở được viết bằng TypeScript. Google duy trì framework này và mục đích chính của nó là phát triển các ứng dụng SPA. Là một framework, Angular có những lợi thế rõ ràng đồng thời cũng cung cấp một cấu trúc chuẩn để các nhà phát triển làm việc. Nó cho phép người dùng tạo các ứng dụng lớn theo cách có thể bảo trì được.
* Kiến trúc, thành phần:



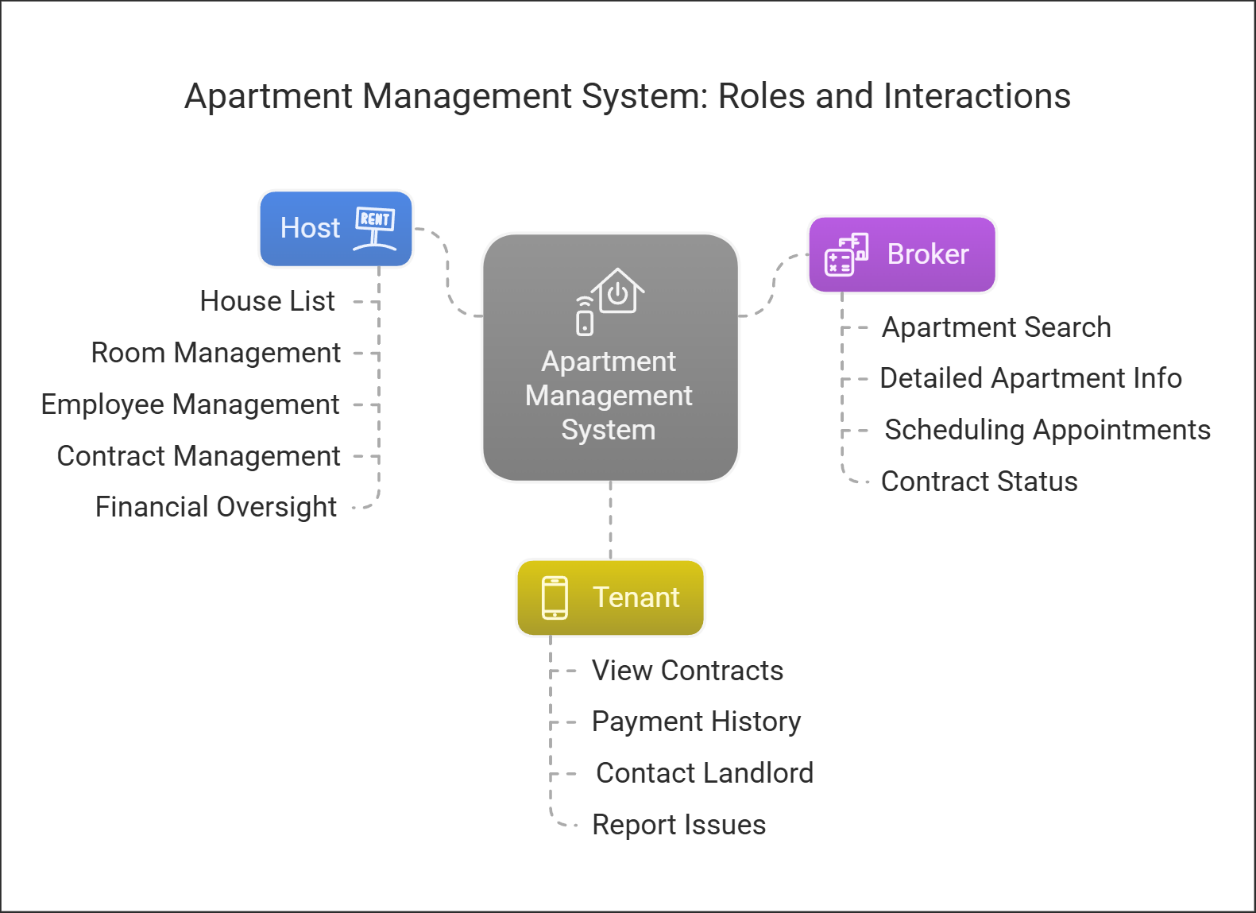
Hình 3.22 Kiến trúc, thành phần của Angular

* Modules:
* Module là một đơn vị bao gồm một khối riêng biệt để thực hiện chức năng cụ thể và cho phép chúng ta chia nhỏ ứng dụng thành các phần nhỏ hơn.
* Trong một module, chúng ta có thể xuất và nhập các thành phần và dịch vụ từ các module khác.
* Các module được tạo bằng cách sử dụng decorator @NgModule .
* Tuy nhiên với phiên bản Standalone thì cấu hình module đã bị loại bỏ, có thể không còn file \*.module.ts trong các phiên bản mới.
* Components:
* Thành phần là khối xây dựng nên ứng dụng Angular.
* Một thành phần bao gồm một mẫu (giao diện người dùng HTML), kiểu (giao diện/thiết kế CSS) và một lớp TypeScript chứa logic nghiệp vụ.
* Để chỉ ra một lớp là thành phần, sử dụng trình trang trí @Component .
* Trình trang trí @Component cung cấp siêu dữ liệu cho thành phần.
* Thuộc tính siêu dữ liệu thành phần bao gồm selector, directive, providers, styleUrl và templateUrl.
* Templates:
* Giao diện người dùng hoặc chế độ xem của người dùng cuối được xác định bằng cách sử dụng template.
* Các template được tạo bằng HTML và nó liên kết các thuộc tính và phương thức, do đó giúp chúng ta hiển thị dữ liệu một cách linh hoạt.
* Cú pháp mẫu bao gồm directives, interpolation, built-in directives, template expression operators, property binding, và event binding để tạo chế độ xem động và tương tác.
* Directives:
* Directive là các hướng dẫn trong DOM (Document Object Model).
* Các Directive được sử dụng trong các mẫu để tùy chỉnh hành vi của các phần tử.
* Angular cung cấp các chỉ thị tích hợp như @if, @for cũng như các chỉ thị tùy chỉnh do các nhà phát triển tạo ra.
* Services: Service là một lớp đóng gói logic, dữ liệu hoặc chức năng được chia sẻ, có thể được nhiều thành phần sử dụng trong toàn bộ ứng dụng. Dịch vụ thúc đẩy tính module và khả năng tái sử dụng mã, cho phép các thành phần phân công các tác vụ như truy xuất dữ liệu, xác thực đầu vào hoặc ghi nhật ký cho các đơn vị tập trung này.
* Dependency Injection(DI): Hệ thống DI của Angular hỗ trợ việc đưa các dịch vụ vào các thành phần hoặc dịch vụ khác, quản lý việc tạo và vòng đời của các phụ thuộc này.
* Router: Router là dịch vụ cốt lõi giúp điều hướng giữa các chế độ xem hoặc thành phần khác nhau trong ứng dụng một trang (SPA). Nó diễn giải URL của trình duyệt thành hướng dẫn để tải các thành phần cụ thể, cho phép chuyển đổi liền mạch và trải nghiệm người dùng động mà không cần tải lại toàn bộ trang.
* Tính năng của Angular:
* Custom Component: Angular cho phép người dùng xây dựng các thành phần có thể đóng gói chức năng cùng với logic kết xuất thành các phần có thể tái sử dụng.
* Data Binding: Angular cung cấp khả năng đồng bộ hóa liền mạch giữa mô hình và chế độ xem, cho phép quản lý dễ dàng dữ liệu đầu vào của người dùng.
* Dependency Injection: Hệ thống dependency injection tích hợp sẵn của Angular cung cấp khả năng phát triển theo mô-đun và khả năng tái sử dụng mã.
* Directives: Angular cung cấp một bộ chỉ thị tích hợp phong phú để thao tác DOM, chẳng hạn như @for, @if,….
* Routing: Mô-đun định tuyến mạnh mẽ của Angular cho phép xây dựng các SPA với nhiều chế độ xem và điều hướng giữa chúng.
* HTTP Client: Angular bao gồm một mô-đun HTTP Client để thực hiện các request đến server, đơn giản hóa việc truy xuất và xử lý dữ liệu.
* Testing: Các bài kiểm tra là công cụ hạng nhất và Angular được xây dựng từ đầu với mục tiêu là khả năng kiểm tra. Chúng ta sẽ có khả năng kiểm tra mọi phần của ứng dụng, điều này được khuyến khích rất nhiều.
* Browser Compatibility: Angular hoạt động đa nền tảng và tương thích với nhiều trình duyệt. Một ứng dụng Angular thường có thể chạy trên tất cả các trình duyệt (ví dụ: Chrome, Firefox) và hệ điều hành, chẳng hạn như Windows, macOS và Linux.
* Ưu điểm:
* Năng suất: Hệ sinh thái và công cụ mở rộng của Angular giúp hợp lý hóa các tác vụ phát triển, giúp hoàn thành dự án nhanh hơn.
* Khả năng bảo trì: Kiến trúc mô-đun của Angular và sự phân tách rõ ràng các mối quan tâm thúc đẩy khả năng tổ chức mã và bảo trì.
* Khả năng mở rộng: Angular rất phù hợp để xây dựng các ứng dụng quy mô lớn nhờ kiến ​​trúc dựa trên thành phần và hiệu suất mạnh mẽ.
* Hỗ trợ cộng đồng: Được Google và cộng đồng các nhà phát triển đông đảo hỗ trợ, Angular có được sự hỗ trợ mạnh mẽ của cộng đồng và cải tiến liên tục.
* Nhượt điểm:
* Đường cong học tập: Angular có đường cong học tập khá dốc, đặc biệt là đối với người mới bắt đầu, do các khái niệm phức tạp và tài liệu hướng dẫn chi tiết.
* Chi phí hiệu suất: Các tính năng mạnh mẽ của Angular đi kèm với chi phí hiệu suất và các ứng dụng được tối ưu hóa kém có thể gặp phải các vấn đề về hiệu suất.
* Kích thước: Các ứng dụng Angular thường có kích thước tệp lớn hơn so với các nền tảng khác, điều này có thể ảnh hưởng đến thời gian tải, đặc biệt là trên thiết bị di động.
* Di chuyển: Việc nâng cấp giữa các phiên bản Angular chính có thể khó khăn và tốn thời gian, đòi hỏi phải thay đổi đáng kể cơ sở mã hiện có.
* Ứng dụng thực tế:
* Hệ thống quản lý nội dung (CMS): Khả năng xử lý dữ liệu động và kiến ​​trúc mô-đun của Angular khiến nó phù hợp để xây dựng nền tảng CMS, cho phép tạo và quản lý nội dung hiệu quả.
* Ứng dụng dữ liệu thời gian thực: Các ứng dụng yêu cầu cập nhật dữ liệu thời gian thực, chẳng hạn như tỷ số thể thao trực tiếp hoặc trình theo dõi thị trường chứng khoán, được hưởng lợi từ cấu trúc dựa trên thành phần và liên kết dữ liệu hiệu quả của Angular.
* Ứng dụng web lũy tiến (PWA): Angular hỗ trợ phát triển PWA, cung cấp trải nghiệm giống như ứng dụng gốc trên web, bao gồm quyền truy cập ngoại tuyến và thông báo đẩy.
* Hệ thống hoạch định nguồn lực doanh nghiệp (ERP): Khả năng mở rộng và bảo trì của Angular khiến nó trở nên lý tưởng để xây dựng các hệ thống ERP phức tạp tích hợp nhiều quy trình kinh doanh khác nhau.
* Nền tảng thương mại điện tử: Angular được sử dụng để phát triển nền tảng thương mại điện tử, cung cấp giao diện người dùng năng động và trải nghiệm mua sắm liền mạch.
* Tích hợp phương tiện truyền thông xã hội: Khả năng của Angular được tận dụng trên các nền tảng truyền thông xã hội để xây dựng các tính năng tương tác và cập nhật theo thời gian thực.
  + 1. **Linux**



