**CHƯƠNG II: CƠ SỞ LÝ THUYẾT**

2.3. **Front-end:**

1. **Bootstrap**:

**Bootstrap** là 1 framework HTML, CSS, và JavaScript cho phép người dùng dễ dàng thiết kế website theo 1 chuẩn nhất định, tạo các website thân thiện với các thiết bị cầm tay như mobile, ipad, tablet,...

**Bootstrap** bao gồm những cái cơ bản có sẵn như: typography, forms, buttons, tables, navigation, modals, image carousels và nhiều thứ khác. Trong bootstrap có thêm nhiều Component, Javascript hỗ trợ cho việc thiết kế reponsive của bạn dễ dàng, thuận tiện và nhanh chóng hơn.

**Bootstrap** là một trong những framework được sử dụng nhiều nhất trên thế giới để xây dựng nên một website. **Bootstrap** đã xây dựng nên 1 chuẩn riêng và rất được người dùng ưa chuộng. Chính vì thế, chúng ta hay nghe tới một cụm từ rất thông dụng "Thiết kế theo chuẩn **Bootstrap**".

Từ cái "chuẩn mực" này, chúng ta có thể thấy rõ được những điểm thuận lợi khi sử dụng **Bootstrap**.

* *Rất dễ để sử dụng*: Nó đơn giản vì nó được base trên HTML, CSS và Javascript chỉ cẩn có kiến thức cơ bản về 3 cái đó là có thể sử dụng bootstrap tốt.
* *Responsive*: **Bootstrap** xây dựng sẵn reponsive css trên các thiết bị Iphones, tablets, và desktops. Tính năng này khiến cho người dùng tiết kiệm được rất nhiều thời gian trong việc tạo ra một website thân thiện với các thiết bị điện tử, thiết bị cầm tay.
* *Tương thích với trình duyệt*: Nó tương thích với tất cả các trình duyệt (Chrome, Firefox, Internet Explorer, Safari, and Opera). Tuy nhiên, với IE browser, **Bootstrap** chỉ hỗ trợ từ IE9 trở lên. Điều này vô cùng dễ hiểu vì IE8 không support HTML5 và CSS3.

Có hai cách để sử dụng **Bootstrap:**

1. Download Bootstrap packet từ [getbootstrap.com](https://getbootstrap.com/).
2. Thêm Bootstrap từ CDN:

<!-- Latest compiled and minified CSS -->

<link rel="stylesheet" href="//netdna.bootstrapcdn.com/bootstrap/3.1.1/css/bootstrap.min.css">

<!-- Optional theme -->

<link rel="stylesheet" href="//netdna.bootstrapcdn.com/bootstrap/3.1.1/css/bootstrap-theme.min.css">

<!-- Latest compiled and minified JavaScript -->

<script src="//netdna.bootstrapcdn.com/bootstrap/3.1.1/js/bootstrap.min.js"></script>

1. **ESP**:

**EJS**, viết tắt của “**Embedded JavaScript**” là một ngôn ngữ mẫu (template language) được sử dụng trong phát triển ứng dụng web. Nó giúp phát triển các trang web động bằng cách cho phép tạo ra các mẫu HTML được kết hợp với mã JavaScript. **EJS** thường được sử dụng cùng với Node.js và các framework web như Express để tạo giao diện người dùng động dựa trên dữ liệu từ máy chủ.

Một số đặc điểm quan trọng của EJS:

* *Tích hợp dễ dàng*: **EJS** tích hợp dễ dàng vào các dự án **Node.js** thông qua **npm** (Node Package Manager). Bạn có thể cài đặt **EJS** bằng lệnh *npm install ejs* và sau đó sử dụng nó trong ứng dụng của mình.
* *Cú pháp gần gũi với HTML*: Mã **EJS** được nhúng trực tiếp vào các tệp HTML, làm cho cú pháp rất gần gũi và dễ đọc. Điều này giúp người phát triển dễ dàng thao tác với mã HTML cùng với mã JavaScript.
* *Hiển thị dữ liệu động*: Với **EJS**, bạn có thể hiển thị dữ liệu động từ máy chủ trên trang web. Bằng cách sử dụng các biến và mã JavaScript, bạn có thể hiển thị danh sách, dữ liệu cơ sở dữ liệu, và thông tin động khác trên trang web.
* *Cấu trúc kiến thức*: **EJS** hỗ trợ cấu trúc kiến thức như vòng lặp, câu điều kiện, và các hàm tạo mẫu. Điều này cho phép bạn tạo các trang web phức tạp và linh hoạt.
* *Thay thế biến môi trường*: Bạn có thể sử dụng biến môi trường trong **EJS** để truyền các giá trị môi trường như đường dẫn tới tệp tin, ngôn ngữ, và cài đặt khác vào các mẫu.
* *Phát triển web đa ngôn ngữ*: **EJS** hỗ trợ phát triển các ứng dụng web đa ngôn ngữ bằng cách cho phép bạn thay đổi các văn bản trong mẫu dựa trên ngôn ngữ được chọn.
* *Tương thích với nhiều trình duyệt*: Mã **EJS** được tạo ra dưới dạng mã HTML chuẩn, do đó nó tương thích với hầu hết các trình duyệt web phổ biến.

