**Documentación Proyecto**

**Plataforma de Gestión de Empleados BRF**

**NRC:** 641-41-9

**Equipo No.** 9

**Integrantes:**

* *Franklin Villegas (Product Owner).*
* *Massimiliano Riposati (Scrum Master).*
* *German Tercero Robles Monterosa (Development Team).*
* *Héctor Fabián Pulido Martínez (Development Team).*
* *Camilo Ernesto Vásquez Bolívar (Development Team).*

**Casos de uso**

* Nombre Caso de uso 1

Diagrama, Gráfico de burbujas

Descripción generada automáticamente

<https://github.com/maxriposati/equipo9/blob/main/1_Sprint-Entregables/CasosDeUso_Equipo9.pdf>

**Diagrama de Clases**

Diagrama

Descripción generada automáticamente

<https://github.com/maxriposati/equipo9/blob/main/1_Sprint-Entregables/Diagrama%20de%20Clases.pdf>

Diagrama

Descripción generada automáticamente**Mapa de Navegabilidad**

<https://github.com/maxriposati/equipo9/blob/main/2_Sprint-Entregables/Map.pdf>

**Mockups**

* Nombre Mockup 1: Home

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

* Nombre Mockup 2: Contáctanos

Interfaz de usuario gráfica, Texto, Correo electrónico

Descripción generada automáticamente

* Nombre Mockup 3: Visión

Interfaz de usuario gráfica, Texto, Correo electrónico

Descripción generada automáticamente

* Nombre Mockup 4: Misión

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

* Nombre Mockup 5: Servicios

Interfaz de usuario gráfica, Texto, Correo electrónico

Descripción generada automáticamente

* Nombre Mockup 6: Login

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

* Nombre Mockup 7: Recuperar\_Pass

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

* Nombre Mockup 8: Políticas de Privacidad

Interfaz de usuario gráfica, Texto, Correo electrónico

Descripción generada automáticamente

* Nombre Mockup 9: Main\_Admin

Tabla

Descripción generada automáticamente

* Nombre Mockup 10: Main\_Admin\_Alerta Eliminar

Interfaz de usuario gráfica

Descripción generada automáticamente

* Nombre Mockup 11: Crear usuario

Interfaz de usuario gráfica

Descripción generada automáticamente

* Diagrama

  Descripción generada automáticamenteNombre Mockup 12: Crear usuario\_Alerta Crear
* Nombre Mockup 13: Editar Registro

