

-

-

- **– Proyecto Barbería –**

- **Taller de diseño y desarrollo de soluciones**

NOMBRE: Luis Almonacid, Alban Fernández, Franco Lleuful

CARRERA: Ingeniería informática

ASIGNATURA: Taller de Diseño y Desarrollo de Soluciones

PROFESOR: Ricardo Roberto Hidalgo Hidalgo

FECHA: 14/12/2022

1 Introducción

En el presente informe se darán a conocer todos los detalles del proyecto “La navaja del Barbero”. Aquí podrán observar los planes de pruebas que se diseñaron en las etapas posteriores, la documentación del proyecto y los distintos tipos de planes de mantención que se le harán al proyecto en determinadas situaciones.

2 Objetivo

| | |
|--|--|
| 2.1.5.- Implementa plan de pruebas. | 1.- Efectúa las pruebas funcionales, de integración y de sistemas. |
| | 2.- Las pruebas realizadas corresponden a los criterios de aceptación. |
| 2.1.6.- Confecciona la documentación del proyecto. | 3.- Genera la documentación de la solución, incluyendo los manuales respectivos (instalación, usuario). |
| | 4.- Entrega en el ambiente de aprendizaje el respaldo del código fuente, el ejecutable y el script de la base de datos. |
| 2.1.7.- Propone plan de mantención en el tiempo. | 5.- Define la forma en que se recibirán las peticiones de modificaciones ulteriores. |
| | 6.- Define un protocolo para mantenciones correctivas, preventivas, adaptativas y perfectivas. procedimientos almacenados. |
| 2.1.8.- Expone resultado y características de la aplicación frente al cliente. | 7.- Presenta los resultados de la solución. |
| 2.1.9.- Considerando información, variables y criterios establecidos. | 8.- Responde a las preguntas y consultas en la presentación del proyecto. |

3 Desarrollo

3.1 Criterios de aceptación

| ID | Rol | Característica / Funcionalidad | Criterios de aceptación |
|-------|------------|---|---|
| RF01 | Cliente | Quiero que el sistema me permita realizar reservaciones. | <ul style="list-style-type: none"> ● Presenta selección del servicio, día y hora ● Deshabilita los días y horas no disponibles ● Formulario permite ingresar los datos de reservación (Rut, nombre, email) ● Muestra un resumen (servicios, fecha, hora, total a pagar, etc.) |
| RF02 | Secretario | Quiero que el sitio web tenga un sistema de log in. | <ul style="list-style-type: none"> ● Formulario permite ingresar con usuario y contraseña ● El sistema permite listar, modificar y eliminar reservaciones ● El sistema permite listar, crear, modificar y eliminar usuarios |
| RNF03 | Secretario | Quiero visualizar en el sistema los servicios que ofrecemos | <ul style="list-style-type: none"> ● Muestra descripción y costo de cada servicio |
| RF04 | Cliente | Quiero que el sistema me permita elegir el barbero. | <ul style="list-style-type: none"> ● Muestra nombre de los barberos disponibles ● Muestra el servicio que ofrece cada barbero |
| RNF05 | Cliente | Necesito visualizar en el sistema la ubicación de la barbería. | <ul style="list-style-type: none"> ● Presenta Google Maps señalando la ubicación |
| RNF06 | Secretario | Quiero que el sistema tenga un apartado con información acerca de la barbería (una breve historia del negocio, ¿Quiénes somos?) | <ul style="list-style-type: none"> ● Muestra historia y fotos de la barbería |

3.2 Pruebas funcionales

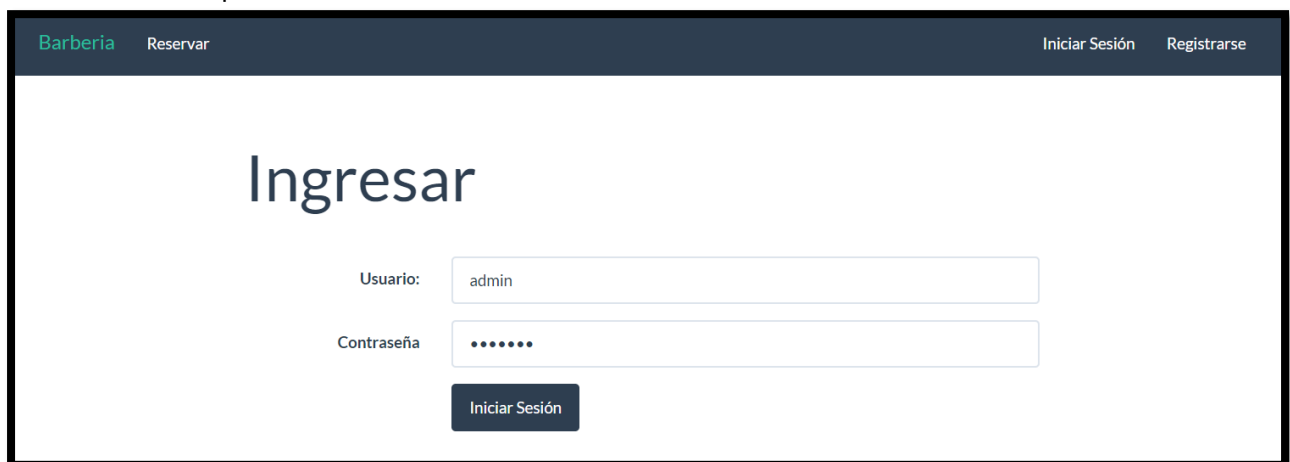
Las pruebas funcionales se centran en los requisitos empresariales de una aplicación. Solo verifican el resultado de una acción y no comprueban los estados intermedios del sistema al realizar dicha acción.

Las ventajas que nos ofrecen las pruebas funcionales son:

- Se asegura de que el sitio web/aplicación esté libre de defectos.
- Garantiza el comportamiento esperado de todas las funcionalidades.
- Garantiza que la arquitectura sea correcta con la seguridad necesaria.
- Mejora la calidad y las funcionalidades generales.
- Minimiza los riesgos empresariales asociados con el sitio web/aplicación.

Con relación a nuestro proyecto, se aplicará este tipo de pruebas validando las entradas y salidas de cada funcionalidad del sistema, como el registro de usuarios, barberos, servicios, horarios y reservaciones, como también el inicio de sesión y cambio de contraseña.

El usuario admin puede iniciar sesión en el sitio web



Acá se encuentra el perfil del administrador



Aquí puede registrar nuevos servicios

[eservar](#) [Agregar Barbero](#) [Agregar Servicio](#) [Agregar Horario](#) [Mi cuenta](#)

Registrar

Nombre

Precio

Registrar

Acá puede ver la lista de servicios creados

Servicios

| ID | Nombre | Precio |
|---------------------|-----------------------|--------|
| # 2 | Corte de pelo Classic | 6000 |

Aquí puede registrar nuevos barberos

[Agregar Barbero](#) [Agregar Servicio](#) [Agregar Horario](#) [Mi cuenta](#)

Registrar

Nombre

RUT

Email

Telefono

Servicio

Registrar

Acá se puede registrar nuevas horas de atención

[Agregar Barbero](#) [Agregar Servicio](#) [Agregar Horario](#) [Mi cuenta](#)

Registrar

Desde

Hasta

Registrar

Aquí puede ver la lista de horarios

[Agregar Servicio](#) [Agregar Horario](#) [Mi cuenta](#)

Horarios

| ID | Desde | Hasta |
|----------------------|-------|-------|
| # 13 | 17:30 | 18:15 |
| # 12 | 16:45 | 17:25 |
| # 11 | 16:00 | 16:40 |
| # 10 | 15:15 | 15:55 |
| # 9 | 14:30 | 15:10 |
| # 8 | 12:45 | 13:25 |
| # 7 | 12:00 | 12:40 |
| # 6 | 11:15 | 11:55 |
| # 5 | 10:30 | 11:10 |
| # 4 | 09:45 | 10:25 |
| # 3 | 09:00 | 09:40 |

Acá el usuario admin puede ver las reservaciones de todos los usuarios

Mi Cuenta


Cambiar Contraseña

Cambiar Email y datos Personales

Mis Reservas

Listar Reservas

Reservaciones

| ID | Barbero | Día | Horario | Usuario | Opciones |
|--------------------|------------------------------|------------|---------------------|---------|---|
| #2 | Luis Almonacid - 22755863-6 | 15/12/2022 | 09:45:00 - 10:25:00 | Alexdev | Editar  Eliminar  |