Hình 3.23 Linux

* Linux là một hệ điều hành mã nguồn mở, được phát triển dựa trên hệ điều hành Unix và viết bằng ngôn ngữ lập trình C. Được giới thiệu lần đầu tiên vào năm 1991 bởi Linus Torvalds, Linux đã nhanh chóng trở thành một trong những hệ điều hành phổ biến trên thế giới.
* Kiến trúc:
* Kernel (Nhân): Phần cốt lõi của hệ điều hành, quản lý tài nguyên phần cứng và cho phép phần mềm tương tác với phần cứng.
* Shell: Giao diện dòng lệnh cho phép người dùng tương tác với hệ thống thông qua các lệnh.
* Hệ thống tệp (File System): Quản lý và tổ chức dữ liệu trên ổ đĩa.
* Ưu điểm:
* Miễn phí và mã nguồn mở: Người dùng có thể sử dụng, chỉnh sửa và phân phối lại mà không phải trả phí bản quyền.
* Bảo mật cao: Linux được thiết kế với cấu trúc bảo mật chặt chẽ, giảm thiểu nguy cơ bị tấn công bởi phần mềm độc hại.
* Linh hoạt và tùy chỉnh cao: Người dùng có thể tùy chỉnh hệ thống theo nhu cầu cá nhân, từ giao diện đến các chức năng hệ thống.
* Hiệu suất ổn định: Linux hoạt động mượt mà trên cả những máy tính có cấu hình thấp.
* Nhượt điểm:
* Hỗ trợ ứng dụng hạn chế: Nhiều phần mềm phổ biến không có phiên bản dành cho Linux, đặc biệt là các ứng dụng chuyên dụng hoặc trò chơi. Điều này có thể gây khó khăn cho người dùng khi chuyển từ hệ điều hành khác sang Linux.
* Khả năng tương thích phần cứng: Một số nhà sản xuất không cung cấp driver cho Linux, dẫn đến việc một số thiết bị phần cứng có thể không hoạt động hoặc hoạt động không ổn định trên hệ điều hành này.
* Thời gian làm quen: Người dùng mới chuyển từ các hệ điều hành khác có thể cần thời gian để làm quen với giao diện và cách thức hoạt động của Linux.
* Hỗ trợ kỹ thuật hạn chế: Không phải tất cả các nhà sản xuất phần cứng và phần mềm đều cung cấp hỗ trợ cho Linux, do đó người dùng có thể gặp khó khăn khi tìm kiếm sự trợ giúp chính thức.
* Các bản phân phối:
* Ubuntu: Thân thiện với người dùng, phù hợp cho cả người mới bắt đầu và chuyên gia.
* Fedora: Luôn cập nhật các công nghệ mới nhất, được cộng đồng hỗ trợ mạnh mẽ.
* Debian: Nổi tiếng với tính ổn định và bảo mật cao, là nền tảng cho nhiều bản phân phối khác.
* CentOS: Phiên bản miễn phí của Red Hat Enterprise Linux, thường được sử dụng cho máy chủ.
* Ứng dụng của Linux trong thực tế:
* Máy chủ (Server): Linux được sử dụng rộng rãi trong các máy chủ web, cơ sở dữ liệu và email nhờ tính ổn định và bảo mật.
* Máy tính cá nhân: Nhiều người dùng lựa chọn Linux cho máy tính cá nhân để trải nghiệm sự linh hoạt và hiệu suất cao.
* Thiết bị nhúng: Linux được tích hợp trong nhiều thiết bị như điện thoại thông minh, TV thông minh và các thiết bị IoT.
* Siêu máy tính: Phần lớn các siêu máy tính trên thế giới chạy Linux do khả năng tùy chỉnh và hiệu suất vượt trội.
  1. **Các chức năng chính**



Hình 3.24 Các chứng năng chính của ứng dụng

Ứng dụng được chia thành ba module chính: Host, Tenant, và Broker, mỗi module phục vụ một nhóm người dùng cụ thể với các chức năng riêng biệt:

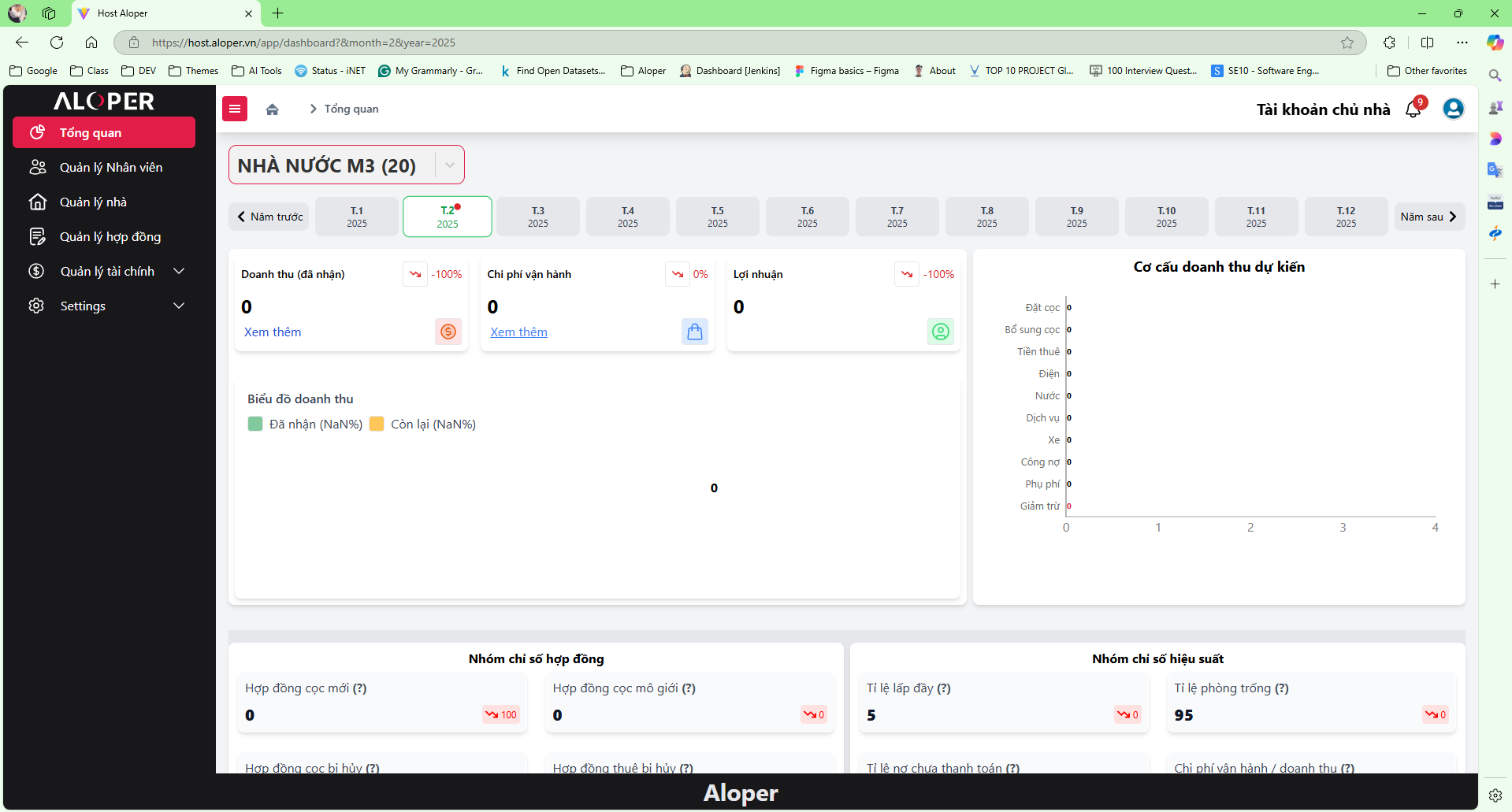
* Host: Đây là ứng dụng dành cho chủ nhà, giúp họ quản lý danh sách bất động sản của mình. Chủ nhà có thể theo dõi và quản lý từng căn hộ, bao gồm các phòng, nội thất, dịch vụ đi kèm, cũng như nhân viên vận hành. Ngoài ra, họ có thể quản lý hợp đồng đặt cọc, hợp đồng thuê, theo dõi tài chính và dòng tiền ra vào của từng căn hộ.
* Broker: Ứng dụng dành cho môi giới, giúp họ tìm kiếm căn hộ dựa trên bộ lọc mong muốn. Môi giới có thể xem chi tiết từng căn hộ, bao gồm thông tin về dịch vụ, nội thất và tình trạng phòng (trống, sắp trống, đã đặt cọc). Họ cũng có thể xem tổng quan tòa nhà, đặt lịch hẹn với chủ nhà, lập hợp đồng đặt cọc và theo dõi trạng thái của các hợp đồng cũng như danh sách lịch hẹn.
* Tenant: Ứng dụng mobile dành cho khách thuê, cho phép họ xem hợp đồng đặt cọc, hợp đồng thuê, cũng như lịch sử thanh toán hàng tháng. Người thuê cũng có thể liên hệ trực tiếp với chủ nhà hoặc quản trị viên để phản ánh sự cố, khiếu nại hoặc nhận hỗ trợ khi cần.

Ứng dụng giúp tối ưu hóa quy trình quản lý, kết nối hiệu quả giữa các bên, mang lại trải nghiệm thuận tiện và minh bạch trong lĩnh vực cho thuê căn hộ.

