Especificación de Requerimientos
Descripción del Diseño

Waint

Sistema de búsqueda y agendamiento de libros para repositorios físicos y virtuales de bibliotecas

Apellidos, Nombres	Correo electrónico	Rol
Luis Carlos Fernández Vargas	Luiscafer728@hotmail.com	Base de datos
Omar Ricardo Castañeda Camargo	omarricardocc@gmail.com	Master Scrum
Miguel Angel Antonio Fernandez	miguelangelantoniofernadez@gmail.com	Back End

Fecha de presentación: 28/08/2022

Contenido

1 INTRODUCCIÓN 3

1.1 Propósito 3
1.2 Alcance o Ámbito del Sistema 3
1.3 Definiciones, Acrónimos y Abreviaturas 3
1.3.1 Definiciones 3
1.3.2 Referencias 4
1.4 Perspectiva General del Documento 4

2 DESCRIPCIÓN GENERAL DE LA APLICACIÓN 4

2.1 Perspectiva de la Aplicación 4
2.2 Funciones de la Aplicación 4
2.3 Características de los Usuarios 5
2.4 Restricciones 5
2.5 Suposiciones y Dependencias 5
2.6 Requerimientos Diferidos 5

3 REQUERIMIENTOS ESPECÍFICOS 6

3.1 Requerimientos 6
3.1.1 Product Backlog 6
3.1.2 Ciclo de Sprints del proyecto 6
3.1.3 Sprint Backlog 7
3.1.4 Historias de usuario (Tareas y Subtareas) 7
3.1.5 Mecánica de organización del grupo. (Reuniones, evidencias/artefactos) 8
3.2 Modelo de Requerimientos 8
3.2.1 Modelo de Casos de Uso 8

4 DESCRIPCIÓN DEL DISEÑO 9

4.1 Interfaz gráfica (Mockups) 9

5 GESTIÓN DE LA CONFIGURACIÓN 12

5.1 Frontend 12
5.2 Base de datos 14

6 PRUEBAS 18

6.1 Descripción de pruebas unitarias 18
6.2 Descripción de pruebas de aceptación ..... **¡Error! Marcador no definido.**

7 GLOSARIO ..... ¡ERROR! MARCADOR NO DEFINIDO.

8 ANEXO(S) ..... ¡ERROR! MARCADOR NO DEFINIDO.

1 INTRODUCCIÓN

El presente documento tiene como finalidad presentar la documentación, descripción, explicación y detalles sobre el uso configuración y código empleado para realizar una aplicación amigable para el usuario y operador en la interminable tarea de almacenar, vender prestar o remover de un inventario un libro

1.1 Propósito

El documento tiene como fin presentar al cliente un resumen ejecutivo de las principales características del programa del cual se va a hacerse acreedor, para llevar la tarea de inventarios y prestamos de una biblioteca de manera más amigable y fácil.

1.2 Alcance o Ámbito del Sistema

Waint es un programa que integra los diferentes beneficios a la hora de almacenar, prestar eliminar o agregar un libro a un su inventario, permitiendo que tanto el usuario que se presente a su biblioteca a pedir un libro prestado o para comprarlo sepa si se encuentra o no disponible y que día será devuelta del préstamo para poder solicitarlo y al operario cuales libro faltan en el inventario, agregar nuevos o simplemente sacarlos de circulación.

1.3 Definiciones, Acrónimos y Abreviaturas

1.3.1 Definiciones

PhP: Es un lenguaje de programación enfocado en el desarrollo web, que permite a los usuarios cambiar modificar y agregar opciones al entorno de trabajo visual del programa.
CSS: Es un lenguaje de programación muy usado para establecer el diseño visual de los documentos web, e interfaces de usuario escritas en HTML o XHTML.
Python: Es un lenguaje de programación de cuarta generación en su versión 3 enfocado en resolver los paradigmas, imperativo, procedural, orientado a objetos y funcional.
Paradigma: Es una teoría o conjunto de teorías cuyo núcleo es aceptado por todos los integrantes que usan este paradigma y sirve como modelo para resolver este ripo de problemas

Django: es un framework de trabajo para Python que permite la integración de front end -backend – base de datos de manera intuitiva y rápida.
PEP 8: Guia de estilo para el desarrollo de código en Python.

1.3.2 Referencias

1) IEEE Std-830-1998.
2) ISO-IEC-IEEE-298148
3) IEEE Std-1016-2009
4) ISO/IEC/IEEE 29148:2011
5) OMG Unified Modeling Language
6) Schwinger, W.; Koch, N. "Modeling Web Applications", Chapter 3 en: Kappel, G.; Pröll,
7) Koch, N.; Knapp, A.; Zhang, G.; Baumeister, H. "UML-Based Web Engineering. An Approach Based on Standards", Chapter 7 en: Rossi, G.; Pastor
8) Python a Fondo Oscar Ramirez Jimenez.

1.4 Perspectiva General del Documento

El documento está dirigido para el que operador y el usuario tengan una idea general de cómo funciona Waint y que ventajas sobre otros programas de almacenamiento interactivo de libros en cuanto a tiempo y requerimientos a partir de una interfaz minimalista.

2 DESCRIPCIÓN GENERAL DE LA APLICACIÓN

2.1 Perspectiva de la Aplicación

Waint es una aplicación que permite el almacenamiento de libros con una gran numero de criterios como, Genero, autor, año y costo permitiendo con gran detalle y precisión la información de los diferentes libros, permitiendo a los usuarios saber toda la información relevante del libro a la hora de querer hacerse con una copia o simplemente pedirlo prestado.

2.2 Funciones de la Aplicación

Waint cuenta con las siguientes funciones

1) Búsqueda de libros
2) Agendamiento o préstamo de libros

3) Trazabilidad en los préstamos de libros
4) Información relevante de los libros
5) Actualización del inventario de libros que permita agregar, quitar o actualizar el estado de los libros.

2.3 Características de los Usuarios

Los usuarios amates de los libros encuentran en Waint una aplicación fácil de usar en donde podrán realizar búsquedas por genere, autor o simplemente el nombre del libro del cual quieren pedir prestado o comprar.

2.4 Restricciones

Las siguientes son las restricciones con las cuales Waint cuenta.

1) No alcance información del usuario más allá del número de cedula.
2) Tiene un límite de búsquedas simultaneas por usuario 1 por libro.
3) El sistema solo permite el ingreso o la modificación de un libro a la vez y se debe realizar sobre la base de datos, no permite cargar listar y su reorganización.
4) La interfaz gráfica es minimalista por tal motivo no tiene integrada un gran número de funciones o botones y funciona de manera secuencial.

2.5 Suposiciones y Dependencias

Las siguientes son las suposiciones de Waint para asegurar su optimo funcionamiento.

1) La biblioteca cuenta con una información de usuarios o suscriptores de fácil ingreso
2) La biblioteca cuenta con información mínima sobre el estado actual de inventario de libros que permita que se realice la carga de información de a manera sencilla.
3) La biblioteca cuenta con un operario que realice la carga de información a Waint.

2.6 Requerimientos Diferidos

Actualmente Waint tiene como propósito implementar en versiones futuras la conexión a diferentes bases de datos almacenadas en la nube, que no requieran un servidor físico ni de un tipado de base de datos especifico, además se busca la interacción con diferente host que permitan la estimar cuales generan menor latencia y mayor rendimiento en la generación de resultados.

3 REQUERIMIENTOS ESPECÍFICOS

3.1 Requerimientos

Los requerimientos específicos para Waint actualmente son los siguientes:

- Interacción con las tres diferentes bases de datos usuarios, libros, fechas
- Guardar información acerca de la reserva compra o eliminación de un libro bajo solicitud de un usuario o administrados de la base datos
- Informar a los usuarios de manera clara y sencilla si el libro en cuestión objeto de su búsqueda o interés esta disponible, de no estarlo informarle la fecha en la cual este va ser retornado para su reserva.

3.1.1 Product Backlog

Los siguientes serán the producto Backlog a entregar a lo largo del ejercicio.

1) Creación de Repositorio de Githut
2) Enlace de repositorios
3) Creación de Trello
4) Documento de gestión de configuración
5) Plantilla Front End
6) Historias de usuario en Trello
7) Historia de usuario
8) Acta de entrega y reuniones semanales
9) Generación de Botones
10) Interfaz Grafica
11) Conexión Front End Back End
12) APIs
13) Código
14) Evaluación de conectividad con bases de datos
15) Evaluación de conectividad con Front End
16) Conectividad Base de datos
17) Conectividad Front End
18) Base de datos Usuario
19) Base de datos Libros
20) Base de datos Fechas
21) Testear Base de datos
22) Testear Back End
23) Testear Front End
24) Testear la página en total

3.1.2 Ciclo de Sprints del proyecto

A continuación, se relacionarán los esprints y sus fechas para evaluación.

- Sprint 1 28/08/2022
- Sprint 2 11/09/2022
- Sprint 3 18/09/2022
- Sprint 4 25/09/2022

3.1.3 Sprint Backlog

Sprint 1

- Creación de Repositorio de Githut
- Enlace de repositorios
- Creación de Trello
- Documento de gestión de configuración (avance)
- Historias de usuario en Trello
- Historia de usuario
- Plantilla Front End

Sprint 2

- Interfaz Grafica
- Conexión Front End Back End
- Documento de gestión de configuración (avance)
- Base de datos Usuario
- Base de datos Libros
- Base de datos Fechas
- Historias de usuario en Trello (Actualizadas)
- Historia de usuario (Actualizadas)
- Acta de entrega y reunión (Actulizadas)

Sprint 3

- APIs
- Código
- Conectividad Base de datos
- Conectividad Front End
- Testear Base de datos
- Testear Front End
- Historias de usuario en Trello (Actualizadas)
- Historia de usuario (Actualizadas)
- Acta de entrega y reunión (Actulizadas)

Sprint 4

- Código
- Evaluación de conectividad con bases de datos
- Evaluación de conectividad con Front End
- Conectividad Base de datos
- Conectividad Front End
- Historias de usuario en Trello (Actualizadas)
- Historia de usuario (Actualizadas)
- Acta de entrega y reunión (Actulizadas)

3.1.4 Historias de usuario (Tareas y Subtareas)

3.1.5 Mecánica de organización del grupo. (Reuniones, evidencias/artefactos)

3.2 Modelo de Requerimientos

Para la evaluación de Waint se llevarán a cabo reunión semanales y reuniones para acta de entrega de los diferentes product backlog y de esta forma llamar un alto grado de control en el desarrollo de la aplicación.

3.2.1 Modelo de Casos de Uso

A continuación, se validarán los métodos de caso que serán usados en los diferentes métodos a evaluar o información de entrada.

CU-01: "Validación de datos de usuario y contraseña"

Iniciador	Entrar	
Otros actores	Backend and base of dates	
Precondiciones	Condiciones que deben cumplirse para que pueda realizarse el caso de uso.	
Flujo básico		
Actor		**Sistema**
1 FrontEnd		
		2. Rest API.
		3. Rest API.
4. BackEnd		
5. Base of dates		
Flujo alternativo 1		
Poscondiciones	Número de identificación y la cedula.	

