UTPL

UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA

FICHAS MOVIMIENTOS DE MASAS

ARQUITECTURA DE APLICACIONES
PROYECTO IIBIMESTRE
TUTOR: MSC. LUIS QUIÑONES

INTEGRANTES
*SILVANA P. VÉLEZ M.
*ROOSEVELT J. JARAMILLO

PROBLEMÁTICA

El principal problema que tienen los Geólogos al momento de realizar las visitas de campo son:

- La hoja de las fichas técnicas se llena a mano.
- Ocurre algunas veces que se pierde la información en el momento que se extravía las hojas, por lo que llevan mucho material al momento de realizar las salidas de campo.
- La información que se obtiene luego de realizar las visitas de campo es estática por lo que cuando se transcribe la información a la computadora se guarda como un archivo diferente y en el momento de buscar algo específico se debe de elegir la carpeta, la fecha, el nombre, en si diferentes factores por lo cual se necesita una aplicación que sea centralizada para que todo se guarde en un mismo lugar y sea usable para el USUARIO.
- Falta de seguridad al momento de guardar la información ya que se puede borrar sin ningún problema ya que todos pueden acceder a la información porque no posee un usuario y contraseña de las personas que administran l a información.
- El problema que presentan los Geólogos de la UTPL es similar al problema suscitado en el Registro Civil cuando se perdía la información por tenerla en físico en carpetas siendo así el problema principal la perdida de la información, dificultad al buscar, al momento de registrar era un proceso tardío y obsoleto.

SOLUCIÓN

La solución propuesta es la siguiente:

- Desarrollar un aplicativo Web que permita lo siguiente:
  - Permitir la Gestión de Usuarios.
  - Permitir la Gestión de la Ficha de Movimientos de Masas.
  - Visualizar, Presentar y Descargar en Formato Pdf las Fichas de Movimientos de Masas.

FUNCIONALIDADES

FUNCIONES ADMINSITRADOR

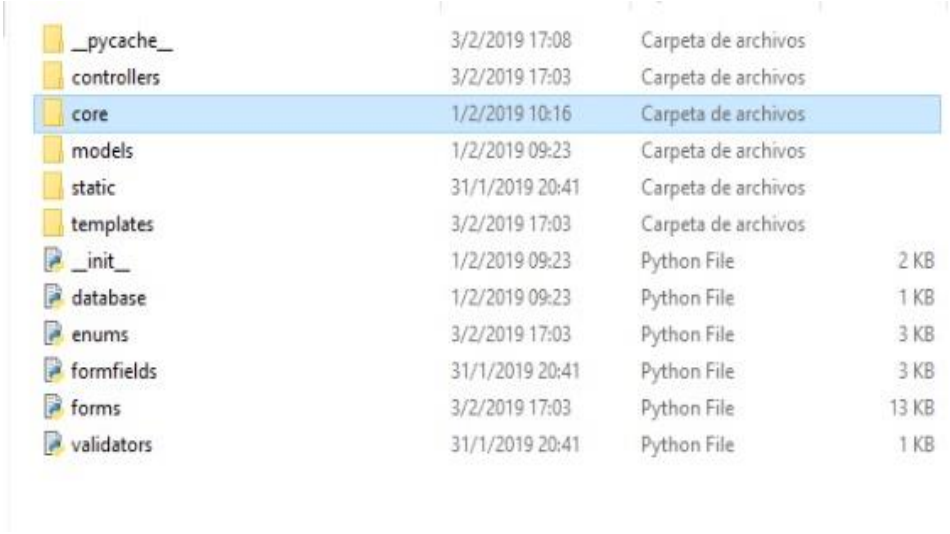
- El administrador puede crear uno o varios geólogos.
- El administrador puede realizar el CRUD en la lista de usuarios geólogos.
- El administrador puede visualizar las fichas y permitirle eliminarlas. Presentar todas las fichas con el usuario que lo ha creado y poder ELIMINARLAS. PRESENTAR Y ELIMINAR.

