
Ingeniería en Sistemas Computacionales

Diseño de Software

# Proyecto

Marco Ricardo Cordero Hernández

Zapopan, Jal., 14 de noviembre de 2022

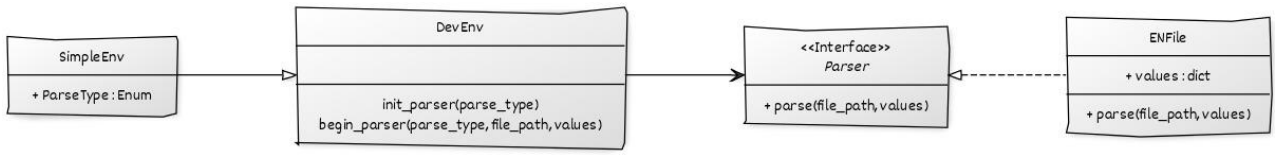
## Proyecto Patrones Creacionales y Estructurales

Cuando se habla acerca del desarrollo del software usualmente se suele pensar en algoritmos complejos y estructuras de datos que harían de la resolución de un problema algo elegante y complejo, sin embargo, la mayoría del tiempo se dejan de lado aspectos de esta parte de la resolución de un problema como lo son la estructuración del código y su funcionamiento a un grado abstracto de operación. El curso actual se ha encargado de visualizar una amplia cantidad de métodos de escritura del código para una amplia gama de problemas que se presentan en el mundo real, ofreciendo la oportunidad de adecuar cada patrón a las necesidades personales.

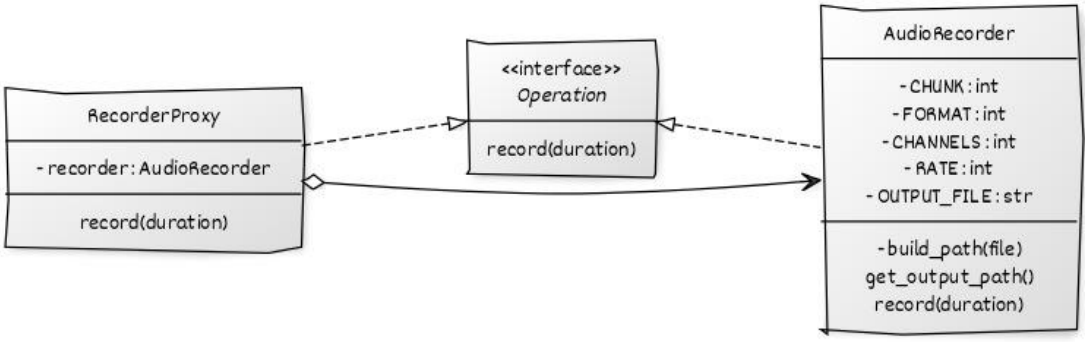
La forma en la que se ha construido una base intelectual de conocimientos relevantes al diseño del software culmina (por ahora) en este proyecto, en donde se trata de demostrar lo revisado a lo largo del curso en una herramienta sumamente útil de reconocimiento de audio. Este entregable ha sido pensado como la parte inicial de lo que pudiera llegarse a convertir en una utilidad más completa, inclusive integrando alguna interfaz gráfica con el fin de brindarle a los usuarios un punto de entrada cómodo e intuitivo.

## Diagrama(s) UML

Para comenzar con el desarrollo de la herramienta, primero se ha pensado de forma conceptual acerca de ella a través de infames diagramas UML. Ahora, como este proyecto cuenta con dos partes que se complementan, se han generado dos diagramas por separado:




*Factory method* (creacional) para parser




*Proxy* (estructural) para grabadora de audio.

# Código

*Patrón creacional Factory method*

```python
# IS727272 - Proyecto - Parser
'''
    Esta parte del proyecto implementa Factory method (patrón creacional)
    con posibilidad de extender sus funcionalidades en releases posteriores.
'''
from __future__ import annotations
from abc import ABC, abstractmethod
from enum import Enum
import re

# Clase auxiliar para implementaciones futuras
class ParserException(Exception):
    def __init__(self, msg:str) -> None:
        ''' Inicializa un error personalizado para parsers '''
        self.msg = msg

    def __str__(self) -> str:
        ''' Retorna la representación del error '''
        return self.msg

# [1] Interface de producto
class Parser(ABC):
    ''' Función principal compartida '''

    @abstractmethod
    def parse(self, file_path, *values) -> object:
        ''' Retorna los valores parseados '''

# [2] Producto(s) concreto(s) abstracto(s)
class ENVFile(Parser):
    def parse(self, file_path, *values) -> dict:
        if (file_path[-3:] != 'env'): raise ParserException('Incorrect file extension != .env')

        res_dict = {}

        regex_str = r"^(" + ("|".join(values) if values else r'.*') + r")\ ?=\ ?'?([^']*)'?"

        with (open(file_path, 'r')) as file:
            for line in file.readlines():
                holder = re.search(regex_str, line)
                if holder:
                    res_dict[holder.group(1)] = holder.group(2).rstrip('\n')

        return res_dict

    @property
    def values(self) -> dict:
        ''' Almacenamiento de valores parseados '''
        return self.__values

    @values.setter
    def values(self, entries: dict) -> None:
        self.__values = entries

# [3] Clase creadora
class DevEnv:
    ''' Clase creadora para recabación de valores en el ambiente '''
    def __init__(self) -> None:
        self.current_parse = None

    def init_parser(self, parse_type: object) -> Parser:
        ''' Factory Method '''
        self.current_parse = parse_type.value
        return self.current_parse

    def begin_parse(self, parse_type: object, file_path, *values):
        return self.init_parser(parse_type).parse(file_path, *values)

# [4] Creadores concretos
class SimpleEnv(DevEnv):
    class ParseType(Enum):
```

```
        # name      value
        ENV     =   ENVFile()
```

*Patrón estructural proxy*

```
# IS727272 - Proyecto - Recorder
'''
    Esta parte del proyecto implementa Proxy (patrón estructural).
'''
import pyaudio, wave, os
from abc import ABC, abstractmethod

# [1] Interfaz de servicio
class Operation(ABC):
    @abstractmethod
    def record(self, duration:float) -> None:
        ''' Recording operation '''

# [2] Servicio
class AudioRecorder(Operation):
    class AudioRecorderException(Exception):
        def __init__(self, msg:str) -> None:
            self._msg = msg

        def __str__(self) -> str:
            return self._msg

    def __init__(self, file_name:str = 'recording', file_extension:str = 'mp3') -> None:
        self._CHUNK = 1024
        self._FORMAT = pyaudio.paInt16
        self._CHANNELS = 2
        self._RATE = 44100
        self._OUTPUT_FILE = self._build_path('.'.join([file_name, file_extension]))

    def _build_path(self, file:str) -> str:
        ''' Build recorded file path '''
        return os.path.join(os.getcwd(), r'..\..\generated', file)

    def get_output_path(self) -> str:
        ''' Returns path of generated recording file '''
        return self._OUTPUT_FILE

    def record(self, duration: float = 5.0) -> None:
        if (duration < 5.0): raise AudioRecorder.AudioRecorderException('Duration too short...')

        p = pyaudio.PyAudio()
        stream = p.open(format=self._FORMAT,
                        channels=self._CHANNELS,
                        rate=self._RATE,
                        input=True,
                        frames_per_buffer=self._CHUNK)

        print(f'Starting recording with lenght of {duration} seconds...')

        frames = []
        for i in range(int(self._RATE / self._CHUNK * duration)):
            frames.append(stream.read(self._CHUNK))
        else:
            print('Recording done...')

        stream.stop_stream()
        stream.close()
        p.terminate()

        wf = wave.open(self._OUTPUT_FILE, 'wb')
        wf.setnchannels(self._CHANNELS)
        wf.setsampwidth(p.get_sample_size(self._FORMAT))
        wf.setframerate(self._RATE)
        wf.writeframes(b''.join(frames))
        wf.close()

        print(f'Audio file "{self._OUTPUT_FILE}" generated within specified directory')

# [3] Proxy
```

```python
class RecorderProxy(Operation):
    def __init__(self, recorder: AudioRecorder) -> None:
        ''' Recorder proxy initializer '''
        self.__recorder = recorder

    def record(self, duration: float = 5.0) -> None:
        ''' Creates audio recording with given duration '''
        self.__recorder.record(duration)

# [4] Cliente (Prueba para el módulo)
if __name__ == '__main__':
    original_service = AudioRecorder('prueba')
    helper = RecorderProxy(original_service)
    helper.record(5)
```

*Código del cliente (herramienta final)*

```python
import requests, os
import Parser, Recorder, json

if __name__ == '__main__':
    parser_helper = Parser.SimpleEnv()
    dotenv_file = parser_helper.begin_parse(parser_helper.ParseType.ENV, r'..\..\.env')
    API_KEY = dotenv_file['API_KEY']
    API = dotenv_file['API']

    recorder_helper = Recorder.AudioRecorder('send_file')
    recorder_proxy = Recorder.RecorderProxy(recorder_helper)
    recorder_proxy.record(duration = 5.0)

    api_data = {
        'api_token': API_KEY,
        'return': 'spotify'
    }

    files = {
        'file': open(recorder_helper.get_output_path(), 'rb')
    }

    result = requests.post(API, data = api_data, files = files).json()['result']

    if (result):
        print(f'\n--- CANCIÓN ENCONTRADA ---\nTítulo: {result["title"]}\nArtista: {result["artist"]}\nLinks:
{result["song_link"]}')
    else:
        print('CANCIÓN NO ENCONTRADA, VUELVE A INTENTAR...')
```

## Demostración

```
(Proyecto) PS C:\Users\marck\OneDrive\Escritorio\Diseño de software\Proyecto\Lib\src> py .\Client.py
Starting recording with lenght of 5.0 seconds...
Recording done...
Audio file "C:\Users\marck\OneDrive\Escritorio\Diseño de software\Proyecto\Lib\src\..\..\generated\send_file.mp3" generated within specified directory

--- CANCIÓN ENCONTRADA ---
Título: Purity
Artista: A$AP Rocky
Links: https://lis.tn/TuKlJh
```

## Consideraciones adicionales

Como se pudo apreciar, este proyecto contiene una funcionalidad de parser, la cual se emplea para parsear un archivo *.env* (IBM, 2022). Este archivo cuenta con dos claves vitales para el funcionamiento del proyecto: la dirección de la API de reconocimiento de audio y una llave de acceso. Si se desea replicar el acceso, se tendría que solicitar la revelación de esta API mencionada y generar una llave propia.

## Repositorio con código completo

[Vínculo a GitHub]

## Referencias bibliográficas

IBM. (2022). *.env file*. Recuperado el 14 de noviembre del 2022 de https://www.ibm.com/docs/en/aix/7.2?topic=files-env-file.