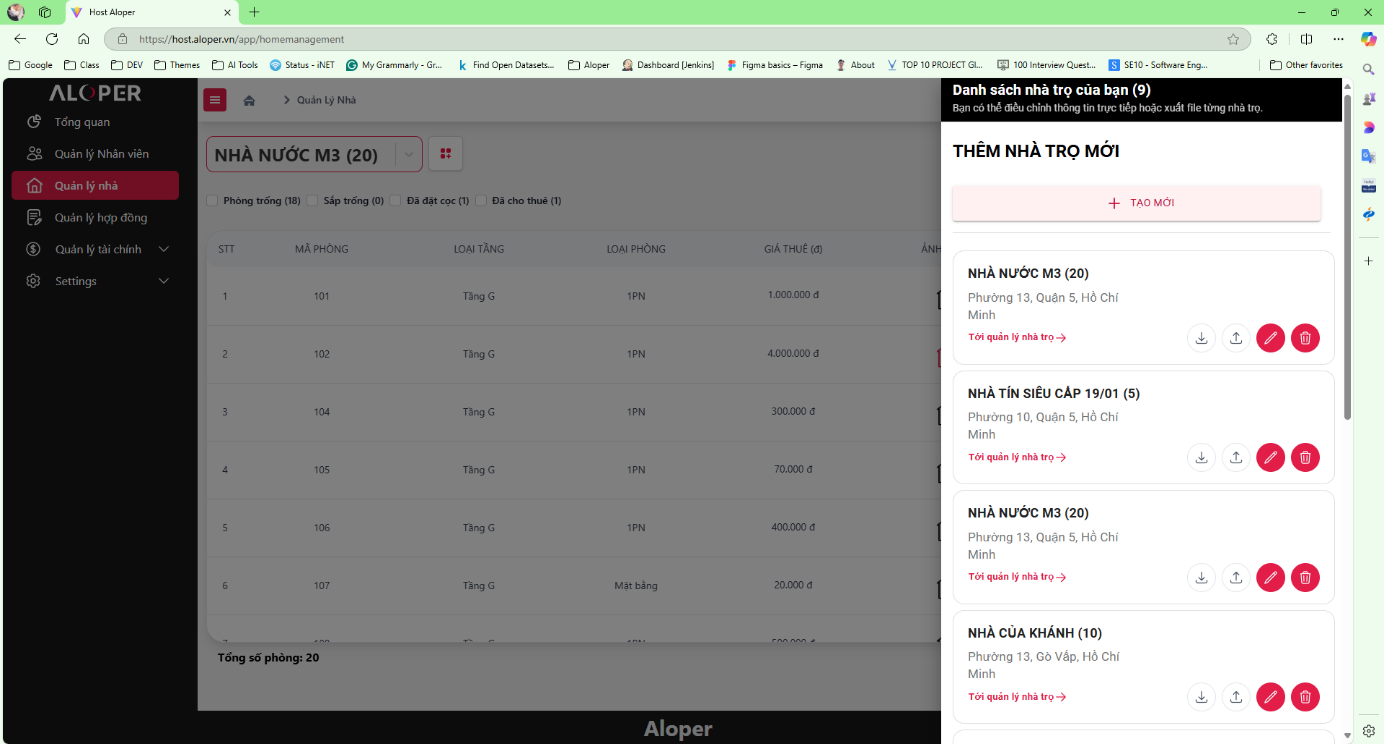
* 1. **Các nội dung học tập, kết quả đạt được tại công ty**
     1. **Môi trường thực tế**
* Hàng ngày, điểm danh khi vừa tới văn phòng.
* Buổi sáng, buổi trưa lập kế hoạch công việc trong ngày, cuối ngày báo cáo tiến độ công việc đồng thời đẩy code mới nhất lên Github, deploy chức năng lên máy chủ nếu có.
* Cuối tuần, các chức năng được kiểm thử trên môi trường production và thông báo hoàn thành hoặc có lỗi thông qua công cụ quản lý công việc Lark.
* Đầu tuần, review lại công việc của tuần trước, bày tỏ những khó khăn thách thức, lập kế hoạch cho tuần mới.
* Tiếp nhận chức năng mới:
* Tiếp nhận yêu cầu: Trao đổi với khách hàng hoặc Product Owner để hiểu rõ tính năng cần phát triển.
* Lập kế hoạch: Chia nhỏ nhiệm vụ và ước tính thời gian thực hiện.
* Phát triển: Viết mã nguồn, thực hiện unit test, đảm bảo code đạt tiêu chuẩn.
* Kiểm thử: Tester kiểm tra lỗi, báo cáo và gửi lại cho developer chỉnh sửa.
* Triển khai: Đưa sản phẩm lên môi trường staging hoặc production.
* Môi trường làm việc tại công ty thân thiện, khuyến khích học hỏi và sáng tạo. Có cơ hội tham gia vào các buổi code review, trao đổi với mentor để nâng cao kỹ năng lập trình và hiểu rõ hơn về cách làm việc chuyên nghiệp.
  + 1. **Về nguyên tắc làm việc**
* Tuân thủ quy trình làm việc:
* Lập kế hoạch, kiểm thử, triển khai.
* Tham gia đầy đủ các buổi họp nhóm cập nhật tiến độ, điều chỉnh công việc kịp thời.
* Nguyên tắc viết mã nguồn:
* Tuân thủ quy tắc đặt tên biến, hàm, class theo tiêu chuẩn của nhóm phát triển.
* Viết code dễ hiểu, dễ bảo trì, tránh lặp code và tuân thủ các nguyên tắc.
* Sử dụng hệ thống quản lý mã nguồn Git: commit rõ ràng, tạo nhánh theo quy ước, thường xuyên đẩy code lên repository.
* Kiểm thử và đảm bảo chất lượng phần mềm:
* Theo dõi và sửa lỗi (debug) kịp thời khi phát sinh trong quá trình phát triển.
* Review code cùng với các thành viên trong nhóm để đảm bảo chất lượng và cải thiện kỹ năng lập trình.
* Quản lý thời gian và công việc:
* Ước lượng thời gian thực hiện nhiệm vụ một cách chính xác và hoàn thành đúng hạn.
* Sử dụng công cụ quản lý công việc Lark để theo dõi tiến độ.
* Báo cáo công việc định kỳ với mentor hoặc trưởng nhóm để đảm bảo đúng định hướng.
* Giao tiếp và làm việc nhóm:
* Trao đổi rõ ràng, minh bạch khi gặp vấn đề, tránh giấu lỗi hoặc trì hoãn báo cáo lỗi.
* Hỗ trợ và chia sẻ kiến thức với đồng nghiệp để cùng phát triển.
* Giữ thái độ tôn trọng, chuyên nghiệp khi làm việc với đội nhóm hoặc khách hàng.
* Bảo mật và bảo vệ dữ liệu:
* Không chia sẻ thông tin, mã nguồn hoặc dữ liệu của công ty ra bên ngoài.
* Sử dụng mật khẩu mạnh và các công cụ bảo mật để bảo vệ hệ thống.
* Tuân thủ các chính sách bảo mật dữ liệu theo quy định của công ty.

