Documentación Proyecto Plataforma INSTAYA

NRC: 2242
Equipo No. 03
Integrantes:

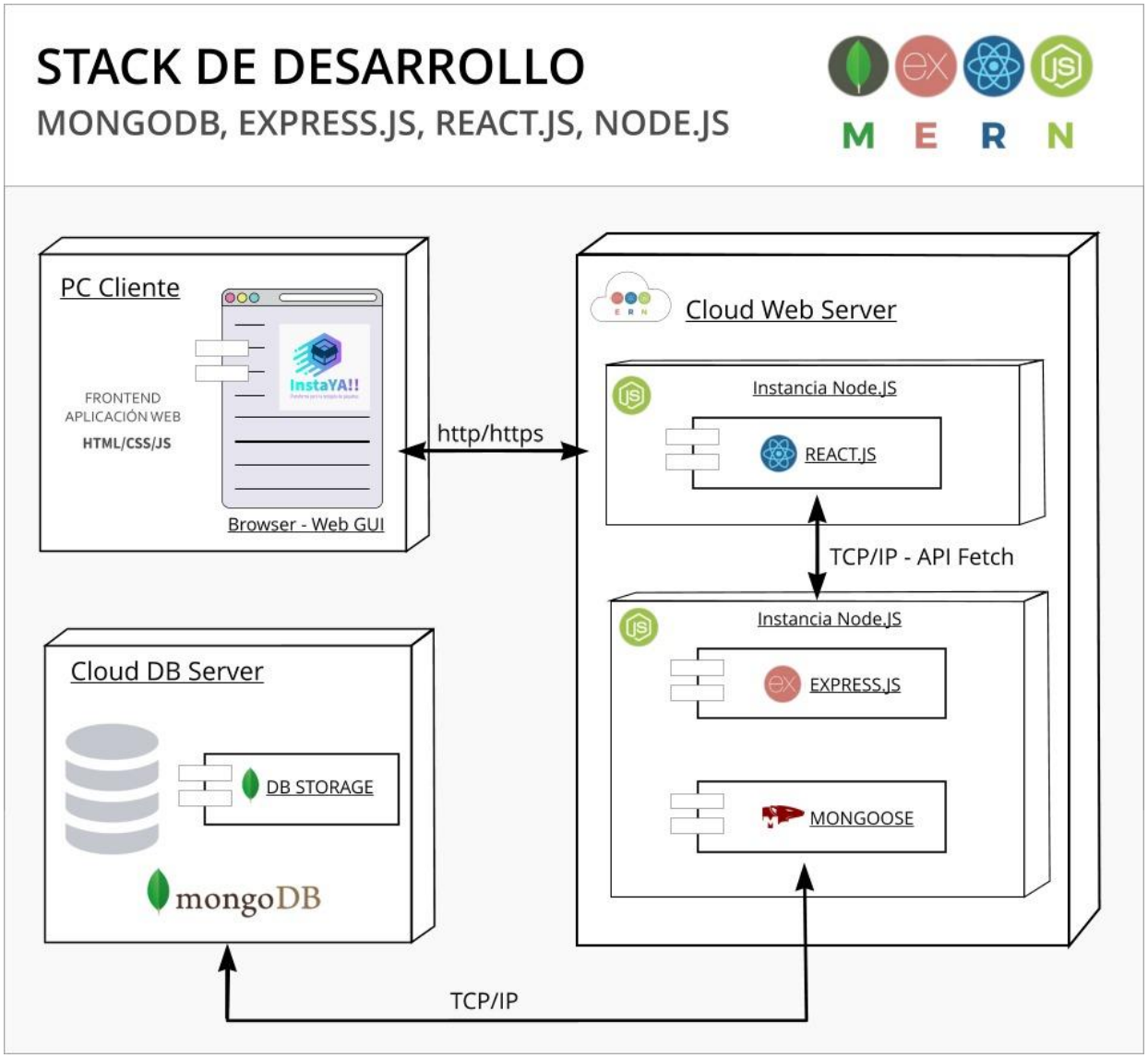
Fernando Andres Palacio
Laura Alejandra Mendoza
Luis Darío García
Lindy Johana Vásquez
Manuel Eduardo Ortega

Descripción de roles del equipo

Rol	Integrante	Descripción	Tareas
PRODUCT OWNER	Fernando Palacio	Lidera procesos y revisión de producto	-Trazar objetivos - Características y creación de Backlog - Creación de historias de usuario - Gestión Backlogs -Supervisión etapas de desarrollo -Revisión de pruebas -Documentación
DISEÑADOR UX/UI	Lindy Vásquez	Desarrolladora web	-Desarrollo de mockups -implementación de vistas en HTML
PROGRAMADOR FRONTEND	Luis Darío García	Desarrollador web	-Codificador html,css,javascript -Responsable de vistas
PROGRAMADOR BACKEND	Laura Alejandra Mendoza	Desarrolladora Backend	-Programación de controles y funciones -Responsable de desarrollo de Apis
PROGRAMADOR BACKEND	Manuel Ortega	Desarrollador Backend	-Responsable Base de datos -Realización de pruebas -integración con el desarrollo en Frontend

