



BASE DE DATOS CLASE 3

{Team DarkCode}

{Juan Ignacio Sorato} – {Sebastián García} – {Javier Brega} – {Gustavo Cesaretti}

{Gabriela Manca} – {Federico Sosa}

Actividad:

- 1- Realizar el cuestionario para la asistencia en el campus
- 2- En grupo crear un nuevo documento con el nombre página web que contenga los archivos columnas.html y columnas.css, añadir la etiqueta viewport y el enlace a la hoja de style de nombre columnas.css como en el ejemplo. Realizar una 1 captura por grupo.
- 3- Realizar 1 ejemplo de Encapsulación, Abstracción, Herencia y Polimorfismo. Utilizar cualquier diagrama UML online para realizarlo.
- Enviar trabajo en formato PDF o Word con el nombre del grupo y alumnos que participaron.
- Fecha de entrega :15/09

The screenshot shows a code editor with two files open: `columnas.html` and `columnas.css`. The `columnas.html` file contains the following code:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0"> <!-- Configuración del viewport para dispositivos móviles -->
  <link rel="stylesheet" type="text/css" href="columnas.css"> <!-- Enlaza tu hoja de estilo CSS -->
  <title>Página web</title>
</head>
<body>
  <div class="contenedor">
    
    <p>
      <!-- Texto de ejemplo -->
      Lorem ipsum dolor sit amet consectetur adipisicing elit. Corporis illum tenetur, laboriosam laborum dolor ad explicabo possimus enim maiores esse eos iure iusto commodi ducimus quo sunt numquam, aliquam suscipit, temporibus alias. Omnis eos placeat dignissimos pariatur magnam commodi, nesciunt veritatis laboriosam illum ipsum nostrum nobis reiciendis? Asperiores iste reiciendis eum libero ipsam voluptatum sequi aliquam ab consequuntur quia quisquam nulla commodi natus alias, perferendis temporibus quo beatae distinctio excepturi itaque eligendi omnis? Autem soluta cum odit, consequuntur dignissimos nisi veniam temporibus necessitatibus quo officia nam, est illo adipisci rerum impedit laborum inventore quas eligendi debitis minus, et ea velit? Provident, officiis impedit. Aspernatur necessitatibus veniam officiis, blanditiis voluptatem architecto quasi tempora doloremque, laboriosam quibusdam porro tempore similique consequuntur natus, cum fugit! Possimus, ut! Architecto tempore veniam voluptatibus expedita? Aliquam exercitationem, delectus modi dignissimos itaque distinctio labore totam dolore repudiandae non alias ipsam ex. Quidem non atque voluptates voluptatum quaerat?
    </p>
  </div>
</body>
</html>
```

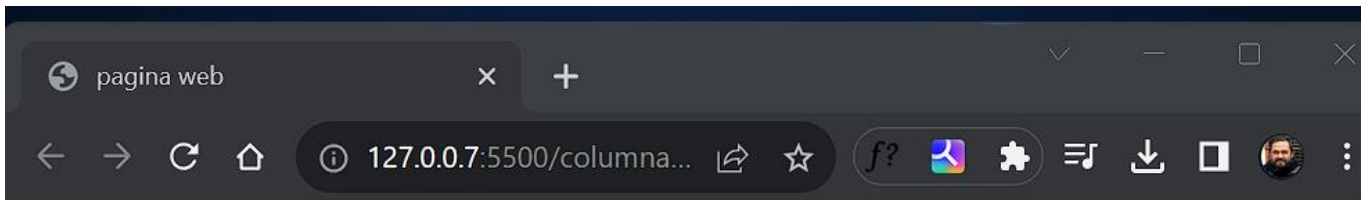
The `columnas.css` file contains the following code:

```
/* Normalización de estilos para mejorar la consistencia entre navegadores */
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

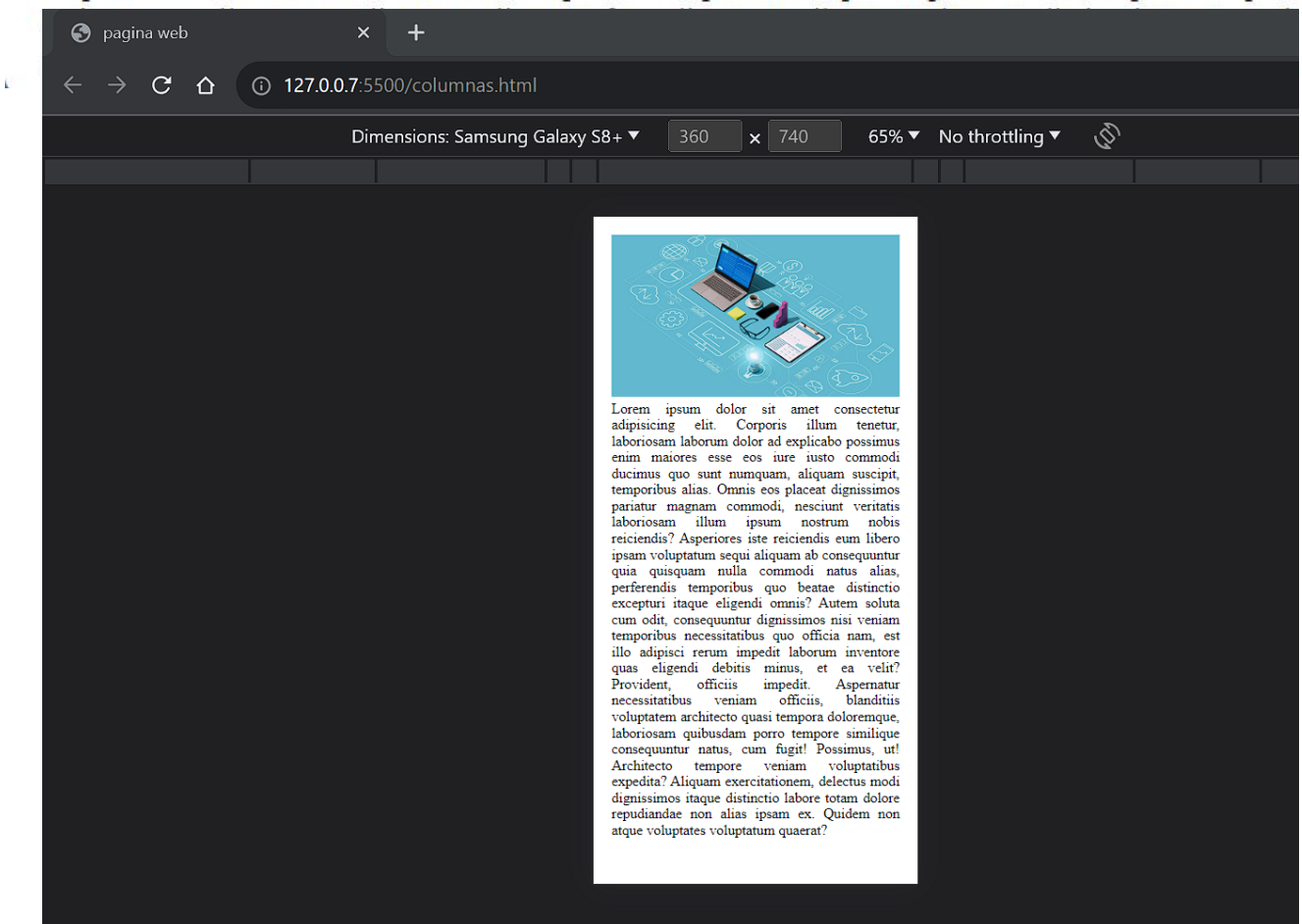
/* Estilos para una imagen en el encabezado (header) */
.header-img {
  width: 100%; /* La imagen ocupa todo el ancho disponible */
  margin: 0; /* Sin margen alrededor de la imagen */
  padding: 0; /* Sin relleno alrededor de la imagen */
}

/* Estilos para un contenedor */
.contenedor {
  text-align: center; /* Centra el contenido horizontalmente */
  max-width: 100%; /* Evita que el contenido sea más ancho que el viewport */
  margin: 0 auto; /* Centra el contenedor horizontalmente */
  padding: 20px; /* Espacio opcional alrededor del contenido */
}