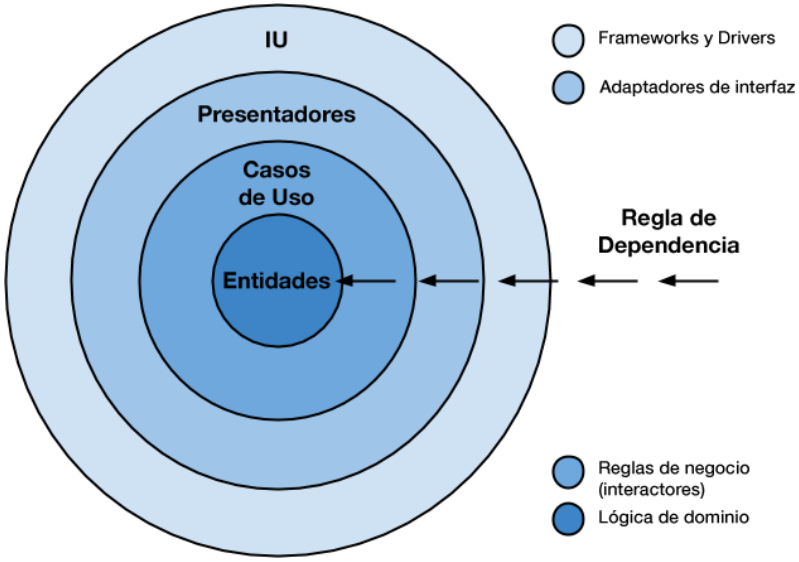
FUNCIONES DE GEOLOGO

- El geólogo puede realizar el CRUD en las Fichas Geológicas.
- El geólogo puede PRESENTAR LAS FICHAS en formato PDF.
- El geólogo NO PUEDE ADMINISTRAR SU USUARIO; solo puede cambiar su contraseña.

Arquitectura

- Se trabaja con una arquitectura limpia





Frameworks

- Framework python llamado Flaks que nos permite relizar paginas web dinamicas.
- En la actualidad hay muchos lenguajes como php, java pero en este caso les presentaremos una tecnologia que nos permite crear de una manera muy censilla aplicaciones web en python.
- Flask es un “micro” Framework escrito en Python y concebido para facilitar el desarrollo de Aplicaciones Web bajo el patrón MVC

HERRAMIENTAS

- Python

El lenguaje en el que se desarrollo es Python ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma.

- PostgreSQL

Es un sistema de gestión de bases de datos relacional orientado a objetos y de código abierto.

Patrones Builder

```python
class AppBuilder:
    def __init__(self):
        self.app = Flask(__name__, instance_relative_config=True)

    def load_config(self, test_config=None):
        if test_config is None:
            self.app.config.from_pyfile('config.py', silent=True)
        else:
            self.app.config.update(test_config)

        if not os.path.isdir(self.app.instance_path):
            os.makedirs(self.app.instance_path)

    def init_db(self):
        with self.app.app_context():
            from . import database

            self.app.teardown_appcontext(database.close_db)
            self.app.cli.add_command(database.init_db_command)

    def register_routes(self):
        @self.app.route('/')
        def index():
            return render_template('index.html')
```

- Separa en pasos la construcción de un Objeto Complejo.