Acá el usuario admin puede ver la lista de usuarios creados

Mi Cuenta

Cambiar Contraseña

Cambiar Email y datos Personales

Mis Reservas

Listar Reservas

Usuarios

| ID | Nombre | Apellido | RUT | Sexo | Telefono | Email | Usuario |
|--------------------|-----------------|-----------------|------------|------|-----------|-----------------|---------|
| #3 | Alban Alexander | Fernández Palma | 20224978-7 | MAS | 987654123 | Alban@gmail.com | Alexdev |
| #2 | Jose | ulloa | 123456789 | MAS | 912345671 | ulloa@gmail.com | admin |

Aquí se encuentra el registro de un usuario normal

Registro

| | |
|------------------------------|--|
| Nombre | <input type="text" value="Alban Alexander"/> |
| Apellido | <input type="text" value="Fernández Palm"/> |
| RUT | <input type="text" value="20224978-7"/> |
| Sexo | <input type="text" value="Masculino"/> ▼ |
| Telefono | <input type="text" value="987654123"/> |
| Email | <input type="text" value="Alban@gmail.com"/> |
| Usuario | <input type="text" value="Alexdev"/> |
| Contraseña | <input type="password" value="....."/> |
| Contraseña (confirmación) | <input type="password" value="....."/> |

Registrarse

Acá está el login del usuario normal

Ingresar

| | |
|------------|--|
| Usuario: | <input type="text" value="Alexdev"/> |
| Contraseña | <input type="password" value="....."/> |

Iniciar Sesión

Acá el usuario normal puede realizar reservaciones

Barberia

Reservar

Mi cuenta

Cerrar Sesión

Reservar

Barbero

Luis Almonacid| - 22755863-6

▼

Dia

15-12-2022

📅

Horario

09:45:00 - 10:25:00

▼

Guardar

Aquí el usuario normal puede ver sus reservaciones

| Mi Cuenta | | | | | Mis Reservas | | | | |
|----------------------------------|------------------------------|------------|---------------------|---------------------|--------------|--|--|--|--|
| Cambiar Contraseña | | | | | | | | | |
| Cambiar Email y datos Personales | | | | | | | | | |
| Mis Reservas | | | | | | | | | |
| ID | Barbero | Dia | Horario | Opciones | | | | | |
| #2 | Luis Almonacid - 22755863-6 | 15/12/2022 | 09:45:00 - 10:25:00 | Editar ✎ Eliminar 🗑 | | | | | |

3.3 Pruebas integrales

Las pruebas integrales o de integración verifican que los distintos módulos o servicios utilizados por tu aplicación funcionan bien en conjunto. Por ejemplo, se puede probar la interacción con la base de datos o asegurarse de que los microservicios funcionan bien en conjunto y según lo esperado. Estos tipos de pruebas son más costosos de ejecutar, ya que requieren que varias partes de la aplicación estén en marcha.

Las ventajas de estas son:

- Se asegura de que todos los módulos de aplicación estén bien integrados y funcionen juntos según lo esperado.
- Detecta problemas y conflictos interconectados para resolverlos antes de crear un gran problema.
- Valida la funcionalidad, fiabilidad y estabilidad entre diferentes módulos.
- Detecta excepciones ignoradas para mejorar la calidad del código.
- Admite la canalización de CI/CD.

Las pruebas integrales las aplicaremos probando la interacción del sistema con la base de datos. Es decir, que los datos se agreguen y actualicen correctamente

Aquí django especifica que se creará la base de datos 'db.sqlite3' en la carpeta del proyecto. La base de datos se crea cuando nosotros hacemos la migración de los modelos, los cuales se transformarán en tablas.

```
# Database
# https://docs.djangoproject.com/en/4.1/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
```

Aquí se encuentran las tablas de las bases de datos, algunas de estas nos las provee el framework de django.

| Nombre | Tipo | Esquema |
|--|------|---|
| ▼ Tablas (15) | | |
| auth_group | | CREATE TABLE "auth_group" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "name" varchar(150) NOT NULL UNIQUE) |
| auth_group_permissions | | CREATE TABLE "auth_group_permissions" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "group_id" integer NOT NULL REFERENCES "auth_group" ("id") DEFERRABLE INITIALLY DEFERRED, "permission_id" integer NOT NULL REFERENCES "auth_group_permissions" ("group_id", "permission_id")) |
| auth_permission | | CREATE TABLE "auth_permission" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "content_type_id" integer NOT NULL REFERENCES "django_content_type" ("id") DEFERRABLE INITIALLY DEFERRED, "codename" varchar(255) NOT NULL, "name" varchar(255) NOT NULL, "applied" datetime NOT NULL) |
| barberos_barbero | | CREATE TABLE "barberos_barbero" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "nombre" varchar(200) NOT NULL, "rut" varchar(15) NOT NULL, "email" varchar(254) NOT NULL, "telefono" varchar(20) NOT NULL) |
| barberos_horario | | CREATE TABLE "barberos_horario" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "desde" time NOT NULL, "hasta" time NOT NULL) |
| barberos_servicio | | CREATE TABLE "barberos_servicio" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "nombre" varchar(200) NOT NULL UNIQUE, "precio" integer NOT NULL) |
| cuentas_usuario | | CREATE TABLE "cuentas_usuario" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "password" varchar(128) NOT NULL, "last_login" datetime NULL, "is_superuser" bool NOT NULL, "username" varchar(150) NOT NULL, "email" varchar(254) NOT NULL, "is_staff" bool NOT NULL, "is_active" bool NOT NULL, "date_joined" datetime NOT NULL) |
| cuentas_usuario_groups | | CREATE TABLE "cuentas_usuario_groups" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "usuario_id" bigint NOT NULL REFERENCES "cuentas_usuario" ("id") DEFERRABLE INITIALLY DEFERRED, "group_id" integer NOT NULL REFERENCES "auth_group" ("id") DEFERRABLE INITIALLY DEFERRED) |
| cuentas_usuario_user_permissions | | CREATE TABLE "cuentas_usuario_user_permissions" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "usuario_id" bigint NOT NULL REFERENCES "cuentas_usuario" ("id") DEFERRABLE INITIALLY DEFERRED, "permission_id" integer NOT NULL REFERENCES "auth_permission" ("id") DEFERRABLE INITIALLY DEFERRED) |
| django_admin_log | | CREATE TABLE "django_admin_log" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "object_id" text NULL, "object_repr" varchar(200) NOT NULL, "action_flag" smallint unsigned NOT NULL CHECK (0 <= action_flag < 32), "change_message" text NULL) |
| django_content_type | | CREATE TABLE "django_content_type" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "app_label" varchar(100) NOT NULL, "model" varchar(100) NOT NULL) |
| django_migrations | | CREATE TABLE "django_migrations" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "app" varchar(255) NOT NULL, "name" varchar(255) NOT NULL, "applied" datetime NOT NULL) |
| django_session | | CREATE TABLE "django_session" ("session_key" varchar(40) NOT NULL PRIMARY KEY, "session_data" text NOT NULL, "expire_date" datetime NOT NULL) |
| reservas_reserva | | CREATE TABLE "reservas_reserva" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "dia" date NOT NULL, "barbero_id" bigint NOT NULL REFERENCES "barberos_barbero" ("id") DEFERRABLE INITIALLY DEFERRED, "horario_id" integer NOT NULL REFERENCES "barberos_horario" ("id") DEFERRABLE INITIALLY DEFERRED, "usuario_id" bigint NOT NULL REFERENCES "cuentas_usuario" ("id") DEFERRABLE INITIALLY DEFERRED) |
| sqlite_sequence | | CREATE TABLE sqlite_sequence(name,seq) |
| ▼ Índices (21) | | |
| auth_group_permissions_group_id_b120cbf9 | | CREATE INDEX "auth_group_permissions_group_id_b120cbf9" ON "auth_group_permissions" ("group_id") |
| auth_group_permissions_group_id_permission_id_0cd325b0 | | CREATE UNIQUE INDEX "auth_group_permissions_group_id_permission_id_0cd325b0" ON "auth_group_permissions" ("group_id", "permission_id") |
| auth_group_permissions_permission_id_84c5c92e | | CREATE INDEX "auth_group_permissions_permission_id_84c5c92e" ON "auth_group_permissions" ("permission_id") |
| auth_permission_content_type_id_codename_01ab375a | | CREATE UNIQUE INDEX "auth_permission_content_type_id_codename_01ab375a" ON "auth_permission" ("content_type_id", "codename") |
| barberos_barbero_rut_servicio_id_7aa29351 | | CREATE UNIQUE INDEX "barberos_barbero_rut_servicio_id_7aa29351" ON "barberos_barbero" ("rut", "servicio_id") |
| barberos_barbero_servicio_id_2864dcb5 | | CREATE INDEX "barberos_barbero_servicio_id_2864dcb5" ON "barberos_barbero" ("servicio_id") |
| cuentas_usuario_groups_group_id_e6725cc8 | | CREATE INDEX "cuentas_usuario_groups_group_id_e6725cc8" ON "cuentas_usuario_groups" ("group_id") |
| cuentas_usuario_groups_usuario_id_251ff583 | | CREATE INDEX "cuentas_usuario_groups_usuario_id_251ff583" ON "cuentas_usuario_groups" ("usuario_id") |
| cuentas_usuario_groups_usuario_id_group_id_5ef6c73 | | CREATE UNIQUE INDEX "cuentas_usuario_groups_usuario_id_group_id_5ef6c73" ON "cuentas_usuario_groups" ("usuario_id", "group_id") |
| cuentas_usuario_user_permissions_permission_id_1753ad79 | | CREATE INDEX "cuentas_usuario_user_permissions_permission_id_1753ad79" ON "cuentas_usuario_user_permissions" ("permission_id") |
| cuentas_usuario_user_permissions_usuario_id_5a9718f1 | | CREATE INDEX "cuentas_usuario_user_permissions_usuario_id_5a9718f1" ON "cuentas_usuario_user_permissions" ("usuario_id") |
| cuentas_usuario_user_permissions_usuario_id_permission_id_26c0acea | | CREATE UNIQUE INDEX "cuentas_usuario_user_permissions_usuario_id_permission_id_26c0acea" ON "cuentas_usuario_user_permissions" ("usuario_id", "permission_id") |
| django_admin_log_content_type_id_c4bce8eb | | CREATE INDEX "django_admin_log_content_type_id_c4bce8eb" ON "django_admin_log" ("content_type_id") |
| django_admin_log_user_id_c564eb69 | | CREATE INDEX "django_admin_log_user_id_c564eb69" ON "django_admin_log" ("user_id") |
| django_content_type_app_label_model_76bd3d3b | | CREATE UNIQUE INDEX "django_content_type_app_label_model_76bd3d3b" ON "django_content_type" ("app_label", "model") |
| django_session_expire_date_a5c62663 | | CREATE INDEX "django_session_expire_date_a5c62663" ON "django_session" ("expire_date") |
| reservas_reserva_barbero_id_dia_horario_id_4dce3291 | | CREATE UNIQUE INDEX "reservas_reserva_barbero_id_dia_horario_id_4dce3291" ON "reservas_reserva" ("barbero_id", "dia", "horario_id") |
| reservas_reserva_horario_id_29466197 | | CREATE INDEX "reservas_reserva_horario_id_29466197" ON "reservas_reserva" ("horario_id") |
| reservas_reserva_usuario_id_531da18c | | CREATE INDEX "reservas_reserva_usuario_id_531da18c" ON "reservas_reserva" ("usuario_id") |
| Vistas (0) | | |
| Disparadores (0) | | |