2.4**. Back-end:**

1. **Node.js:**

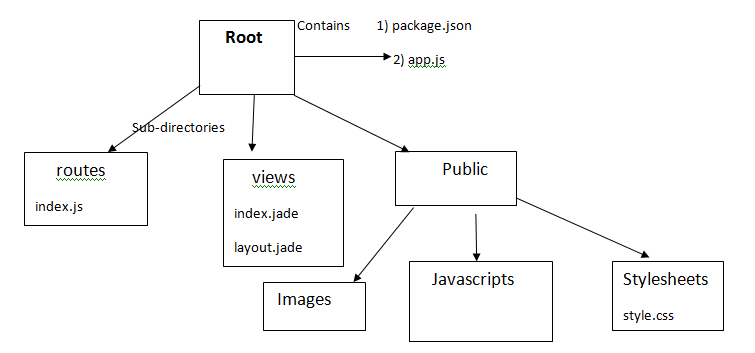
Định nghĩa trên trang chủ của **Node.js**: **Node.js** là một JavaScript runtime được build dựa trên Chrome’s V8 JavaScript engine. **Node.js** sử dụng mô hình event-driven, non-blocking I/O khiến nó trở nên nhẹ và hiệu quả.

**Node** đặc biệt thích hợp để xây dựng những app cần đến tương tác hay hợp tác real-time, ví dụ như chat site, hoặc app kiểu CodeShare, nơi mà bạn có thể xem document được chỉnh sửa trực tiếp bởi người khác. **Node** cũng phù hợp để tạo các API nơi bạn cần xử lý một lượng lớn request liên quan đến I/O hoặc với những site liên quan đến data streaming, **Node** mang lại khả năng xử lý file trong khi chúng đang trong quá trình upload. Nếu bạn quan tâm đến vấn đề real-time.

Bên cạnh tốc độ và khả năng mở rộng, việc dùng JavaScript trên server và trình duyệt giúp cho bạn đỡ phải chuyển đổi giữa các ngôn ngữ. Bạn có thể làm bất cứ thứ gì với duy nhất một ngôn ngữ. Một điểm mạnh nữa của **Node** đó là nó cực thích hợp với JSON. JSON được coi là lý tưởng khi được sử dụng bởi một chương trình JavaScript. Khi làm việc với **Node**, dữ liệu có thể chuyển qua các tầng layer mà không cần phải reformat. Cuối cùng, JavaScript thực sự quá phổ biến, điều này giúp cho việc chuyển sang phát triển **Node** app dễ dàng hơn so với những ngôn ngữ phía server khác.

1. **Express**:

**Expressjs** là một framework được xây dựng trên nền tảng của **Nodejs**. Nó cung cấp các tính năng mạnh mẽ để phát triển web hoặc mobile. **Expressjs** hỗ trợ các method HTTP và midleware tạo ra API vô cùng mạnh mẽ và dễ sử dụng.



Hình 1: Cấu trúc thư mục của Express

Tính năng của **Express**:

A purple and white circle with white circles and white circles with white text

Description automatically generated with medium confidence

Hình 2: Các tính năng của Express

* **Phát triển máy chủ nhanh hơn**: **ExpressJS** tối ưu hóa cú pháp và cung cấp các phương thức và hàm tiện ích để xử lý các tác vụ phổ biến trong lập trình web. Nhờ đó, bạn có thể viết code ngắn gọn và tối giản hóa quy trình phát triển.
* *Định tuyến (Routing)*: **ExpressJS** cung cấp một hệ thống định tuyến mạnh mẽ, cho phép bạn xác định các tuyến đường (routes) để xử lý yêu cầu từ người dùng và phản hồi tương ứng. Điều này giúp tổ chức và quản lý các thành phần của ứng dụng một cách dễ dàng.
* *Middleware*: **ExpressJS** hỗ trợ middleware, cho phép bạn thêm các chức năng trung gian vào quy trình xử lý yêu cầu và phản hồi. Middleware giúp xác thực người dùng, ghi log, xử lý lỗi, nén dữ liệu và thực hiện nhiều tác vụ khác một cách linh hoạt.
* *Cấu hình môi trường*: **ExpressJS** cung cấp một cách để cấu hình môi trường phát triển và môi trường sản xuất. Bạn có thể thiết lập các biến môi trường, cấu hình định dạng và quy tắc, tùy chỉnh ứng dụng của mình theo các môi trường khác nhau.
* *Xử lý lỗi*: **ExpressJS** cung cấp cơ chế xử lý lỗi cho phép bạn kiểm soát và xử lý các lỗi xảy ra trong quá trình xử lý yêu cầu. Bạn có thể tạo ra các middleware để xử lý lỗi và phản hồi với các thông báo lỗi tùy chỉnh.