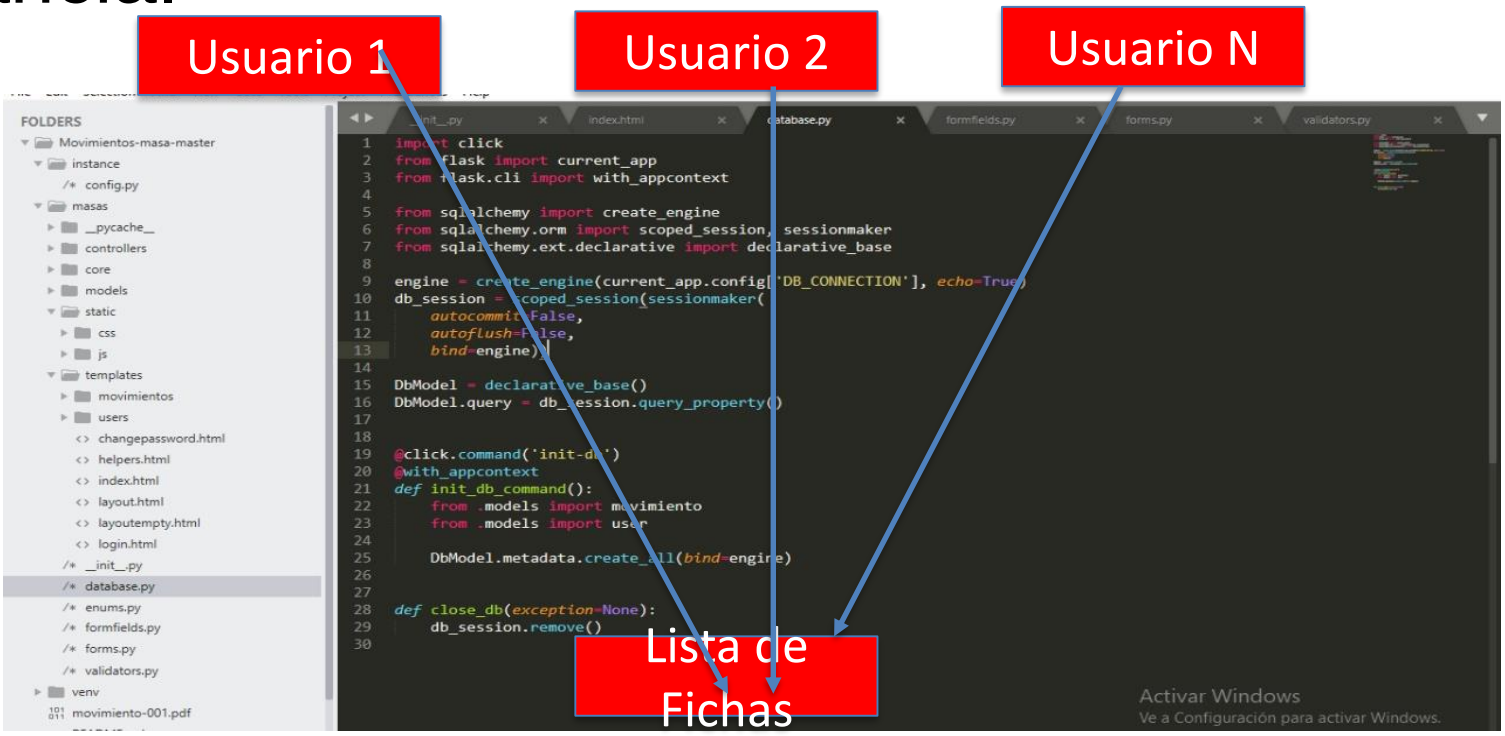
```python
def create_app(test_config=None):
    builder = AppBuilder()

    builder.load_config(test_config)
    builder.init_db()
    builder.register_routes()

    return builder.app
```

Patron Singleton

- Este patrón busca garantizar que por cada clase exista una única instancia para ello utiliza un método de clase que devuelve esa instancia.



Patrón Decorator

```python
def authorize(role=None):
    def wrapper(view):
        @functools.wraps(view)
        def wrapped_view(*args, **kwargs):
            if not session.get('user_id'):
                return redirect(url_for('account.login'))
            elif role and session.get('user_role') != role.name:
                flash('No tienes permiso para esta sección')
                return redirect(url_for('account.login'))
            return view(*args, **kwargs)

        return wrapped_view

    return wrapper
```

Patrón Observer

```python
class NotifyObserver:
    def send_mail(self, subject, message):
        to = current_app.config['EMAIL_NOTIFY']
        user = current_app.config['EMAIL_USER']
        pwd = current_app.config['EMAIL_PWD']

        try:
            server = smtplib.SMTP(current_app.config['EMAIL_SERVER'])
            server.ehlo()
            server.starttls()
            server.ehlo()
            server.login(user, pwd)
            header = 'To:%s\nFrom:%s\nSubject:%s\n' % (to, user, subject)
            msg = header + '\n ' + message + '\n\n'

            server.sendmail(user, to, msg)
            server.quit()
        except(smtplib.SMTPException):
            pass

    def update(self, author):
        pass
```

```python
class CreateObserver(NotifyObserver):
    def update(self, author):
        msg = 'El usuario %s ha creado un movimiento en masa. Codigo: %s' % (
                author,
                self.publisher.codigo)

        self.send_mail('Nuevo movimiento en masa', msg)
```

```python
class DeleteObserver(NotifyObserver):
    def update(self, author):
        msg = 'El movimiento en masa con codigo: %s fue eliminado por %s' % (
              self.publisher.codigo,
              author)

        self.send_mail('Movimiento en masa eliminado', msg)
```

Beneficios

- El patrón Singleton crea un único objeto para el manejo de las peticiones a base de datos.
- El patrón Builder permite separar la construcción del objeto de la aplicación en varios pasos para inicializar el framework.
- El patrón Decorator nos ayuda a validar que el usuario haya ingresado.(decora los controladores)
- El patrón Observer nos ayuda a notificar cuando se crea una Ficha mediante correo electrónico.

Gracias

UTPL
UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA