HNRG

Versión 0.1.0

grupo 74

10 de julio de 2018

Índice general

CAPÍTULO 1

Introducción

En la presente documentación se detallan teclogías y estandares adoptados para el desarrollo de la aplicación para el Hospital de Niños Ricardo Gutierrez, así como también la fundamentación de cada una de las decisiones tomadas a lo largo del desarrollo.

1.1 Convensiones utilizadas en la documentación

Este manual contiene muchos ejemplos que se pueden escribir en el teclado. Un comando ingresado en la terminal se muestra así:

```
$ command
```

El primer cáracter en la línea es el prompt de la terminal, y no debería ser tipeado. El signo de dólar $ se usa como la solicitud estándar en esta documentación, aunque algunos sistemas pueden usar un símbolo diferente.

Los ejemplos asumen el uso del sistema operativo GNU. Puede haber diferencias menores en la salida en otros sistemas. Los comandos para las variables de entorno de configuración utilizan la sintaxis del shell Bourne del shell GNU estándar (`bash`).

CAPÍTULO 2

Roles y Permisos

En la aplicación se definen 3 roles básicos con permisos para acceder a diferentes secciones del sistema.

2.1 Roles básicos

A continuación se describiran los roles y sus permisos asociados:

- **Administrador:**
    - usuario_index
    - usuario_show
    - usuario_new
    - usuario_update
    - usuario_destroy
    - paciente_index
    - paciente_destroy
    - rol_index
    - rol_show
    - rol_new
    - rol_update
    - rol_destroy
    - control_salud_destroy
- **Recepcionista:**
    - paciente_index
    - paciente_show

        - paciente_new
        - paciente_update
- **Pediatra:**
        - paciente_index
        - paciente_show
        - paciente_new
        - paciente_update
        - control_salud_index
        - control_salud_show
        - control_salud_new
        - control_salud_update

A modo de proveer una manera de recorrer más facilmente el sistema para busqueda de errores, o simplemente para conocer el sitio, se provee un rol más, denominado su el cual cuenta con **todos** los permisos disponibles en el sistema. Si bien con contar con el rol Administrador y cualquiera de los roles restantes ya se cuenta con todos los permisos, se creo a modo de abstraerse de dicha asignación, ya que esta puede cambiar con el tiempo.

- **Su:**
        - usuario_index
        - usuario_show
        - usuario_new
        - usuario_update
        - usuario_destroy
        - paciente_index
        - paciente_show
        - paciente_new
        - paciente_update
        - paciente_destroy
        - rol_index
        - rol_show
        - rol_new
        - rol_update
        - rol_destroy
        - control_salud_index
        - control_salud_show
        - control_salud_new
        - control_salud_update
        - control_salud_destroy
        - debug_index

- log_index

2.2 Usuarios

Conociendo los roles, se cuenta con 5 usuarios básicos.

- **Administrador:**
  - rol: Administrador
  - usuario: admin
  - contraseña: admin
- **Recepcionista:**
  - rol: Recepcionista
  - usuario: recepcionista
  - contraseña: recepcionista
- **Pediatra:**
  - rol: Pediatra
  - usuario: pediatra
  - contraseña: pediatra
- **Super Administrador:**
  - rol: Administrador y Su
  - usuario: su
  - contraseña: su

Se decidió asignar el rol admninistrador al *Super Administrador* debido a que ciertas secciones requiren no de un permiso especifico, sino del rol **Administrador**.

2.3 Errores

El sistema esta preparado para mostrar pantallas de error con los siguientes códigos:

- 403: Esto significa que su usuario no cuenta con el permiso necesario para acceder a la página solicitada. Normalmente bastaría con asignar un rol que cuente con el permiso necesario.
- 404: Esto significa, como todos sabemos, que el recuerso buscado no se encuentra disponible.
- 500: Este error es el mostrado cuando el sistema se encuentra en *Mantenimiento* y va a ser el inicial durante estas semanas a modo de ilustración (ya que nos encontramos en mantenimiento del sitio cerrando ciertos módulos, ¿no?). Para poder acceder al sitio libremente, se debe ingresar como **Administrador** o como **Super Administrador** y desactivar el modo mantenimiento.

Todos los errores cuentan con un pequeño texto descriptivo para orientar al usuario del sistema la causa del error.

CAPÍTULO 3

API para la gestión de turnos

La app provee una interfaz para interactuar con el módulo de gestión de turnos del hospital.

La API se encuentra en la URL `grupo74.proyecto2017.linti.unlp.edu.ar/api/turnos`.

3.1 Métodos de la API

A continuación se muestran los métodos disponibles en la API así como las consultas a ejecutar para probar el correcto funcionamiento de los mismos utilizando el comando `curl` o `wget`.