Interfaz de usuario gráfica

Descripción generada automáticamente

* Nombre Mockup 14: Editar Registro\_ Alerta editar

Diagrama

Descripción generada automáticamente

* Nombre Mockup 15: Evaluación de desempeño

Interfaz de usuario gráfica

Descripción generada automáticamente

* Nombre Mockup 16: Retroalimentación usuario

Interfaz de usuario gráfica

Descripción generada automáticamente

**Métodos Manejadores del Backend**

* index()
  + Ruta: ‘/’, ‘/home’
  + Métodos HTTP permitidos: GET
  + Lógica algorítmica: Asignación de valores de session["usuario"] = "unknown" y session["rol"] = 0 y renderizado de la página index.html.
* mision()
* Ruta: ‘/mision’
  + Métodos HTTP permitidos: GET
  + Lógica algorítmica: renderizado de la página mision.html.
* vision()
* Ruta: ‘/vision’
  + Métodos HTTP permitidos: GET
  + Lógica algorítmica: renderizado de la página vision.html.
* servicios()
* Ruta: ‘/servicios’
  + Métodos HTTP permitidos: GET
  + Lógica algorítmica: renderizado de la página servicios.html.
* politicas()
* Ruta: ‘/politicas’
  + Métodos HTTP permitidos: GET
  + Lógica algorítmica: renderizado de la página politicas.html.
* contacto()
* Ruta: ‘/contacto’
  + Métodos HTTP permitidos: GET,POST
  + Lógica algorítmica: Con el método GET se hará renderizado de la página contacto.html. Con el método POST se realizará un envío de datos capturados por el usuario y llevados a un correo administrativo.
* recovery(correo,doc)
* Ruta: ‘/recovery/<string:correo>/<string:doc>’
  + Métodos HTTP permitidos: GET, POST
  + Lógica algorítmica: Con el método GET se hará renderizado de la página recovery.html. Con el método POST se realizará una consulta a la tabla usuarios de la base de datos para verificar al usuario, se solicita la nueva password y verificación para después realizar la actualización de la clave en la tabla usuarios.
* recuperar()
* Ruta: ‘/recuperarpass’
  + Métodos HTTP permitidos: GET, POST
  + Lógica algorítmica: Con el método GET se hará renderizado de la página recuperarpass.html. Con el método POST se enviará un correo electrónico con link de recuperación al usuario que se capture en el textfield.
* administrador()
* Ruta: ‘/mainadmin’
  + Métodos HTTP permitidos: GET, POST
  + Lógica algorítmica: Con el método GET se hará renderizado de la página mainadmin.html. Con el método POST se realizará una consulta a la tabla usuarios de la base de datos para verificar el rol de cada usuario, y solo de acuerdo con el rol (superadministrador o administrador) se pueden obtener en pantalla un primer listado de usuarios. Ejemplo: Si quien ingresa es un administrador este solo puede visualizar empleados para gestionarlos. El usuario normal o empleado tendría otra ruta con sus respectivos métodos.
* crear()
* Ruta: ‘/mainadmin/crear’
  + Métodos HTTP permitidos: GET, POST
  + Lógica algorítmica: Con el método GET se hará renderizado de la página crearusuario.html. Con el método POST se agregará la información de cada usuario obtenida del formulario y la enviará a las tablas persons, users y contracts de la base de datos, realizando las validaciones pertinentes en javascript que no permiten la captura de datos erróneos y validando que el usuario no exista.
* editar(id)
* Ruta: ‘/mainadmin/editar/<int:id>’
  + Métodos HTTP permitidos: GET, POST
  + Lógica algorítmica: Con el método GET se hará renderizado de la página editarusuario.html. Con el método POST se actualizará la información de cada usuario obtenida del formulario y la enviará a las tablas persons, users y contracts de la base de datos, realizando las validaciones pertinentes en javascript que no permiten la captura de datos erróneos.
* eliminar(id)
* Ruta: ‘/mainadmin/eliminar/<int:id>’
  + Métodos HTTP permitidos: GET
  + Lógica algorítmica: Con el método GET se redirecciona a la página mainadmin.html y se eliminan los datos de cada usuario en las tablas persons, users y contracts de la base de datos.
* evaluar(id)
* Ruta: ‘/mainadmin/evaluar/<int:id>’
  + Métodos HTTP permitidos: GET
  + Lógica algorítmica: Con el método GET se hará renderizado a la página evaluación.html
* usuario(id)
* Ruta: ‘/mainusuario/<int:id>’
  + Métodos HTTP permitidos: GET
  + Lógica algorítmica: Con el método GET se hará renderizado a la página mainusuario.html la cual permite traer un formulario al que acceden únicamente los usuarios normales o empleados, los cuales pueden visualizar sus datos de contratación y retroalimentación del desempeño. Los campos de dicho formulario no pueden ser modificados por el manejo de las SESIONES en los roles configurados.
* login()
* Ruta: ‘/login’
  + Métodos HTTP permitidos: GET, POST
  + Lógica algorítmica: Con el método GET se hará renderizado a la página login.html y con el método POST realizará consulta en la base de datos para verificar la existencia del usuario verificando la contraseña que se encuentra cifrada en la base de datos; y si existe renderizará las páginas mainusuario.html o mainadmin.html dependiendo del rol o SESION iniciada.
* logout()
* Ruta: ‘/logout’
  + Métodos HTTP permitidos: GET
  + Lógica algorítmica: Con el método GET se sale de cualquier página o formulario, se cierra la SESIÓN y se redireccionará a la página home.html

**Base de Datos**

Diagrama

Descripción generada automáticamente

*La tabla* ***persons*** *guarda la información básica de todos los usuarios y se relaciona con las otras tres tablas a partir del campo Id. Por otra parte, la tabla* ***users*** *almacena los campos relacionados con la autenticación de todos los usuarios y es donde se cifran las contraseñas; así mismo la tabla* ***contracts*** *archiva la información inherente al contrato y finalmente la tabla* ***performances*** *guarda los datos relacionados con el desempeño de los colaboradores.*

**Prácticas de Programación Segura**

*Se emplearon para la ejecución de proyecto las siguientes prácticas de programación segura:*

* *Método de Autenticación a partir de Sesiones: El sistema valida los requerimientos en base de datos dependiendo de la SESION iniciada por cada rol del usuario.*
* *Cifrado de Contraseñas basadas en funciones de derivación de claves criptográficas (PBKDF2): Resistente a ataques de diccionario y tablas rainbow.*
* *Prepared Statements: Impide la inyección de código y protege la fuente de datos.*

**Librerías empleadas en el proyecto:**

cachelib==0.4.1

cachetools==4.2.4

certifi==2021.10.8

charset-normalizer==2.0.7

click==8.0.1

colorama==0.4.4

cssselect==1.1.0

cssutils==2.3.0

Flask==2.0.1

Flask-Session==0.4.0

Flask-WTF==0.15.1

greenlet==1.1.2

idna==3.3

importlib-metadata==4.8.1

itsdangerous==2.0.1

Jinja2==3.0.1

keyring==23.2.1

lxml==4.6.3

MarkupSafe==2.0.1

numpy==1.21.2

pandas==1.3.3

premailer==3.10.0

python-dateutil==2.8.2

pytz==2021.3

pywin32-ctypes==0.2.0

requests==2.26.0

six==1.16.0

sodapy==2.1.0

urllib3==1.26.7

validate-email==1.3

Werkzeug==2.0.1

WTForms==2.3.3

yagmail==0.14.260

zipp==3.6.0