Diagrama de despliegue

- Nombre Diagrama de despliegue #01



En el diagrama se tiene una base de datos en MongoDB, en donde encontramos dos capas, una donde está el servidor y el otro el Frontend; del lado del servidor encontramos unas tecnologías, unos Frameworks donde vamos a trabajar para conectarnos a la base de datos, utilizamos Express para la lógica de negocio a través de NodeJS para llegar a la tecnología que nos permite escribir consultas para la base de datos de MongoDB y trabajamos con React para conectar con el Frontend,
Del lado del Cliente trabajamos con HTML5 Y CSS3 Utilizando el framework Bootstrap para manejar diseños modernos y React para la parte funcional.

Definición de artefactos:

User story	Descripción	Estimación (Horas)	Responsable

Start	Lectura y análisis del problema	1 hora	Equipo
Plann1	Planificación Backlogs y Artefactos	8 horas	Fernando Palacio
Plann2	Diseño diagrama de despliegue	4 horas	Alejandra
Plann3	Socialización Diagrama	1 hora	Equipo
meeting	Reunión Equipo	2 horas	Equipo
Plann6	Desarrollo de cronograma	4 horas	Fernando Palacio
Rev1	Check Documentación	2 horas	Equipo

Backlog Sprint 2

User story	Descripción	Estimación (Horas)	Responsable
cod1	Definición de Funciones Lógicas Negocio	8 horas	Alejandra
cod2	Mockup gráfico Interfaz	8 horas	Lindy
cod3	Desarrollo de Formularios en HTML Codificación en React	48 horas	Lindy, Luis
cod4	Desarrollo de diseño gráfico visual en Boostrap	48 horas	Lindy, Luis
Rev2	Revisión Vistas	24 horas	Fernando
meeting2	Reunión de Equipo	2 horas	Equipo
cod6	Solución de errores	24 horas	Equipo

Backlog Sprint 3

User story	Descripción	Estimación (Horas)	Responsable
cod7	Creación de Rutas	8 horas	Aleja, Manuel
cod8	Creación base de datos en Mongo	8 horas	Manuel
cod9	Realización de pruebas en las rutas definidas	8 horas	Aleja
cod10	Realizar CRUD	8 horas	Manuel
cod11	Implementación de la lógica para cada ruta y conexión con la base de datos Validaciones	24 horas	Manuel
cod12	Corrección de bugs	24 horas	Equipo

Backlog Sprint 4

User story	Descripción	Estimación (Horas)	Responsable
Rev3	Despliegue en el servidor HEROKU	12	Fernando
Rev4	Revisión Software	48 horas	Fernando
Final Rev.	Documentación		Equipo

PRODUCT BACKLOG						
User Story ID	User Story	Estimate (size)	Priority	Sprint	Task owner	Estimated effort
START PLANN	Analisis y Documentacion	Small	5	1	Fernando	24
COD1	Desarrollo de Funciones	Large	3	2	Laura	64
COD2	Diseño de Mockups	Small	2	2	Lindy	24
COD3	Diseño de Formularion en HTML	Large	1	2	Luis	48
COD4	Integracion Diseño con Bosstrap	Medium	2	2	Lindy,Luis	36
COD5	Revision Detalles	Medium	3	2	Fernando	24
COD7	Codificacion de rutas en React	Large	1	3	Aleajndra	64
COD8	Creacion de base de datos	Medium	2	3	Manuel	48
COD10	Realizar CRUD	small	3	3	Manuel	24
COD11	Logica de negocio y conexión con base de datos	Medium	2	3	Manuel, Alejandra	48
COD12	Correccion de bugs	Medium	1	3	Equipo	36
COD13	Subir a HEROKU	Small	1	4	Fernando	24

CRONOGRAMA

Todo el desarrollo se va a subir al siguiente repositorio

https://github.com/baronred2020/Sprint4Mintic

Mockup Vistas

Registro

En la parte de creación de usuario, ingresaremos datos personales para nuestras nuevas credenciales y ser un usuario oficial y así poder realizar las ordenes que queramos hacer

Login

En la página de inicio de sesión se colocan las credenciales donde podemos ingresar a nuestra cuenta oficial de **InstaYA**, y así poder realizar la gestión de las órdenes

Registro de Ordenes

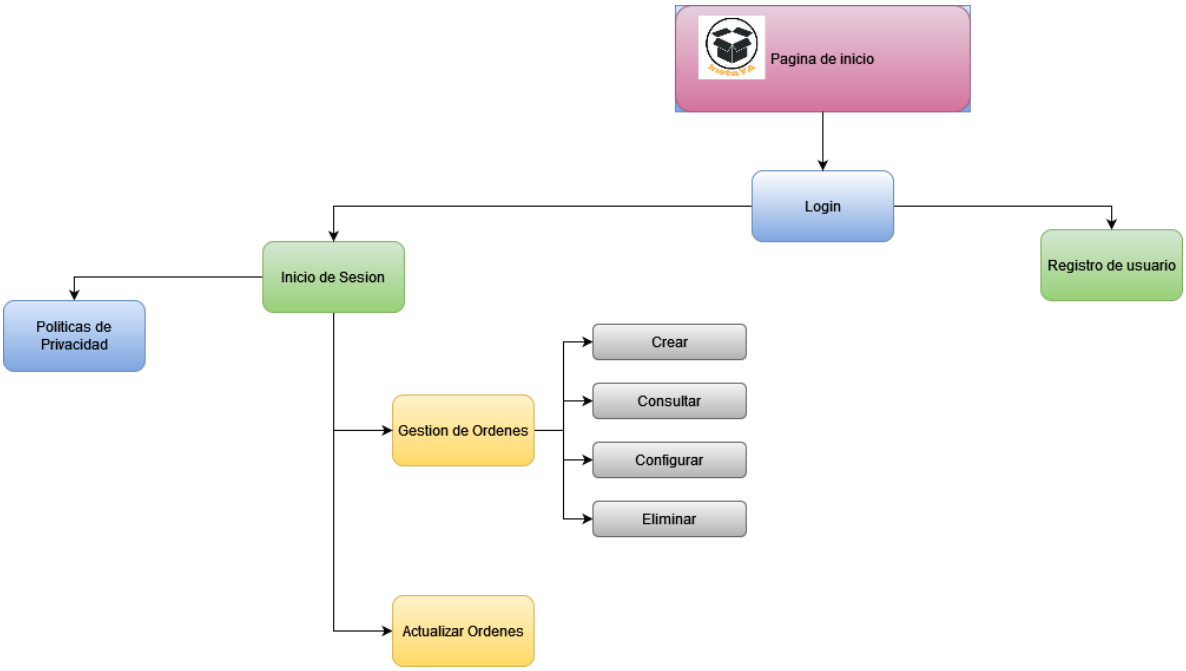
Allí encontraremos la base de datos donde podemos actualizar eliminar agregar y consultar, cada orden que deseemos

Actualización de Ordenes

Así mismo encontraremos la actualización de cada orden

Listado de Ordenes

Mapa de Navegabilidad



Componentes desarrollados (Sprint 3)

Forma de árbol/Esqueleto:

<Header /> Se muestra en todas las vistas

<Body > Wrapper para el contenido de la página

<Router> Se encarga de mostrar los otros componentes según la url

<Welcome/> Se muestra si el usuario no ha ingresado

<LoginWrapper> Wrapper para ingreso y/o registro

<Login/> Formulario para ingresar

<Register/> Formulario para recordar contraseña

</LoginWrapper>

<Content> Se muestra cuando el usuario ya ha ingresado

<Userlist.js/> Lista de usuarios

<AddUser/> Agregar Usuarios

<EditUser/> Editar -Actualizar Usuarios

...

</Content>

</Router>

</ Body>

Formularios desarrollados (Sprint 3)

Formulario	Acción/ruta en backend	Rol	Vista previa
Formulario de Registro	api/Register	Usuario externo	[redacted]
Formulario de Login	api/Login	Usuario interno	Enlace o Imagen
...	...		...
Formulario para Crear pedido	api/AddUser	Usuario Externo	Enlace o Imagen
Formulario para actualizar ordenes	api/EditUser	Usuario Externo	

Formulario para consulta de ordenes	api/UserList	**Usuario Externo**	

Solicitudes a la API (Sprint 3)

Vista	Acción/ruta en backend	Rol	Datos esperados
Validacion Usuario	**api/** `getUserById`	**Usuario**	`return res.json({mensaje: "Usuario no encontrado"})` `res.json({mensaje: "Estas enviando identificacion incorrecta"});`
Login	**api/login**	**Usuario interno**	`return res.json({mensaje: "Usuario no encontrado"})` `return res.json({mensaje: "Ha ingresado correctamente", usuario:{id,nombre}});`
...	**...**		**...**

Register	**api/register**	**Usuario**	`return res.json({mensaje: "Ya existe un usuario con ese correo"}); }else if(!nombre || !usuarioo || !correo || !contrasena){ return res.json({mensaje: "Falta el nombre/usuario/correo/contrasena"});`
Controladores	**api/** `getUsers`	**Usuario**	`try { const users = await User.find(); res.json(users); } catch (error) { res.status(500).json({message: error.message}); }`
Controladores	**Api/** `saveUser`	**Usuario**	`try { const insertedUser = await user.save(); res.status(201).json(insertedUser);`

			`} catch (error) { res.status(400).json({ message: error.message}); }`
Controladores	**Api/** `updateUser`	**Usuario**	`try { const updateduser = await User.updateOne({_id:req.params.id}, {$set: req.body}); res.status(200).json(updateduser); } catch (error) { res.status(400).json({ message: error.message}); }`
Controladores	**Api/** `deleteUser`	**Usuario**	`try { const deleteduser = await User.deleteOne({_id:req.params.id}); res.status(200).json(deleteduser); } catch (error) {`

			`` res.status(400).json({ message: error.message}); } ```