CU-02: "Buscar "

Iniciador	Buscar	
Otros actores	Backend and base of dates	
Precondiciones	Condiciones que deben cumplirse para que pueda realizarse el caso de uso.	
Flujo básico		
Actor		**Sistema**
1 FrontEnd		
		2. Rest API.
		3. Rest API.
4. BackEnd		
5. Base of dates		
Flujo alternativo 1		
Poscondiciones	Base de datos de libros	

CU-03: "Autor, Titulo, Tema, Editoral "

Iniciador	Autor, Titulo, Tema, Editoral	
Otros actores	Backend and base of dates	
Precondiciones	Condiciones que deben cumplirse para que pueda realizarse el caso de uso.	
Flujo básico		
Actor		**Sistema**
1 FrontEnd		
		2. Rest API.
		3. Rest API.
4. BackEnd		
5. Base of dates		
Flujo alternativo 1		
Poscondiciones	búsqueda por Autor, Titulo, Tema, Editoral	

CU-04: "Tipos de documento "

Iniciador	Autor, Titulo, Tema, Editoral	
Otros actores	Backend and base of dates	
Precondiciones	Condiciones que deben cumplirse para que pueda realizarse el caso de uso.	
Flujo básico		
Actor		**Sistema**
1 FrontEnd		
		2. Rest API.
		3. Rest API.
4. BackEnd		
5. Base of dates		
Flujo alternativo 1		
Poscondiciones	búsqueda por Autor, Titulo, Tema, Editoral	

4 DESCRIPCIÓN DEL DISEÑO

4.1 Interfaz gráfica (Mockups)

A continuación, se muestra un bosquejo de lo que se espera se convierta Waint.

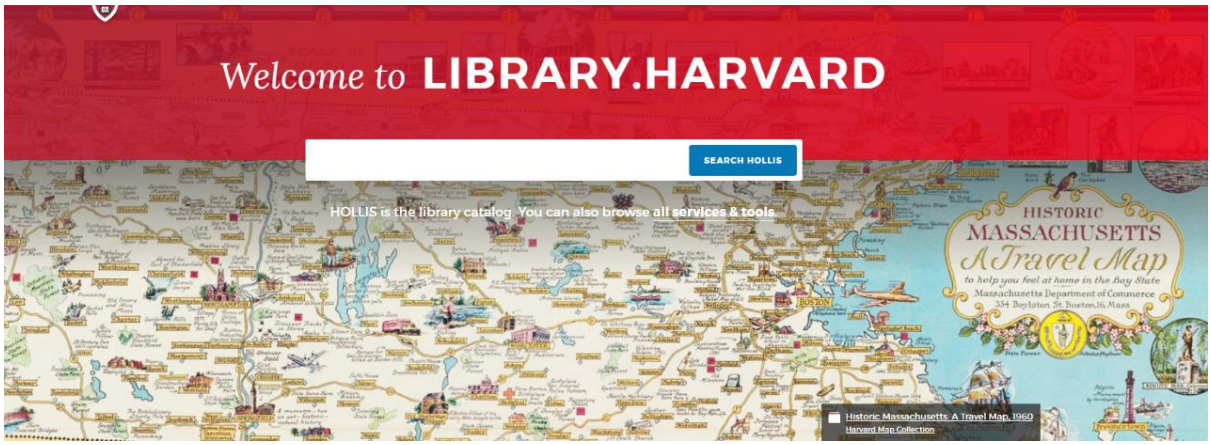

Figura 1 Bosquejo General de Waint

Luego de unas semanas de trabajo la interfaz mejoro tomando su propia esencia como se puede apreciar en la siguiente figura, con matices minimalistas y que gracias a los modelos de casos de uso nombrados anteriormente cuenta con todos los medios necesarios para competir con interfaces de las mas grandes bibliotecas en el mundo.

Figura 2 Interfaz actualizada 11/09/2022

Posterior a las reuniones llevadas acabo se llego a la conclusión que el esquema general de la interna debía cambiar dando como origen los que se muestran acentuación.