- Obtener turnos para la fecha actual

Permite obtener los turnos disponibles para la fecha actual.

GET /

Respuesta correcta con turnos disponibles,

```
$ curl grupo74.proyecto2017.linti.unlp.edu.ar/api/turnos
```

```
HTTP/1.1 200 OK
{
    "appointments": [
        "10:00:00",
        "13:30:00",
        "14:00:00",
        "15:30:00"
    ]
}
```

- Obtener turnos para la fecha dada

Permite obtener los turnos disponibles para la fecha dada.

GET /:date

Respuesta correcta con turnos disponibles,

```
$ curl grupo74.proyecto2017.linti.unlp.edu.ar/api/turnos/YYYY-MM-DD
```

```
HTTP/1.1 200 OK
{
    "appointments": [
        "10:00:00",
        "13:30:00",
        "14:00:00",
        "15:30:00"
    ]
}
```

Respuesta erronea por fecha vencida,

```
$ curl grupo74.proyecto2017.linti.unlp.edu.ar/api/turnos/2017-MM-DD
```

```
HTTP/1.1 204 NO CONTENT
```

Respuesta erronea por error genérico,

```
$ curl grupo74.proyecto2017.linti.unlp.edu.ar/api/turnos/una-frase
```

```
HTTP/1.1 500 Internal Server Error
```

- Reservar un turno

Permite reservar un turno para una fecha y hora dada, para un nro de documento dado.

GET /turnos/:documentNumber/fecha/:date/hora/:time

Donde:

- documentNumber: número de documento del paciente a atender.
- date: fecha del turno (se puede comprobar existencia en la ruta de `turnos disponibles`)
- time: hora del turno (se puede comprobar existencia en la ruta de `turnos disponibles`)

Respuesta correcta con turno reservado,

```
$ curl grupo74.proyecto2017.linti.unlp.edu.ar/api/turnos/40000000/fecah/YYYY-MM-DD/
↪hora/HH:mm
```

```
HTTP/1.1 201 Created
{
    "appointment": {
        "documentNumber": 40000000,
        "date": "DD MM YYYY HH:mm:ss TZ"
    }
}
```

Respuesta erronea por falta de parámetros, o fecha/hora inválida

```
$ curl grupo74.proyecto2017.linti.unlp.edu.ar/api/turnos/no-doc/fecah/una-frase/hora/
```

```
HTTP/1.1 400 BAD REQUEST
```

Respuesta erronea por turno ya tomado

```
$ curl grupo74.proyecto2017.linti.unlp.edu.ar/api/turnos/40000000/fecah/YYYY-MM-DD/
↪hora/HH:mm
```

```
HTTP/1.1 422 UNPROCESSABLE ENTITY
```

Respuesta erronea por error genérico,

```
$ curl grupo74.proyecto2017.linti.unlp.edu.ar/api/turnos/una-frase
```

```
HTTP/1.1 500 Internal Server Error
```

- Eliminar un turno dado un id

Permite eliminar un turno dado su id

DELETE /turnos/:id

Respuesta correcta con un turno eliminado,

```
$ curl -XDELETE grupo74.proyecto2017.linti.unlp.edu.ar/api/turnos/id
```

```
HTTP/1.1 200 OK
```

Respuesta erronea por error genérico,

```
$ curl -XDELETE grupo74.proyecto2017.linti.unlp.edu.ar/api/turnos/id
```

```
HTTP/1.1 500 Internal Server Error
```

CAPÍTULO 4

Ambiente de Desarrollo

4.1 Requerimientos

- Mínimos
  - NodeJs v9.5.0
  - Yarn v1.7.0
  - Mongo v3.6.3
  - direnv v2.4.0 o superior
  - redis v4
- Recomendaciones
  - Utilizar nvm para el versionado de NodeJs
- Opcionales
  - docker
  - docker-compose
  - mongodb-compass

4.2 Instalación del ambiente

si le falta cumplir alguno de los requerimientos, seguir a la siguiente sección

Instalación de los componentes necesarios utilizando `yarn`:

```
$ yarn
$ yarn build:dev
```

Con esto, se crea la carpeta node-modules, y algunos archivos extra de configuración.

4.2.1 Generación de las imágenes docker

Nota: Se recomienda agregar su usuario al grupo de docker para evitar la necesidad de ejecutar docker con sudo. Esto se puede hacer ejecutando:

```
$ sudo usermod -a -G docker $USER
```

Posteriormente, podemos seguir 2 diferentes caminos, usando docker y docker-compose, o usando simplemente docker, llegando siempre al mismo resultado.

Docker + Docker-compose (recomendable)

Se utilizan los archvios `Dockerfile` y `docker-compose.yml` (para producción) o `Dockerfile-development` y `docker-compose-development.yml` (para desarrollo).

