Manual Técnico

En el manual ténico se tendrá la información del backend del programa donde se maneja toda la lógica de la página web como de sus procesos de información

El archivo index será el archivo que nos permitirá montar nuestro servidor para ejecutar la página de manera correcta sin interrupciones ni necesidades de recargar a la hora de modificar los archivos dentro del proyecto.

Se le asignó el puerto 4000 al servidor del backend

El archivo routes nos permitirá enlazar el backend con el frontend por medio de rutas de enlace que nosotros definiremos para cada función.

La primera para la página principal no tendrá nada más que un saludo.

La ruta para la página login tendrá la solicitud al array de los usuarios para poder verificarlo con los datos colocados por el usuario para conocer si puede o no avanzar en la página además de su rol y al módulo que se dirigirá.

```
router.post('/login', (req,res) => {
    const { username, password } = req.body

    /**if (username === 'admin' && password === 'admin') {
        res.send('Login correcto')
    } else {
        res.status(401).send('Login incorrecto')
    }*/

    const user = findUser(username, password)

    if(user) {
        res.json({message: 'Usuaro logueado con éxito', user})
    } else {
        res.status(401).json({ message: 'Usuario o contraseña incorrectos'})
    }
})
```

Las siguientes rutas son /register que nos permitirá mandar los datos colocados en la página al array para registrar nuevos pacientes.

La ruta /patients solicitará el array de los pacientes.

La ruta /medicinas solicitará el array de las medicinas.

La ruta /comprarMedicina mandará los datos de la medicina comprada al array para la factura del paciente.

```
router.post('/register', (req, res) => {
    const user = req.body
    user.push(user)
    res.json({ message: 'Usuario agregado exitosamente' })
})

router.get('/patients', (req, res) => {
    const patients = getPatiens()
    res.json(patients)
})

router.get('/medicinas', (req, res) => {
    const medicinas = getMedicinas()
    res.json(medicinas)
})

router.post('/compraMedicina', (req, res) => {
    const { idMedicina, cantidad, idPaciente } = req.body

    agregarCompraDeMedicina({
        idMedicina,
        cantidad,
        idPaciente
    })

    res.json({ message: 'Compra de medicina exitosa' })
})
```

En el archivo users.js tendremos las funciones que nos permitirá manejar el array de los usuarios, tales como encontrar un usuario por medio del username y el password para su ingreso en el login, agregar nuevos usuarios, encontrar usuarios por medio de su ID, recibir los pacientes registrados, recibir los doctores registrados y exportar todas las funciones.

```
63  const findUser = (username, password) => {
64      return users.find(user => user.username === username && user.password === password)
65  }
66
67  const agregarUser = (user) => {
68      users.push(user)
69  }
70
71  const getPatiens = () => {
72      return users.filter(user => user.role === 'patient')
73  }
74
75  const findUserId = id => {
76      return users.find(user => user.id === id)
77  }
78
79  const findDoctors = () => {
80      return users.find(user => user?.role === 'doctor')
81  }
82
83  export { users, findUser, getPatiens, agregarUser, findUserId, findDoctors}
```

El archivo facturas.js tendrá el array de las facturas de los pacientes además de las funciones para agregar nuevas u obtener las ya registradas.

src > data > JS facturas.js > ...

```
1   const facturas = []
2
3   const agregarFactura = (factura) => {
4       facturas.push(factura)
5   }
6
7   const obtenerFacturas = () => {
8       return facturas
9   }
10
11  export { agregarFactura, obtenerFacturas}
```

El archivo de medicinas.js tendrá el array de las medicinas registradas junto a sus datos tales como para que tipo de problemas funcionan, precio y la cantidad en inventario.

Tendremos la función para recibir las medicinas registradas.

src > data > JS medicinas.js > ...

```
const medicinas = [
    {
        id: 123,
        nombre: 'Paracetamol',
        descripcion: 'Medicamento para el dolor',
        precio: 10,
        cantidad: 100,
    },
    {
        id: 213,
        nombre: 'Gelocatil',
        descripcion: 'Descongestivo nasal',
        precio: 75.5,
        cantidad: 100,
    }
]

const getMedicinas = () => {
    return medicinas
}

export { getMedicinas }
```

El archivo recetas.js contará con los array para las recetas registradas por el doctor y las medicinas compradas por los pacientes de acuerdo a las recetas.

Contará con las funciones para que el doctor agregue nuevas recetas y para agregar nuevas compras de medicina según lo recetado por el doctor al paciente.

src > data > JS recetas.js > ...

```
const recetas = [

]

const medicinasCompradas = [

]

const agregarReceta = (receta) => {
    recetas.push(receta)
}

const agregarCompraDeMedicina = (compra) => {
    medicinasCompradas.push(compra)
}

export { agregarCompraDeMedicina, agregarReceta, medicinasCompradas, recetas }
```

El archivo cita.js tendrá el array con las citas programadas para que la enfermera acepte o rechace para que el doctor los reciba en las consultas.

Tendrá la función que permitirá asignarle un doctor libre al paciente en su consulta.

src > data > JS cita.js > [@] citas

```
import { findUserId } from "./users.js"

const citas = [
    {
        id: 1,
        patientId: 12,
        doctorId: null,
        date: '2023-12-04',
        time: '9:50',
        motivo: 'Dolor de espalda'
    },
    {
        id: 2,
        patientId: 12,
        doctorId: null,
        date: '2023-11-08',
        time: '15:23',
        motivo: 'gripe'
    }
]

const findPendingAppointments = () => {
    const pendingAppointments = citas.filter(appointment => appointment.doctorId === null)
    return pendingAppointments.map(appointment => {
        const patient = findUserId(appointment.patientId)
        return {
            ...appointment,
            patient
        }
    })
}

export {
    findPendingAppointments
}
```