# CHƯƠNG 4: XÂY DỰNG ỨNG DỤNG

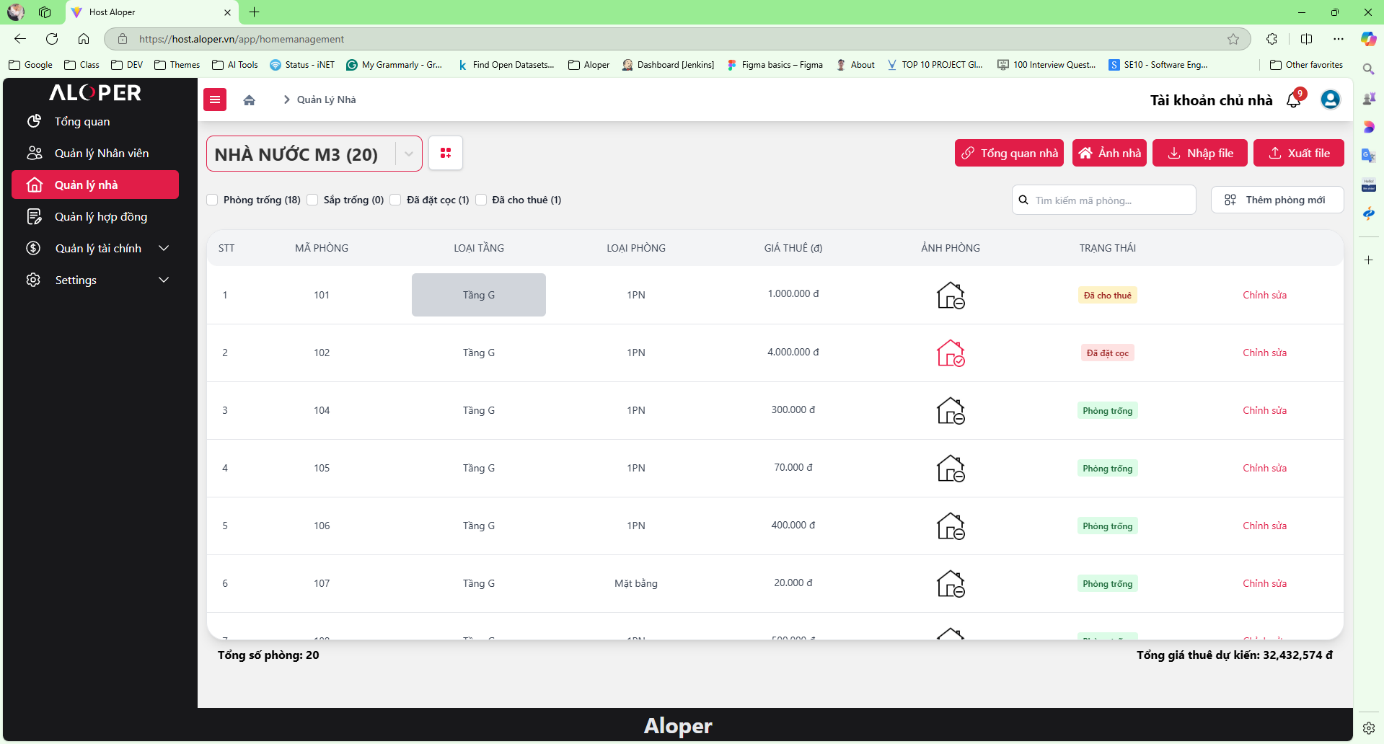
* 1. **Giao diện ứng dụng quản lý nhà**



Hình 4.1 Dashboard

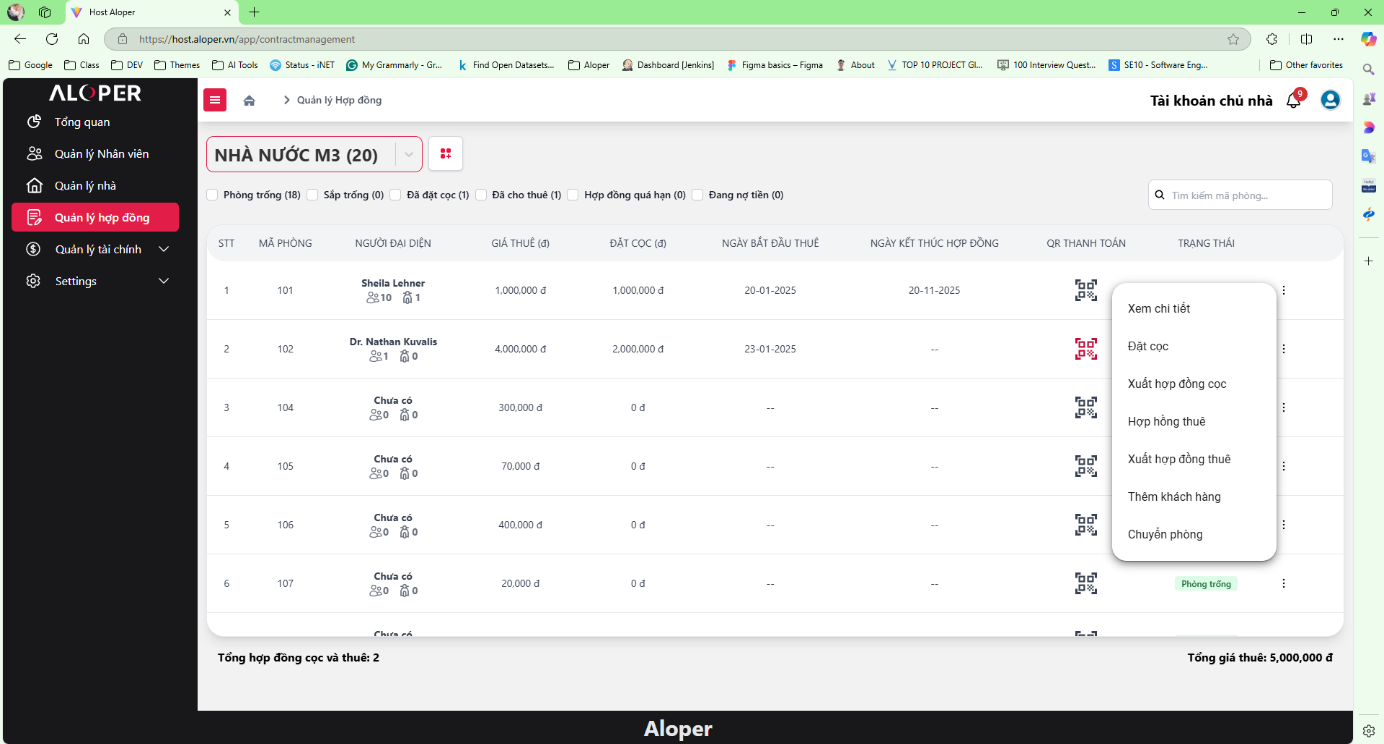


Hình 4.2 Danh sách nhà

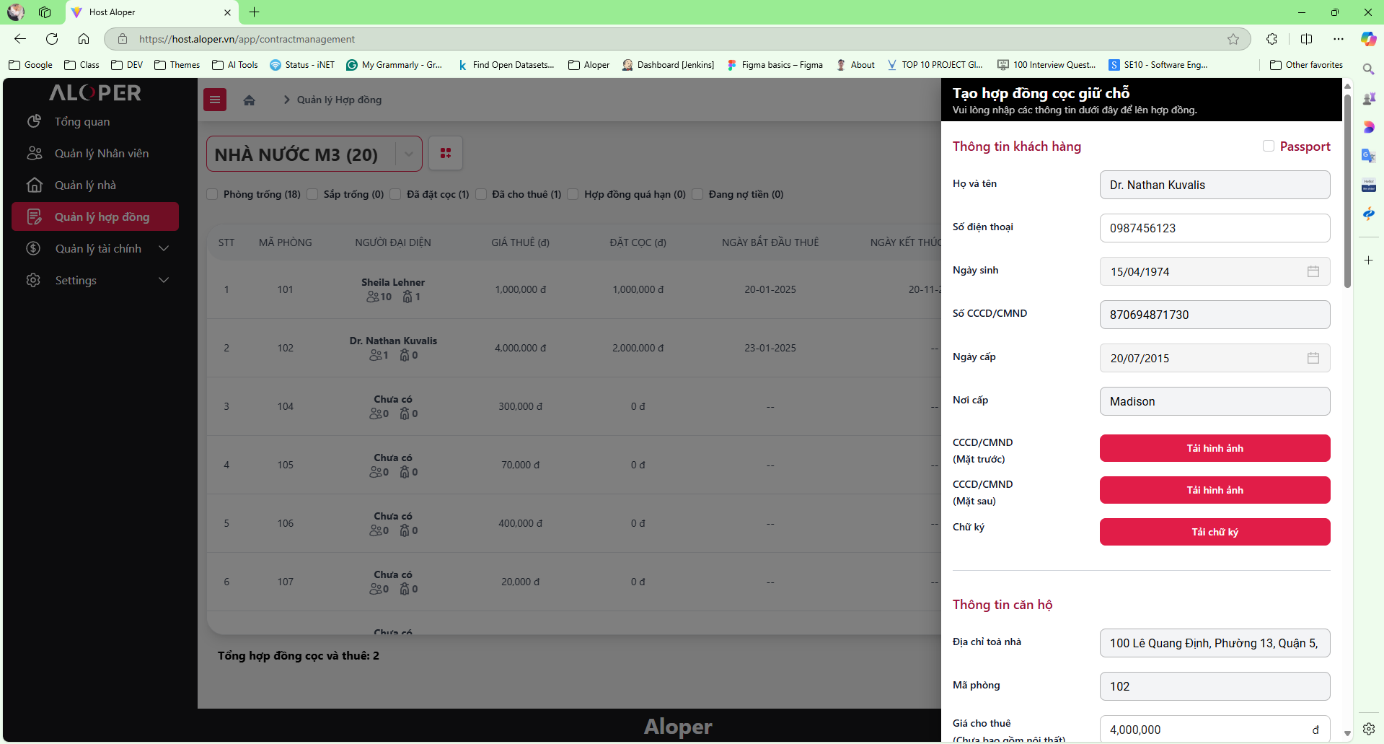


Hình 4.3 Danh sách phòng của một tòa nhà

* 1. **Giao diện quản lý hợp đồng**

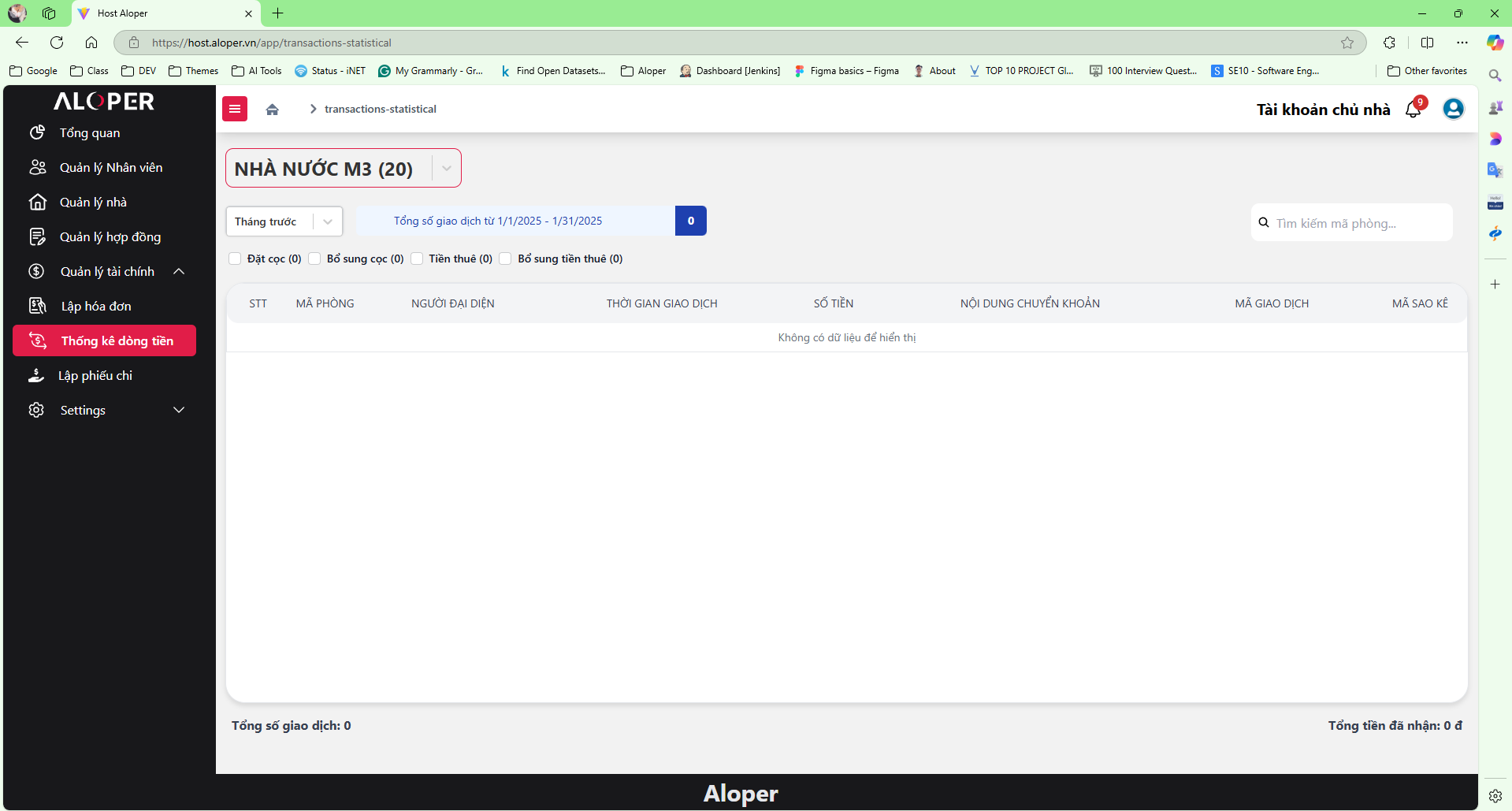


Hình 4.4 Danh sách hợp đồng (cọc và thuê)

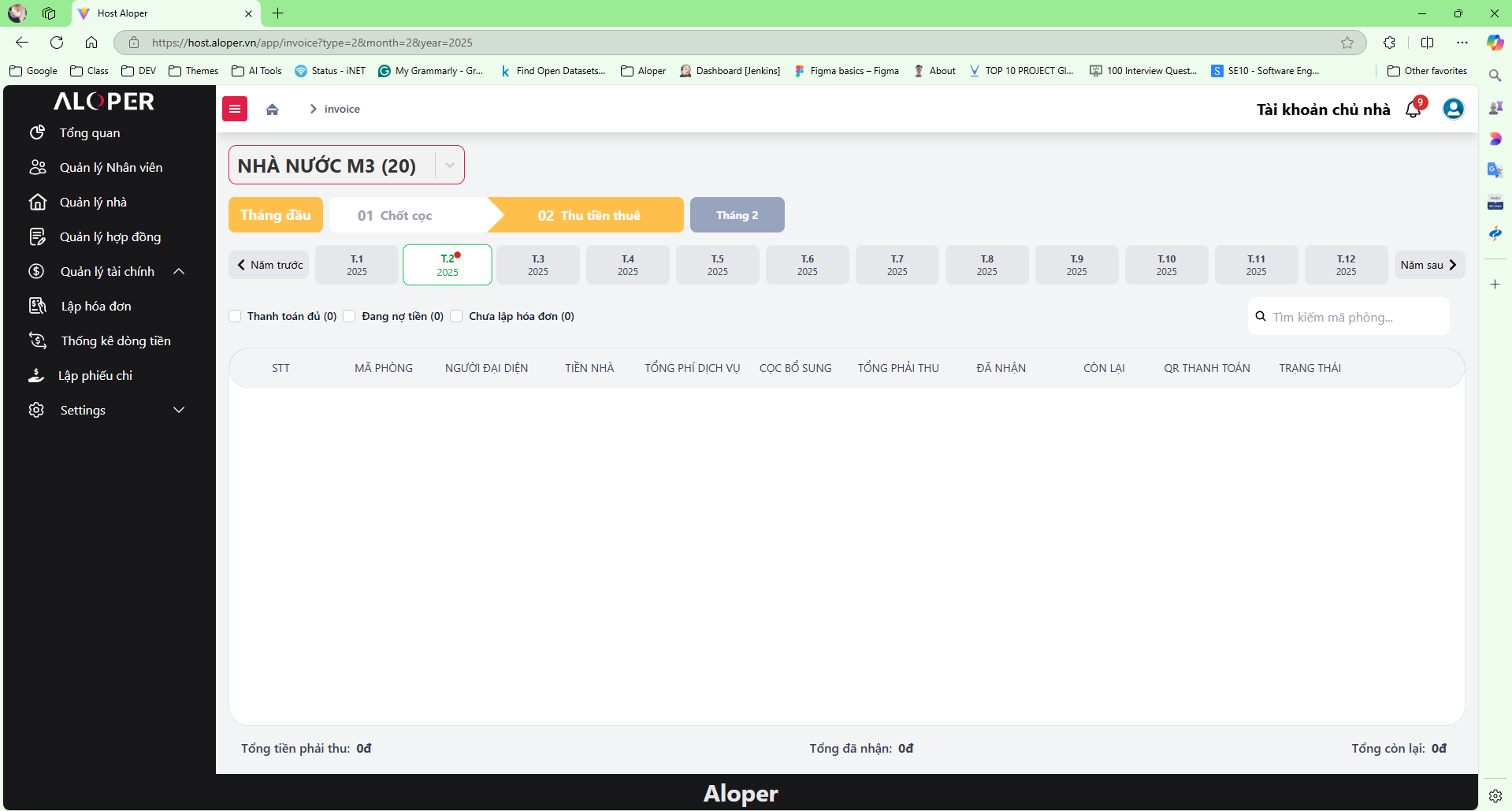


Hình 4.5 Cập nhật cọc

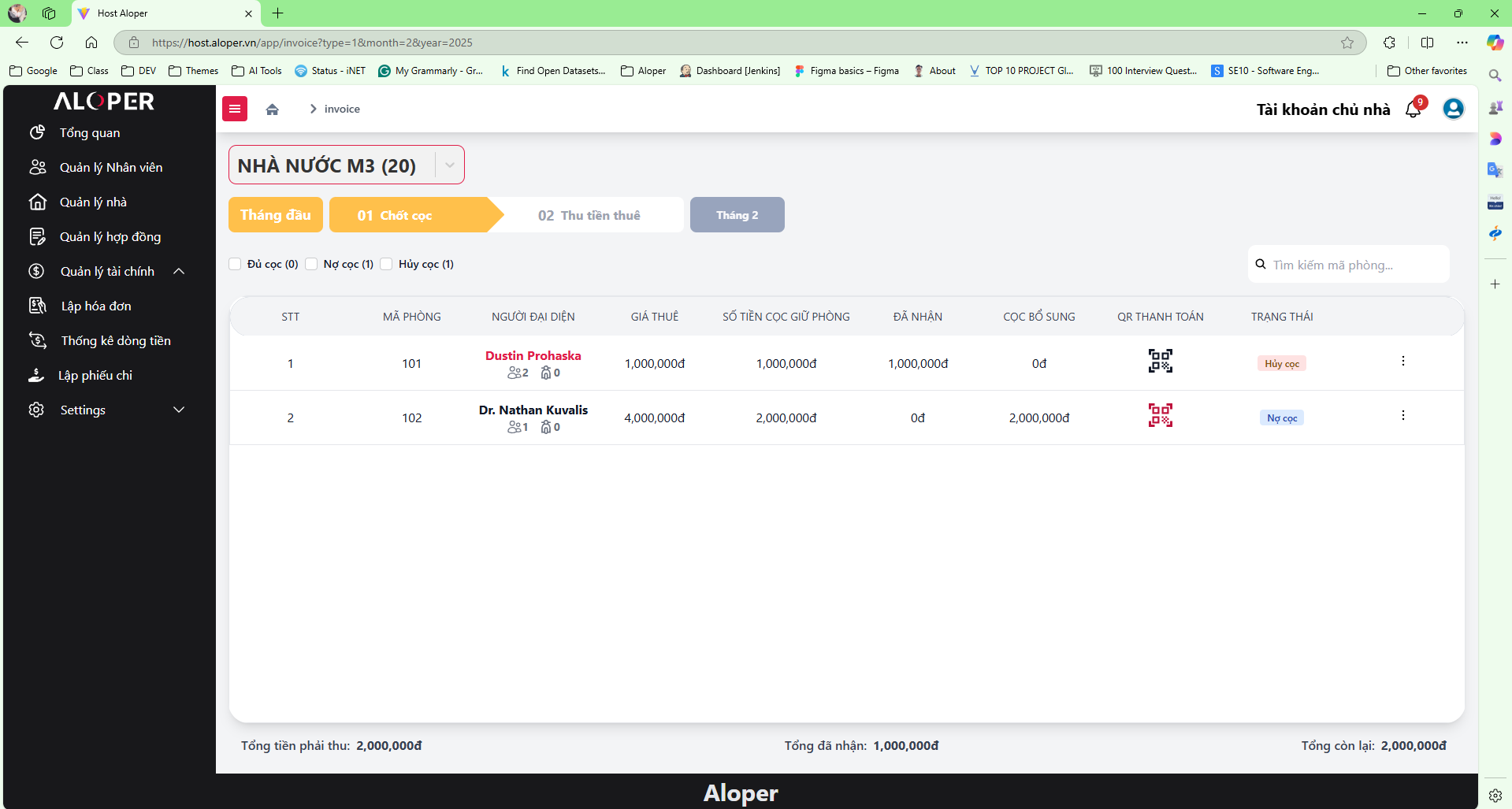
* 1. **Giao diện quản lý tài chính**



Hình 4.6 Thống kê dòng tiền

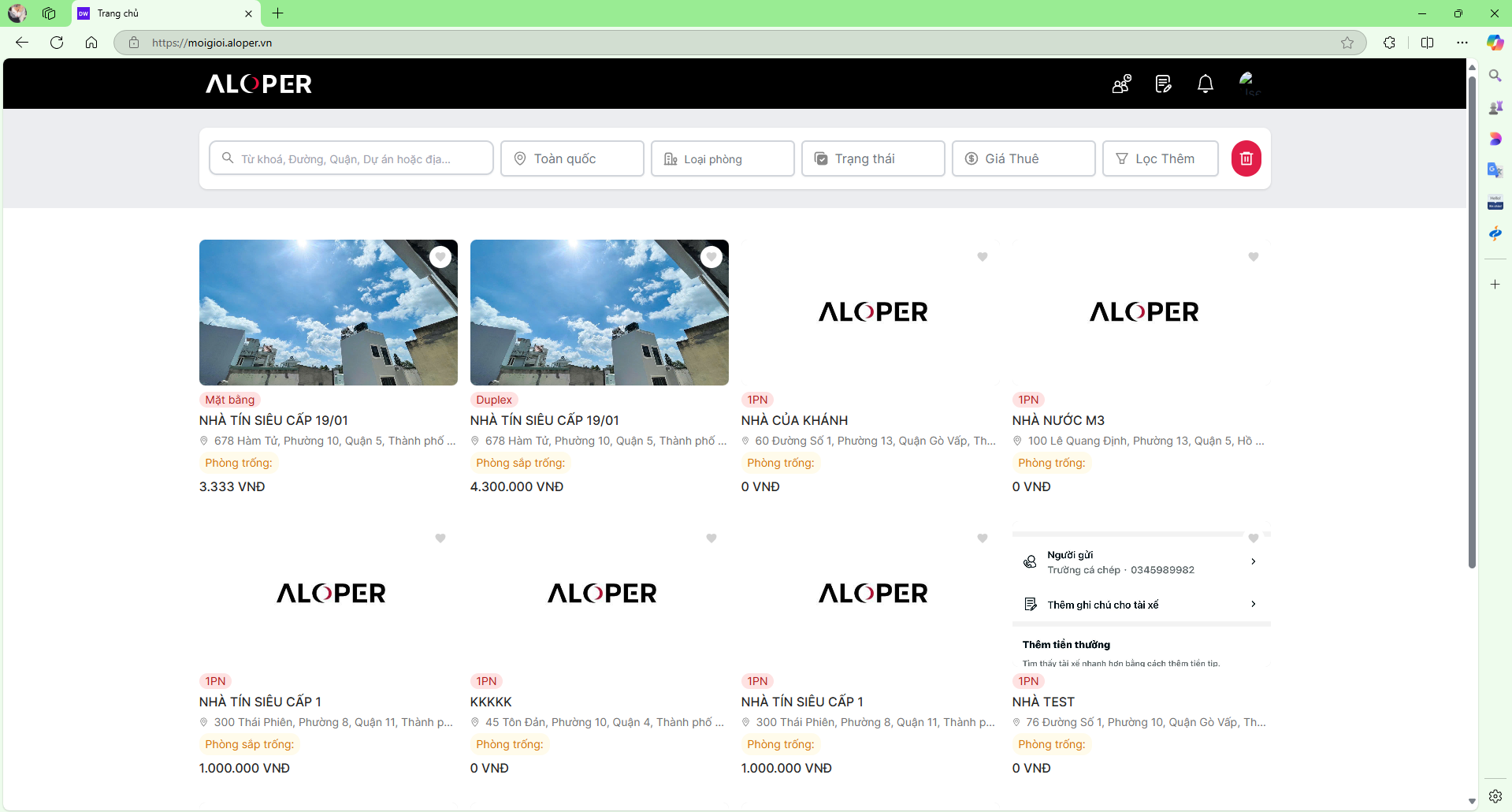


Hình 4.7 Thu tiền thuê tháng đầu



Hình 4.8 Chốt cọc

* 1. **Giao diện ứng dụng môi giới**



Hình 4.9 Danh sách căn hộ

# CHƯƠNG 5: KẾT LUẬN

* 1. **Kết luận**
* Sau thời gian thực tập tại công ty, em đã có cơ hội tiếp xúc và làm việc trong môi trường thực tế, từ đó tích lũy được nhiều kiến thức và kinh nghiệm quý báu về lập trình phần mềm. Em đã được rèn luyện kỹ năng lập trình, làm quen với các công cụ và công nghệ hiện đại, đồng thời hiểu rõ hơn về quy trình phát triển phần mềm trong doanh nghiệp.
* Ngoài kiến thức chuyên môn, em còn cải thiện được kỹ năng làm việc nhóm, tư duy logic, khả năng giải quyết vấn đề và quản lý thời gian. Những trải nghiệm này không chỉ giúp em nâng cao năng lực bản thân mà còn định hướng rõ hơn con đường phát triển nghề nghiệp trong tương lai.
  1. **Đóng góp, nhận xét từ phía công ty**
     1. **Đóng góp**
* Học tập và áp dụng các công nghệ, công cụ phát triển phần mềm theo yêu cầu của công ty.