Figura 3 Interfaz actualizada 18/09/2022

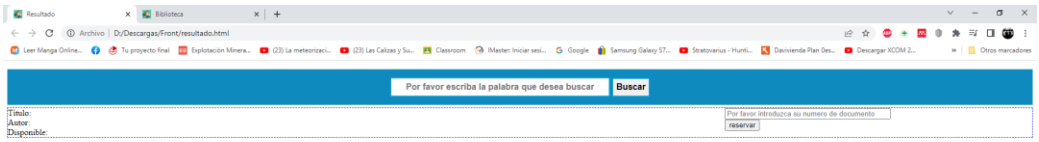

Figura 4 Interfaz actualizada 18/09/2022

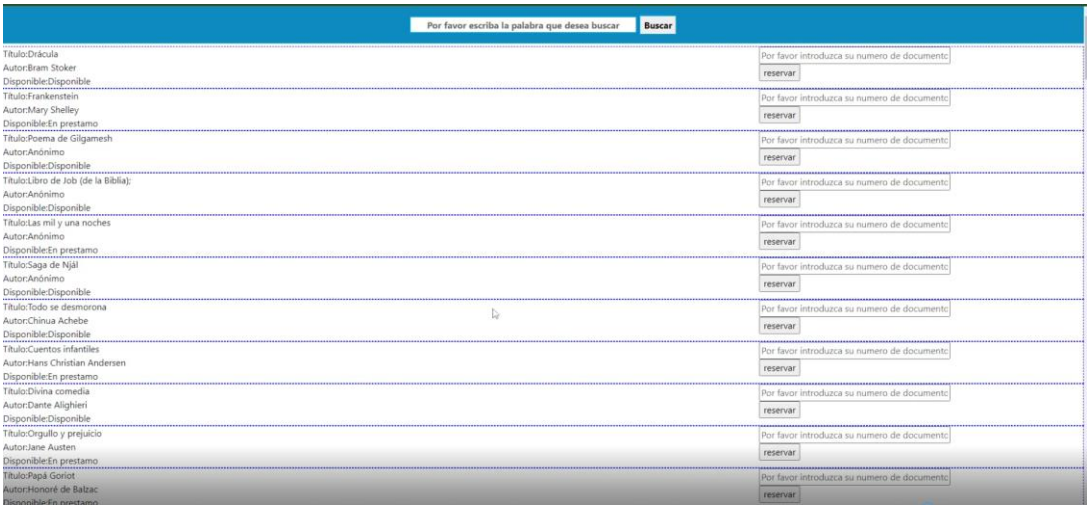

Figura 5 Interfaz actualizada 30/10/2022

5 GESTIÓN DE LA CONFIGURACIÓN

5.1 Frontend

Para la gestión de configuración del Front end se llevo acabo reuniones semanales que permitieran identificar cuales serian los elementos de la interfaz que permitieran un fácil manejo y rápida búsqueda para esta parte se uso CSS, PHP.

```
h1{
    font-size: 70px;
    text-align: center;
    position: absolute;
    width: 100%;
    text-shadow: 2px 2px 4px black;
    color:white;
}
.busqueda_simple{
    text-align: center;
    width: 100%;
    position: absolute; /* permite que la barra de búsqueda se monte sobre la imagen de banner */
    top: 400px;
}
.busqueda_avanzada{
    font-size: 20px;
    text-align: center;
}
h2{
    font-size: 30px;
    text-align: center;
}
```

Mientras que para la pagina de resultado se uso la siguiente.

```
.resultado_busqueda_simple{
    display: block;
    width: 100%;
    padding: 15px;
    border: none;
    margin-bottom: 5px;
    box-sizing: border-box;
    font-size: 1rem;
    text-align: center;
    text-decoration: none;
    background: #0e8abf;
    color: #000;
}
.grid-container{
    display: grid;
    grid-template-columns: 70% auto;
    justify-items: left;
    border: 1px dashed blue;
}
```

Mientras que para la interracion del frontend -backend se uso el framework django

```
STATICFILES_DIRS=['G:/Mi unidad/Mision_tics/Ciclo 3/Proyecto biblioteca/myproject/myproject/static']

# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'myproject.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        #en dirs agregamos la ruta de la carpeta donde esta el html importante cambiarla cuando se suba a github
        'DIRS': ['G:/Mi unidad/Mision_tics/Ciclo 3/Proyecto biblioteca/myproject/myproject/web'],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'myproject.wsgi.application'


# Database
# https://docs.djangoproject.com/en/2.2/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
```

```
}


# Password validation
# https://docs.djangoproject.com/en/2.2/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
   {
      'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
   },
   {
      'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
   },
   {
      'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
   },
   {
      'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
   },
]


# Internationalization
# https://docs.djangoproject.com/en/2.2/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/2.2/howto/static-files/

STATIC_URL = '/static/'
```

5.2 Base de datos

Para las bases de datos se uso la herramienta de desarrollada por Microsoft SQL server en la versión 3 de este documento.

```
#CREACIÓN BASE DE DATOS

CREATE DATABASE MINTIC_CICLO3

# CREACIÓN TABLA LIBROS DISPONIBLES EN BIBLIOTECA
SE MINTIC_CICLO3

DROP TABLE #LIBROS
CREATE TABLE #LIBROS (ID_Libro NVARCHAR(4),Nombre_Libro NVARCHAR(200),Autor
NVARCHAR(200),Estado NVARCHAR(20))
```

```
# CREACIÓN TABLA USUARIOS DE BIBLIOTECA
DROP TABLE #USUARIOS
CREATE TABLE #USUARIOS (ID_User NVARCHAR(10),Nombre_Usuario NVARCHAR(200),Direccion_Usuario NVARCHAR(200))
```

Para la versión 4 de este documento se migro a MySql ya que la versión de Django usada no soportaba del todo el Microsoft SQL server generando conflicto a la hora de realizar la conexión backend – Base de datos, tal y como se muestra continuación.

5.3 BackEnd

Para el backEnd waint usa el framework django a partir de la generación de clases y subclases como se muestra acontinuacion.

```
class Migration(migrations.Migration):

    initial = True

    dependencies = [
    ]

    operations = [
        migrations.CreateModel(
            name='libros',
            fields=[
                ('ID_Libro', models.IntegerField(max_length=40, primary_key=True, serialize=False)),
                ('Nombre_Libro', models.CharField(max_length=40)),
                ('Autor', models.CharField(max_length=40)),
                ('Estado', models.CharField(max_length=10)),
            ],
        ),
        migrations.CreateModel(
            name='prestamos',
            fields=[
                ('ID_Libro', models.CharField(max_length=40, primary_key=True, serialize=False)),
                ('ID_User', models.IntegerField(max_length=20)),
                ('Estado', models.CharField(max_length=10)),
                ('Fecha_inicio', models.DateField()),
                ('Fecha_fin', models.DateField()),
            ],
        ),
        migrations.CreateModel(
            name='usuarios',
            fields=[
```

```
            ('ID_User', models.IntegerField(max_length=20, primary_key=True,
serialize=False)),
            ('Nombre_Usuario', models.CharField(max_length=40)),
            ('Direccion_Usuario', models.CharField(max_length=60)),
        ],
    ),
]
```

Otras de las clases de las que se hace son las siguientes:

```
class libros (models.Model):
    ID_Libro = models.CharField(max_length=40, primary_key=True) # aca
indicamos la llave primaria que ayuda a la conexion con las otras tablas
    Nombre_Libro = models.CharField(max_length=80) #charfield indica el tipo de
dato que la tabla va a recibir y la extension del texto
    Autor = models.CharField(max_length=80) #
    Estado = models.CharField(max_length=40)


class prestamos (models.Model):
    ID_Libro = models.CharField (max_length=40, primary_key=True)
    ID_User = models.CharField (max_length=20)
    Estado = models.CharField (max_length=20)
    Fecha_inicio = models.DateField ()
    Fecha_fin = models.DateField ()

class usuarios (models.Model):
    ID_User = models.CharField (max_length=20, primary_key=True)
    Nombre_Usuario = models.CharField (max_length=40)
    Direccion_Usuario = models.CharField (max_length=60)
```

Posterior a eso se defienen las funcionen que ejecutaran la interacción con el Back End, Frontend y posteriormente la base de datos en este caso se defienen dos variables claves "comunicaciónBD" que sirve para comunicar las búsquedas generadas desde el frontend en el botón búsqueda y la base de datos e "Index" que sirve para redireccionar a la ventana definida en el frontend como inicio y servir como puente para las solicitaciones.

```
def index(request):
    return render(render, 'index.html')
# Creamos la clase que va a mostrar los datos solicitados de la base de datos
def comunicacionDB (request):
    busqueda= request.get.get('consulta')
    libros= libros.objects.all()

    if busqueda:
        libros = libros.objects.filter(
            Q(Nombre_Libro_icontains = busqueda) |
```

```
            Q(Autor_icontains = busqueda)
        ).distinct()
    return render(request, 'resultado.html')
```

5.4 Conexiones con entre base de datos – backend, backend - frontend

Para este caso el framework django permite una fácil conexión entre los diferentes componentes de trabajo como se muestran a continuación

- Conexión base de datos backend

```
# Database
# https://docs.djangoproject.com/en/2.2/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'mintic_ciclo3',
        'USER': 'root',
        'PASSWORD': '123456',
        'HOST': 'localhost',
        'PORT': '3306',
        'OPTIONS': {
            'read_default_file': 'G:/Mi unidad/Mision_tics/Ciclo 3/Proyecto biblioteca/Base de datos/DB - Biblioteca - mySQL.sql',
        },
    }
}
```

- Conexión frontend - backend

```
OOT_URLCONF = 'myproject.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        #en dirs agregamos la ruta de la carpeta donde esta el html importante cambiarla cuando se suba a github
        'DIRS': ['G:/Mi unidad/Mision_tics/Ciclo 3/Proyecto biblioteca/myproject/myproject/web'],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
```

```
        ],
      },
    },
]

WSGI_APPLICATION = 'myproject.wsgi.application'


urlpatterns = [
    path('admin/', admin.site.urls),
    #abajo encontramos el enlace web con el proyecto
    path('index/', index)
]
```

6 PRUEBAS

Para el desarrollo de las pruebas se siguió la literatura propuesta en los documentos de referencia.

6.1 Descripción de pruebas unitarias

Las pruebas unitarias que se llevarán serán las siguientes:

- Prueba unitaria
  Pruebas Unitarias Software Biblioteca
  Verificar partes específicas de la aplicación: Clases, Componentes y métodos
  Escribir prueba – Programar funcionalidad – Mejorar código

  Después de acceder a la codificación del software, se importa la librería unittest, se crea un archivo llamado test.py, se crea una clase ampliando la clase unittest.TestCase, se escriben métodos y en cada método se crean diferentes casos de prueba.
  Para la ejecución de la prueba se utilizó el comando python -m unittest test_manage.py, luego se invocó el método main de unittest en el archivo de prueba.
  Finalmente, para ejecutar las pruebas se utilizó el comando python -m unittest discover.
  Se compara la salida del código con la salida esperada y se procede a realizar la documentación de las pruebas con sus respectivos comentarios para la optimización de la codificación.

- Prueba de Integración

  Las pruebas de integración se llevaron a cabo mediante llamados a diferentes libros y determinar si la conexión se estaba llevando de una manera correcta, de esto se identificó lo siguiente.

1) Se tiene que llamar el libro de manera exacta, por su nombre ya que las funcionen buscan en la base de datos la columna nombre y no autor o género.
2) Las bases de datos tienen que ser generadas desde django de manera nativa ya que de otra forma presentaría problemas por caracteres especiales o letras latinas.
3) Una manera rápida de búsqueda se detecto mediante el filtro por disponibilidad del libro luego de identificando en la base de datos por su nombre permitiendo que arrojara resultados de disponible o no, si no se encuentra disponible por ende se tiene que realizar una solicitud para determinar cual es su fecha de devolución.

- Pruebas de Aceptación.

  La prueba se realizó mediane el uso de las herramientas curl o imsomnia para realizar la llamada, arrojando una correcta conexión y entre las partes del programa y permitiendo depurar el código ya que existían redundancias.