3.4 Pruebas de sistema

Las pruebas de sistema son funciones que se ven en el sistema y que están planteadas dentro del código.

Aquí se encuentran los modelos, dentro de estos se realizan las validaciones de cada campo. Estas validaciones, se hacen a partir de funciones y parámetros que sirven para limitar el número de caracteres, especificar un tipo de dato, configurar que un conjunto de datos sean únicos y que no existan valores nulos, entre otros. También, a partir de estos se crean las tablas y los formularios que luego podemos ver con las vistas y los templates

```
def validarRut(rut):
    rut = rut.upper()
    rut = rut.replace("-", "")
    rut = rut.replace(".", "")
    aux = rut[:-1]
    dv = rut[-1:]
    error = False

    if len(rut) != 9:
        error = True
    if aux.isalpha() == True:
        error = True
    if dv.isalpha() == True:
        if dv != "K":
            error = True

    if error:
        raise ValidationError('Ingrese un rut valido. Ej: 22755865-K')

class Barbero(models.Model):
    nombre = models.CharField(verbose_name="Nombre", max_length=200)
    rut = models.CharField(verbose_name="RUT", max_length=15, validators=[validarRut], help_text="Ingrese un rut valido. Ej: 22755865-K")
    email = models.EmailField(verbose_name="Email")
    telefono_regex = RegexValidator(
        regex=r'^\+71?\d{9,15}$',
        message="El número debe estar en este formato: 933789228")

    telefono = models.CharField(verbose_name="Telefono",
                                validators=[telefono_regex],
                                max_length=17, null=False)
    servicio = models.ForeignKey(Servicio,
                                on_delete=models.CASCADE,
                                related_name='barbero')

    class Meta:
        unique_together = ('rut', 'servicio')

    def __str__(self):
        return f'{self.nombre} - {self.rut}'
```

```
class Horario(models.Model):
    desde = models.TimeField(auto_now=False, auto_now_add=False, help_text="La hora debe estar en este formato: 14:00 o 14:00:00")
    hasta = models.TimeField(auto_now=False, auto_now_add=False, help_text="La hora debe estar en este formato: 15:30 o 15:30:00")

    def __str__(self):
        return f'{self.desde} - {self.hasta}'

class Servicio(models.Model):
    nombre = models.CharField(verbose_name="Nombre", max_length=200, unique=True,)
    precio = models.IntegerField(verbose_name="Precio")

    def __str__(self):
        return f'{self.nombre} - {self.precio}'
```

```

10 def validarRut(rut):
11     rut = rut.upper()
12     rut = rut.replace("-", "")
13     rut = rut.replace(".", "")
14     aux = rut[:-1]
15     dv = rut[-1:]
16     error = False
17
18     if len(rut) != 9:
19         error = True
20     if aux.isalpha() == True:
21         error = True
22     if dv.isalpha() == True:
23         if dv != "K":
24             error = True
25
26     if error:
27         raise ValidationError('Ingrese un rut valido. Ej: 22755865-K')
28
29 # Modelo de la tabla usuario
30 class Usuario(AbstractBaseUser, PermissionsMixin):
31     username = models.CharField(
32         'Usuario', max_length=30, unique=True, validators=[
33             validators.RegexValidator(
34                 re.compile('^[a-zA-Z0-9@+-.]+$'),
35                 'Ingrese un nombre de usuario valido. '
36                 'Este valor debe contener solo letras, números '
37                 'y los caracteres: @/./+/-/_.'
38             ),
39             'invalid'
40         ], help_text='Un nombre corto que será usado'+
41         ' para identificarlo de forma única en la plataforma.'
42     )
43     first_name = models.CharField('Nombre', max_length=150)
44     last_name = models.CharField('Apellido', max_length=150)
45     email = models.EmailField('Email', unique=True)
46     is_staff = models.BooleanField('Equipo', default=False)
47     is_active = models.BooleanField('Activo', default=True)
48     date_joined = models.DateTimeField('Fecha de Ingreso', auto_now_add=True)
49
50     SEXO = (
51         ("MAS", "Masculino"),
52         ("FEM", "Femenino")
53     )

```

```

sexo = models.CharField(max_length=9, choices=SEXO,)

phone_regex = RegexValidator(
    regex=r'^\+?1?\d{9,15}$',
    message="El número debe estar en este formato: 933789228.")

telefono = models.CharField(verbose_name="Telefono",
                             validators=[phone_regex],
                             max_length=9, null=False, help_text='El número debe estar en este formato: 933789228')

rut = models.CharField(verbose_name="RUT", max_length=10, validators=[validarRut], unique=True, help_text='Ingrese un rut valido. Ej: 227558')

USERNAME_FIELD = 'username'
REQUIRED_FIELDS = ['first_name', 'last_name', 'email', 'sexo', 'telefono', 'rut']

objects = UserManager()

class Meta:
    verbose_name = 'Usuario'
    verbose_name_plural = 'Usuarios'

def __str__(self):
    return self.username

```

```
# Create your models here.

def validar_dia(value):
    today = date.today()
    weekday = date.fromisoformat(f'{value}').weekday()

    if value < today:
        raise ValidationError('No es posible elegir una fecha tardía.')
    if (weekday == 5) or (weekday == 6):
        raise ValidationError('Elija un día laborable de la semana.')

class Reserva(models.Model):
    barbero = ForeignKey(Barbero, on_delete=models.CASCADE, related_name='reserva')
    dia = models.DateField(help_text="Selecciona un día en el calendario", validators=[validar_dia])
    horario = models.ForeignKey(Horario, on_delete=models.CASCADE, related_name='reserva')
    usuario = ForeignKey(Usuario, on_delete=models.CASCADE, related_name='reserva')

    class Meta:
        unique_together = ('barbero', 'dia', 'horario')

    def __str__(self):
        return f'{self.barbero} - {self.dia} - {self.horario} - {self.usuario}'
```

3.5 Mantenimiento preventivo

El mantenimiento preventivo de software está mirando hacia el futuro para que su software pueda seguir funcionando como se desee durante el mayor tiempo posible.

Esto incluye realizar los cambios necesarios, actualizaciones, adaptaciones y más. El mantenimiento preventivo del software puede abordar pequeños problemas que en un momento dado pueden carecer de importancia, pero pueden convertirse en problemas mayores en el futuro.

En nuestro proyecto, consideramos utilizar los siguientes protocolos de mantenimiento correctivo:

- Actualización del sistema operativo: Todos los programas y aplicaciones que se utilizan en un ordenador, lo hacen sobre el sistema operativo. Por ello, consideramos que añadir actualizaciones continuas, nos servirá para corregir errores o bugs, aumentar las medidas de seguridad (y tapar posibles agujeros de seguridad), para implementar nuevas funciones y para aumentar su rendimiento.
- Configuración de medidas de seguridad: Otro de los factores importantes a tener en cuenta, son las medidas de seguridad del equipo, debido al uso de redes e internet. Para ello, consideramos necesario la instalación y configuración de un cortafuegos que impida acceso no autorizado de terceros, el uso de software antimalware que impida que el equipo se infecte con malware (virus, gusanos, troyanos, etc.), activar y configurar los puntos de restauración del sistema, para poder volver a un estado anterior en caso de algún desastre que impida utilizar el equipo, y la Implementación de un sistema de copias de seguridad.

3.6 Mantenimiento correctivo

El mantenimiento correctivo del software es la forma clásica y típica de mantenimiento (para el software y cualquier otra cosa). El mantenimiento de software correctivo es necesario cuando algo sale mal en una pieza de software, o cuando la presencia de un virus afecta el desempeño del computador. Estos pueden tener un impacto generalizado en la funcionalidad del software en general y, por lo tanto, deben abordarse lo antes posible.

Los protocolos de mantenimiento correctivo que utilizaremos en este proyecto son:

- Antivirus: Los virus son uno de los enemigos que más problemas causa a nuestro software, haciendo que nuestros programas se desarrollen de manera lenta, que tengan fallas o que se cierren, etc. Lo recomendable para solucionar este problema sería tener un buen antivirus y tener en cuenta las actualizaciones.
- Restauración del sistema: Restaurar sistema sirve para devolver el equipo a un estado de funcionamiento anterior, esto es muy útil para deshacer los cambios en el sistema y volver a un punto en el que el ordenador funcionaba correctamente.

3.7 Mantenimiento adaptativo

El mantenimiento adaptativo de software tiene que ver con las tecnologías cambiantes, así como con las políticas y reglas relacionadas con su software. Las cuales incluyen cambios en el sistema operativo, almacenamiento en la nube, hardware, etc. Cuando se realizan

estos cambios, su software debe adaptarse para cumplir adecuadamente los nuevos requisitos y continuar funcionando bien.

- Controlador de versiones: Un controlador de versiones es un sistema que nos permite guardar un registro de las modificaciones que realizamos sobre un fichero o conjunto de ficheros a lo largo del tiempo de tal manera que sea posible recuperar versiones específicas más adelante.

3.8 Mantenimiento perfectivo

Al igual que con cualquier producto en el mercado, una vez que el software se lanza al público, surgen nuevos problemas e ideas. Los usuarios pueden ver la necesidad de nuevas características o requisitos que les gustaría ver en el software para convertirlo en la mejor herramienta disponible para sus necesidades. Es entonces cuando entra en juego el mantenimiento perfectivo del software.

- Pruebas: Ningún cambio puede llevarse a cabo con éxito si no incluye una etapa de prueba. Las pruebas son importantes, pues nos ayudarán a identificar problemas y a saber si, una vez instalado en el entorno en el que queremos, nuestro software cubre las expectativas que teníamos. Las pruebas son la forma en que podemos estar seguros acerca de la funcionalidad, el rendimiento y la experiencia del usuario.

4 Conclusión

En este proyecto, a través de las pruebas funcionales, integrales y de sistema, hemos demostrado el funcionamiento del sistema web para poder cumplir con los requisitos de calidad, presentando a los clientes nuevos servicios. Además hemos aprendido que el mantenimiento de software no solamente se trata de corregir errores dentro del sistema, sino que también de crear mejoras, hacer programas más ligeros eliminando lo innecesario, optimizando sistemas y adaptándolos a distintos contextos o a nuevas necesidades.

5 Referencias Bibliográficas.

(KeepCoding, 2022)

(Alvarez, 2020)