* Tham gia vào dự án thực tế, hỗ trợ đội nhóm trong việc phát triển, kiểm thử và tối ưu mã nguồn.
* Đóng góp ý kiến và đề xuất giải pháp nhằm cải thiện hiệu suất và tính năng của sản phẩm.
* Tích cực học hỏi, rèn luyện kỹ năng làm việc nhóm và tuân thủ quy trình phát triển phần mềm của công ty.
  + 1. **Nhận xét từ phía công ty**
* Ưu điểm:
* Có tinh thần học hỏi, chủ động trong công việc.
* Nắm bắt nhanh các kiến thức mới và có khả năng áp dụng vào thực tế.
* Làm việc có trách nhiệm, hòa đồng với đồng nghiệp.
* Góp phần hỗ trợ nhóm trong một số công việc như kiểm thử, tối ưu mã nguồn và xử lý lỗi.
* Nhược điểm cần cải thiện:
* Cần nâng cao kỹ năng quản lý thời gian để tối ưu hiệu suất làm việc.
* Chưa thực sự tự tin trong việc đưa ra ý tưởng hoặc trình bày giải pháp.
* Cần rèn luyện thêm về tư duy thuật toán và tối ưu hóa mã nguồn.
  1. **Hướng phát triển của bản thân**
* Tìm hiểu sâu hơn các kỹ thuật nâng cao trong phát triển phần mềm.
* Học hỏi về kiến trúc, thiết kế hệ thống tối ưu, ứng dụng các công nghệ hiện đại vào quá trình phát triển, mở rộng hệ thống.
* Phát triển nhiều hơn kỹ năng giao tiếp, kỹ năng giải quyết vấn đề, phân tích bài toán.

# TÀI LIỆU THAM KHẨO

1. Editions and supported features of SQL Server 2022 - SQL Server | Microsoft Learn. Internet: <https://learn.microsoft.com/en-us/sql/sql-server/editions-and-components-of-sql-server-2022?view=sql-server-ver16>, 15/02/2025.
2. What is Microsoft SQL Server? A Comprehensive Guide. Internet: <https://www.theknowledgeacademy.com/blog/microsoft-sql-server/>, 15/02/2025.
3. Building E-Commerce Web APIs with ASP.NET Core: Flow, Structure, and Control | by Nitheesh Techie | Medium. Internet: <https://medium.com/@nitheesh.techie/building-e-commerce-web-apis-with-asp-net-core-flow-structure-and-control-a9c73f577b82>, 15/02/2025.
4. 25 Essential ASP.NET Core Features Every Developer Should Know. Internet: <https://umbracare.net/blog/aspnet-core-essential-features/>, 15/02/2025.
5. Top Benefits of ASP.NET Core for Modern Web Development. Internet: <https://thecompetenza.com/top-benefits-of-asp-net-core-for-modern-web-development/>, 15/02/2025.
6. Documentation | NestJS - A progressive Node.js framework. Internet: <https://docs.nestjs.com/>, 15/02/2025.
7. What is NestJS? – GeeksforGeeks. Internet: <https://www.geeksforgeeks.org/what-is-nestjs/>, 15/02/2025.
8. NestJS là gì? Tính năng hữu ích của NestJS đối với lập trình viên. Internet: <https://stringee.com/vi/blog/post/nestjs-la-gi>, 15/02/2025.
9. NestJS: Giải Pháp Toàn Diện Cho Ứng Dụng Server-Side | 200Lab Blog. Internet: <https://200lab.io/blog/nestjs-giai-phap-toan-dien-cho-ung-dung-server-side>, 15/02/2025.
10. Tìm hiểu chi tiết về NestJS: Framework Node.js là gì?. Internet: <https://aptech.fpt.edu.vn/nestjs-la-gi.html>, 15/02/2025.
11. MongoDB - Working and Features. Internet: <https://www.geeksforgeeks.org/what-is-mongodb-working-and-features/>, 15/02/2025.
12. MongoDB là gì? Các tính năng nổi bật của MongoDB mà bạn cần biết | 200Lab Blog. Internet: <https://200lab.io/blog/mongodb-la-gi>, 15/02/2025.
13. Giới thiệu về MongoDB: Ưu điểm và Nhược điểm. Internet: <https://igotocode.com/vi/introduction-to-mongodb-advantages-and-disadvantages>, 15/02/2025.
14. What is Nginx (Web Server) and how to install it. Internet: <https://www.geeksforgeeks.org/what-is-nginx-web-server-and-how-to-install-it/>, 15/02/2025.
15. Nginx – wikipedia. Internet: <https://en.wikipedia.org/wiki/Nginx>, 15/02/2025.
16. Tìm hiểu tổng quan về Nginx. Internet: <https://viblo.asia/p/tim-hieu-tong-quan-ve-nginx-63vKjOExZ2R>, 15/02/2025.
17. Nginx là gì? Những thông tin về Nginx lập trình viên cần biết. Internet: <https://fptshop.com.vn/tin-tuc/danh-gia/nginx-la-gi-159345>, 15/02/2025.
18. Introduction to Redis – GeeksforGeeks. Internet: <https://www.geeksforgeeks.org/introduction-to-redis-server/#advantages-of-redis-server>, 15/02/2025.
19. Redis là gì? Cách phân loại dữ liệu trong Redis hiệu quả. Internet: <https://fptshop.com.vn/tin-tuc/danh-gia/redis-la-gi-173207>, 15/02/2025.
20. Redis Explained in 5 Minutes [Ultimate Introduction Guide. Internet: <https://adevait.com/redis/what-is-redis#1-where-and-when-to-use-redis>, 15/02/2025.
21. Redis là gì? Tất tần tật ưu - nhược điểm và ứng dụng. Internet: <https://viettelidc.com.vn/tin-tuc/redis-la-gi-tat-tan-tat-uu-nhuoc-diem-va-ung-dung>, 15/02/2025.
22. MinIO – Wikipedia. Internet: <https://en.wikipedia.org/wiki/MinIO>, 15/02/2025.
23. MinIO. Internet: <https://tranhuy.bachkhoasaigon.com/minio.html>, 15/02/2025.
24. Core Administration Concepts — MinIO Object Storage for Container. Internet: <https://min.io/docs/minio/container/administration/concepts.html>, 15/02/2025.
25. MinIO: a high performance, distributed object storage system | DSRS. Internet: <https://dsrs.illinois.edu/blog/minio/>, 15/02/2025.
26. MinIO nền tảng lưu trữ đối tượng (object storage) hỗ trợ khả năng lưu trữ dữ liệu lớn - Cài đặt, Ứng dụng Linux - Course of Noha. Internet: <https://course.noha.vn/t/minio-nen-tang-luu-tru-doi-tuong-object-storage-ho-tro-kha-nang-luu-tru-du-lieu-lon/484>, 15/02/2025.
27. What is Docker? | Docker Docs. Internet: <https://docs.docker.com/get-started/docker-overview/>, 15/02/2025.
28. Docker (software) - wikipedia. Internet: <https://en.wikipedia.org/wiki/Docker_%28software%29>, 15/02/2025.
29. Docker 101: The Docker Components | Sysdig. Internet: <https://sysdig.com/learn-cloud-native/docker-101-the-docker-components/>, 15/02/2025.
30. What is Docker and what are its advantages? – Dimensiona. Internet: <https://www.dimensiona.com/en/what-is-docker-and-what-are-its-advantages/>, 15/02/2025.
31. Docker Use Cases: Most Common Ways to Use Docker. Internet: <https://www.simplilearn.com/docker-use-cases-article>, 15/02/2025.
32. The Good and the Bad of Docker. Internet: <https://www.altexsoft.com/blog/docker-pros-and-cons/>, 15/02/2025.
33. What Is Angular? Explore Angular Framework Basics & Features. Internet: <https://www.simplilearn.com/tutorials/angular-tutorial/what-is-angular#what_is_angular>, 15/02/2025.
34. What is Angular. Internet: <https://www.geeksforgeeks.org/what-is-angular/>, 15/02/2025.
35. Explain the Architecture Overview of Angular ?. Internet: <https://www.geeksforgeeks.org/explain-the-architecture-overview-of-angular/>, 15/02/2025.
36. Angular. Internet: <https://v17.angular.io>, 15/02/2025.
37. Best Angular Apps Built by 2025 [9 Angular Websites to Check]. Internet: <https://clockwise.software/blog/best-angular-applications>, 15/02/2025.
38. Linux. Internet: <https://vi.wikipedia.org/wiki/Linux>, 15/02/2025.