Ambiente Desarrollo

```
$ docker-compose -f docker/docker-compose-development.yml build
$ docker-compose -f docker/docker-compose-development.yml up
```

Con esto se genera la imagen docker, y corre tanto el servidor node, como mongodb.

Ambiente Producción

```
$ docker-compose -f docker/docker-compose.yml build
$ docker-compose -f docker/docker-compose.yml up
```

4.2.2 Usando yarn start

Para esta opción, vamos a necesitar tener todos los servicios (mongo por ejemplo), corriendo en nuestra maquina al momento de iniciar la aplicación, ademas de contar con las variables de ambiente seteadas correctamente con la dirección de host, puerto, y las credenciales necesarias para acceder al servicio.

La idea es ejecutar la siguiente instrucción en la terminal,

```
$ yarn start
```

4.3 Parametrización del ambiente

Para evitar futuros problemas, se evita en todo momento escribir las credenciales utilizadas para los diferentes servicios en los archivos de configuración. En su lugar, utilizan variables de ambiente.

Para facilitar el uso de las mismas, se recomienda utilizar *direnv* exportando todas las variables necesarias en el archivo *.envrc*.

Eso se hace de la siguiente manera:

```
export TU_CREDENCIAL=CREDENCIAL
```

Recordar nunca quitar el archivo .envrc del gitignore

Para el correcto funcionamiento del archivo .envrc es necesario contar con **direnv**_

Notar que se debe agregar la linea eval en el archivo de configuracion de su shell

En bash por ejemplo, se agrega la siguiente linea en el .bashrc

```
eval "$(direnv hook bash)"
```

Para conocer más sobre que línea agregar y como configurarlo en las diferentes shell, se recomienda leer el repositorio_ en github

Ademas, **cada vez que se modifique** *se debe ejecutar:_*

```
$ direnv allow
```

4.4 Visualización de los datos

Para visualizar los datos de una manera más cómoda, se recomienda la utilización de Mongodb Compass, una app de los creadores de mongo, que permite visualización de datos, estadísticas, entre otras funcionalidades interesantes.

Dependiendo como se haya levantado el ambiente, las configuraciones que se van a tener que ingresar en Mongodb Compass.

Si el ambiente se levantó completamente local (es decir, instalando los programas directamente en el SO), no va a ser necesario cambios, a menos que se hayan cambiados las configuraciones por defecto de Mongodb. Por defecto, son las siguientes:

- Hostname: localhost
- Port: 27017
- Authentication: None
- SSL: off
- SSH Tunel: off

De utilizar el ambiente dockerizado, va a ser necesario especificar el hostname y puerto especificado al momento de levantar el ambiente. En los archivos docker-compose se utiliza la siguiente configuración:

- Hostname: 172.17.0.1
- Port: 27017
- Authentication: none
- SSL: off
- SSH Tunel: off

4.5 Manejo del repositorio en desarrollo

Para el manejo del repositorio en etapas de desarrollo, se recomienda la utilización de `git-flow`, para una mayor organización en el desarrollo.

Además se recomienda también utilizar los `git-flow-hooks`, para una mayor comodidad. Además de agregar la versión actual del código al archivo «VERSION», de no utilizar git-flow-hooks, mantener en **todo momento** actualizado dicho archivo para poder mostrar dicha versión en la app funcionando en producción.

4.6 Extra

Para que las consultas a las apis sean más amigables, se propone utilizar el gist api_request, el cual es un wrapper de curl, agregando un beautifier para las respuestas en JSON.

4.7 Levantar ambiente de desarrollo

Para levantar el ambiente de desarrollo completamente local, hace falta contar con un programa extra llamado ngrok, o algún programa similar para completar el funcionamiento del bot de telegram.

En primera instancia es necesario compilar los js, utilizando `npm` o `yarn`.

```
# con yarn
$ yarn build:dev
# o con npm
$ npm run build:dev
```

Luego es necesario abrir un túnel al localhost para que el webhook del bot de telegram se pueda comunicar con nuestra app localmente, para esto corremos el siguiente comando:

```
$ ngrok http localhost:8000
```

En este caso se utilizó el puerto utilizado por la app, tener en cuenta cambiarlo en el comando si se utiliza otro.

Luego es necesario definir la variable `URL_BASE` en la **misma** terminal que se va a correr el servidor localmente, con la url asignada por ngrok, de la siguiente manera:

```
$ export URL_BASE=<url https de ngrok>
```

Tener en cuenta que se debe utilizar la versión **https** provista por ngrok.

Luego solo hace falta correr el docker-compose de la siguiente manera:

```
$ docker-compose -f docker/docker-compose-development.yml up --build
```

Con esto se debería contar con todo el ambiente levantado, listo para atender consultas en el bot de telegram.

CAPÍTULO 5

Indices and tablas

- genindex

Índice