Ưu điểm của **Express**:

* *Đơn giản và dễ sử dụng*: **ExpressJS** có cú pháp đơn giản và dễ hiểu, giúp lập trình viên dễ dàng nắm bắt và triển khai các tính năng.
* *Linh hoạt*: **ExpressJS** không áp đặt một cấu trúc cụ thể, cho phép lập trình viên tự do tùy chỉnh và xây dựng ứng dụng theo ý muốn.
* *Hỗ trợ middleware*: **ExpressJS** cung cấp hệ thống middleware mạnh mẽ, cho phép thực hiện các chức năng như xác thực, ghi log, nén dữ liệu và xử lý lỗi một cách linh hoạt và dễ dàng.
* *Hiệu suất cao*: **ExpressJS** được xây dựng trên Node.js, nền tảng có hiệu suất cao, cho phép xử lý nhanh chóng các yêu cầu web đồng thời và có khả năng mở rộng tốt.

Nhược điểm của **Express**:

* *Thiếu cấu trúc*: Do **ExpressJS** không áp đặt một cấu trúc nghiêm ngặt, việc tổ chức dự án và quản lý mã nguồn có thể trở nên khó khăn, đặc biệt khi ứng dụng phát triển lớn và phức tạp.
* *Khả năng mở rộng*: Khi ứng dụng phát triển lớn và phức tạp, việc quản lý mã nguồn và mở rộng có thể trở nên khó khăn với **ExpressJS**. Cần có sự kiểm soát cẩn thận để tránh sự phức tạp và rối rắm trong việc quản lý các module và tương tác giữa chúng.
* *Cộng đồng hỗ trợ*: Mặc dù **ExpressJS** có một cộng đồng lớn và đầy đủ tài liệu, tuy nhiên, không đạt được mức độ hỗ trợ như các framework web khác như Angular hoặc React.

**CHƯƠNG III: PHÂN TÍCH THIẾT KẾ HỆ THỐNG.**

3.1. **Use-case**:

* + 1. **Các tác nhân và ca sử dụng**:

Sơ đồ use-case cho biết người dung (actor) sẽ tương tác với hệ thống như thế nào.

* Quản trị viên hệ thống (Admin):
  + Thêm thẻ.
  + Xóa thẻ.
  + Cập nhật thông tin thẻ.
  + Xem thông tin thẻ.
  + Thêm nhân viên,
  + Xóa nhân viên.
  + Phân quyển.
  + Quản lý nhân viên (xem).
  + Xem lịch sử ra vào.
  + Thống kê doanh thu khách vãng lai.
* Nhân viên an ninh:
  + Cập nhật lịch sử ra vào.
  + Đóng barrier.
  + Mở barrier.
* Người dùng (cư dân):
  + Đăng kí tài khoản.
  + Đăng nhập.
  + Đăng xuất.
  + Xem thông tin thẻ.
  + Xem lịch sử ra vào của xe.
  + Yêu cầu thêm, sửa, xóa thẻ.
  1. **Mô tả kịch bản các ca sử dụng**:
     1. **Kịch bản sử dụng ca đăng nhập**:
* Người dùng yêu cầu đăng nhập.
* Hệ thống hiển thị giao diện đăng nhập.
* Người dùng điền thông tin tên đăng nhập, mật khẩu, xác nhận đăng nhập.
* Hệ thống kiểm tra thông tin đăng nhập.
  + Hợp lệ, đăng nhập thành công, chuyển thao tác hoạt động của hệ thống.
  + Không hợp lệ, thống báo người dùng sai thông tin hoặc tài khoản không tồn tại, quay lại bước đăng nhập (có thể đăng ký).
    1. **Kịch bản sử dụng ca đăng ký tài khoản**:
* Người dùng yêu cầu đăng ký tài khoản.
* Hệ thống hiện giao diện đăng ký tài khoản.
* Người dùng điền đầy đủ thông tin hệ thống yêu cầu gồm mã thẻ, mật khẩu, thông tin cá nhân, sau đó yêu cầu đăng ký.
* Hệ thống kiểm tra thông tin đăng ký. Hợp lệ, thông báo “Tạo tài khoản thành công” đồng thời lưu thông tin tài khoản của người dùng trên web và hệ thống.
  + 1. **Kịch bản sử dụng ca tạo thẻ**:
* Quản lý (người dùng) yêu cầu tạo thẻ.
* Hệ thống hiện giao diện đăng ký tạo thẻ.
* Quản lý điền đầy đủ thông tin vào các mục trong giao diện mà hệ thống yêu cầu, bao gồm các thông tin: Tên chủ hộ, ID căn hộ, tên chủ thẻ, ID thẻ, loại thẻ (vé tháng, …), ngày tạo thẻ.
* Hệ thống kiểm tra thông tin đăng ký. Không có thông tin gì sai thì yêu cầu tạo thẻ.
* Hệ thống thông báo tạo thẻ thành công và lưu thông tin thẻ trên hệ thống.
  + 1. **Kịch bản sử dụng ca yêu cầu thêm, sửa, xóa thẻ**:
* Người dùng gửi yêu cầu về việc thêm, sửa hoặc xóa thẻ.
* Hệ thống hiện giao diện form điền thông tin yêu cầu.
* Người dùng thực hiện điền thông tin vào form:
  + Nếu người dùng muốn thêm thẻ cần cung cấp: tên chủ hộ, số căn hộ, tên chủ thẻ.
  + Nếu người dùng muốn sửa thẻ cần cung cấp thông tin muốn sửa và thông tin sau khi đã sửa lại.
  + Nếu người dùng muốn xóa thẻ cần cung cấp tên chủ thẻ, tên chủ hộ, số căn hộ và lý do muốn xóa thẻ.
* Sau khi thực hiện gửi yêu cầu thêm, sửa hoặc xóa thành công, hệ thống sẽ hiện thông báo đã tiếp nhận yêu cầu thành công, đồng thời hệ thống cũng lưu trữ lại tất cả các yêu cầu đã tiếp nhận từ người dùng.
  + 1. **Kịch bản sử dụng ca xem thông tin thẻ**:
* Người dùng truy cập website của hệ thống.
* Hệ thống sẽ hiển thị giao diện có bảng gồm các mục, trong đó có mục quản lý thẻ.
* Người dùng yêu cầu xem thông tin thẻ.
* Hệ thống sẽ hiển thị giao diện danh sách các thông tin chi tiết được lưu trữ trong thẻ bao gồm: tên chủ thẻ, tên chủ hộ, số căn hộ, ID thẻ, loại thẻ (thẻ tháng, …), ngày tạo thẻ.
  + 1. **Kịch bản sử dụng ca đóng mở barrier (tự động)**:
* Khi người dùng vào bãi xe:
  + Người dùng (sử dụng phương tiện cá nhân) di chuyển đến trước khu vực barrier để tiến hành vào bãi xe.
  + Người dùng đứng trước vạch kẻ quy định, tiến hành sử dụng thẻ ra vào cá nhân quẹt thẻ.
  + Hệ thống tại cabin kiểm soát sẽ đọc thẻ của người dùng để xác nhận thông tin. Hệ thống sẽ ghi nhận biển kiểm soát của phương tiện, thời gian vào bãi, ảnh chụp toàn cảnh phương tiện và người điều khiển phương tiện, đồng thời sẽ lưu các thông tin đó vào thẻ người dùng khi vào bãi xe.
  + Hệ thống sau đó ghi nhận phương tiện đã có mặt trong bãi xe.
  + Ngay sau khi các thông tin được hệ thống ghi nhận một cách nhanh chóng thì barrier sẽ được tự động mở lên, các thông tin đồng thời được lưu vào hệ thống.
* Khi người dùng ra khỏi bãi xe:
  + Người dùng sử dụng phương tiện di chuyển đến vạch dừng trước barrier.
  + Người dùng sử dụng thẻ ra vào cá nhân để thực hiện quẹt thẻ.
  + Hệ thống sẽ kiểm tra các thông tin trong thẻ của người dùng.
  + Hệ thống sẽ kiểm tra trạng thái tồn tại của xe trong bãi, nếu đã xuất hiện trong bãi xe thì có nghĩa là xe có thể đi ra, ngược lại là người dùng có thể đã sử dụng sai phương tiện hoặc thẻ để đi ra.
  + Hệ thống đồng thời sẽ đối chiếu biển kiểm soát đọc được từ camera với biển kiểm soát đã lưu trong thẻ lúc xe vào, đối chiếu ID thẻ, số căn hộ, tên chủ hộ:
    - Nếu thông tin trùng hết thì xe có thể ra.
    - Nếu trùng biển kiểm soát mà khác ID thẻ thì đối chiếu số căn hộ và tên chủ hộ, nếu trùng thì xe có thể đi ra, nếu không trùng thì khả năng là xe bị cướp.
  + Sau khi đối chiếu các thông tin xong barrier sẽ tự động mở, các thông tin được đối chiếu được lưu vào hệ thống.
* Đối với khách vãng lai ra vào bãi xe:
  + Lúc vào:
    - Người dùng tiếp nhận thẻ từ nhân viên an ninh, người dùng quẹt thẻ.
    - Hệ thống sẽ đọc biển số, chụp ảnh toàn cảnh phương tiện và người điều khiển phương tiện, ghi nhận thời gian vào bãi xe vào thẻ, trạng thái (có trong bãi).
    - Barrier tự mở lên.
  + Lúc ra:
    - Người dùng quẹt thẻ.
    - Hệ thống đối chiếu biển kiểm soát, trạng thái.
    - Barrier mở.
    1. **Kịch bản sử dụng ca đóng mở barrier (thủ công)**:
* Người dùng (nhân viên an ninh) sử dụng chức năng nhập thủ công.
* Người dùng nhập các thông tin: biển số xe, tên chủ thẻ, tên chủ hộ, mã căn hộ
  + 1. **Kịch bản sử dụng ca xem lịch sử ra vào**:
* Người dùng truy cập website hệ thống.
* Người dùng đăng nhập.
* Người dùng chọn chức năng xem lịch sử ra vào.
* Hệ thống hiển thị lịch sử ra vào mặc định theo dòng thời gian, lần ra vào gần nhất sẽ được hiển thị trên cùng.
* Hệ thống hiển thị các thông tin về lịch sử ra vào gồm: tên chủ thẻ, tên chủ hộ, mã căn hộ, id thẻ, loại phương tiện, biển kiểm soát, thời gian vào bãi, thời gian ra bãi.
* Hệ thống hiển thị chức năng bộ lọc, tăng giảm dần theo thời gian, lọc khoảng thời gian, lọc theo loại phương tiện, …
* Hệ thống hiển thị chức năng tìm kiếm theo tên chủ thẻ, id thẻ, …
  + 1. **Kịch bản sử dụng ca đăng xuất**:
* Người dùng ấn nút đăng xuất.
* Hệ thống kết thúc phiên, giao diện hiện thị trang đăng nhập.
  1. **Sơ đồ phân rã chức năng**:

A black and white screen with white rectangles

Description automatically generated

Hình 3: Sơ đồ phân ra chức năng

* 1. **Sơ đồ mức ngữ cảnh**:

A screenshot of a computer

Description automatically generated

Hình 4: Sơ đồ mức ngữ cảnh

* 1. **Sơ đồ mức đỉnh**:

A screenshot of a computer screen

Description automatically generated

A black background with a black square

Description automatically generated with medium confidence

Hình 5: Sơ đồ mức đỉnh

* 1. **Sơ đồ mức dưới đỉnh**:
     1. **Quản lý hệ thống**:

A screenshot of a computer

Description automatically generated

Hình 6: Sơ đồ mức dưới đỉnh - Quản lý hệ thống

* + 1. **Quản lý danh mục**:

A screenshot of a computer screen

Description automatically generated

Hình 7: Sơ đồ mức dưới đỉnh - Quản lý danh mục

* + 1. **Quản lý lịch sử vào ra**:

A screenshot of a computer

Description automatically generated

Hình 8: Sơ đồ mức dưới đỉnh - Quản lý lịch sử vào ra

* + 1. **Báo cáo thống kê**:

A screenshot of a computer screen

Description automatically generated

Hình 9: Sơ đồ mức dưới đỉnh - Báo cáo thống kê

* 1. **Sơ đồ liên kết thực thể**:
     1. **Các thực thể và thuộc tính**:

A white and black text box

Description automatically generated with medium confidence

Hình 10: Bảng thực thể

* + 1. **Mối liên kết giữa các thực thể**:

A white and black list with black text

Description automatically generated

Hình 11: Mối liên kết giữa các thực thể

* + 1. **Sơ đồ liên kết thực thể**:

A screenshot of a computer screen

Description automatically generated

Hình 12: Sơ đồ liên kết thực thể

* 1. **Bảng cơ sở dữ liệu SQL**:

A screenshot of a computer

Description automatically generated

Hình 13: Cơ sở dữ liệu