/* Estilos para imágenes */
img {
  max-width: 100%; /* Hace que la imagen sea responsiva */
  height: auto; /* Mantiene la proporción de aspecto de la imagen */
}
```



Lorem ipsum dolor sit amet consectetur adipisicing elit. Corporis illum tenetur, laboriosam laborum dolor ad explicabo possimus enim maiores esse eos iure iusto commodi ducimus quo sunt numquam, aliquam suscipit, temporibus alias. Omnis eos placeat dignissimos pariatur magnam commodi, nesciunt veritatis laboriosam illum ipsum nostrum nobis reiciendis? Asperiores iste reiciendis eum libero ipsam voluptatum sequi aliquam ab consequuntur quia



1-Encapsulación:

UMLet - Free UML Tool for Fast UML Diagrams

File Edit Custom Elements Help

Search:

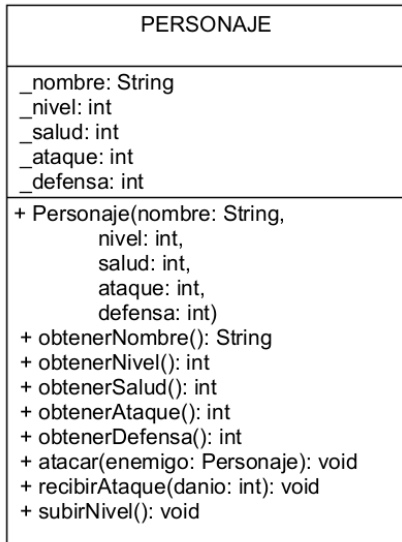
Zoom: 100%

Mail diagram

Encapsulación-{Javier Brega} x

D:\Base de datos - UTN\UML\Encapsulación-{Javier Brega}.uxf opened

ENCAPSULACIÓN



DEFINICIÓN

El diagrama UML muestra los atributos con guiones bajos como prefijo (`_nombre`, `_nivel`, `_salud`, `_ataque`, `_defensa`) para indicar que son atributos protegidos, lo que sigue las convenciones de Python. Además, los métodos públicos (+) para acceder y realizar acciones en los atributos también están representados de la misma manera que se mostró anteriormente.

2-Herencia:

UMLet - Free UML Tool for Fast UML Diagrams

File Edit Custom Elements Help

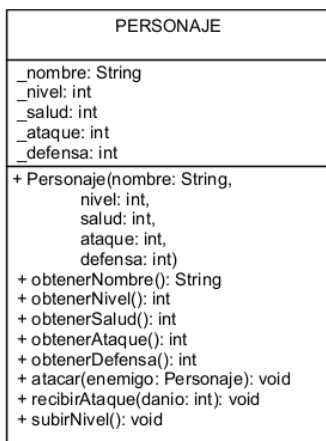
Search:

Zoom: 70%

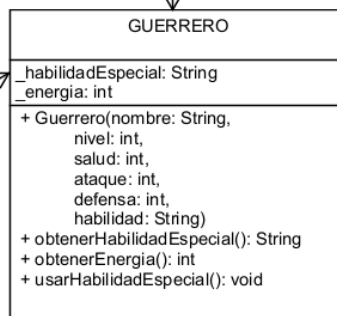
Mail diagram

Herencia-{Javier Brega} x

HERENCIA



GUERRERO



DESCRIPCIÓN EJEMPLO HERENCIA

La clase Guerrero hereda de Personaje utilizando `super().__init__()` para llamar al constructor de la clase base y luego agrega sus atributos y métodos específicos.

La clase Guerrero tiene su propia implementación de `obtenerHabilidadEspecial()` y `usarHabilidadEspecial()`, además de los métodos heredados de Personaje.

La herencia permite que la clase Guerrero aproveche la estructura y el comportamiento de la clase base Personaje, extendiéndola para incluir características específicas de los guerreros.

3-Polimorfismo:

UMLet - Free UML Tool for Fast UML Diagrams

File Edit Custom Elements Help

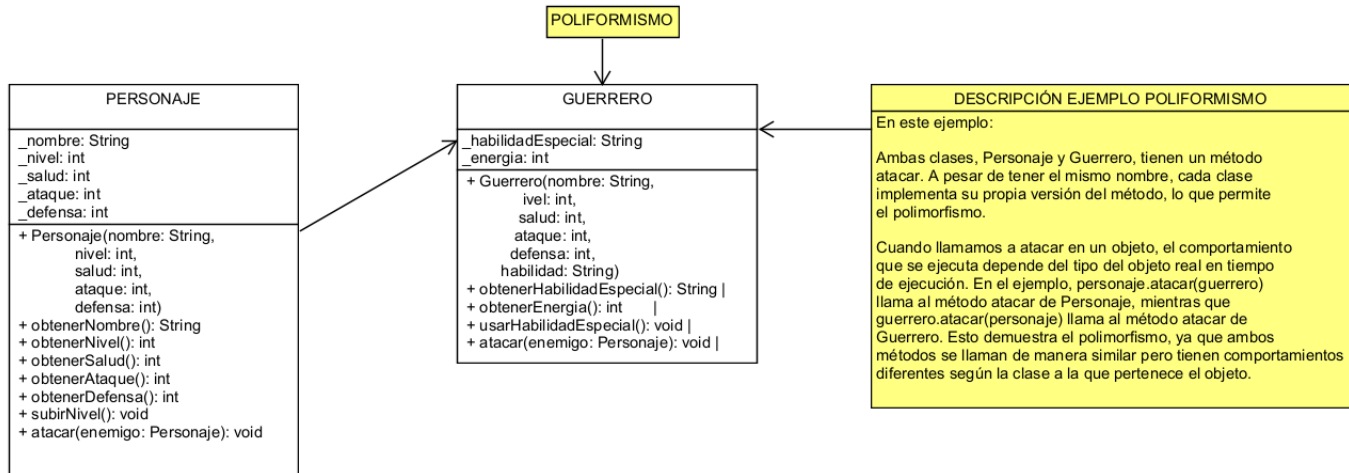
Search:

Zoom:

70%

Mail diagram

Poliformismo-{Javier Brega} * x



4-Abstracción:

UMLet - Free UML Tool for Fast UML Diagrams

File Edit Custom Elements Help

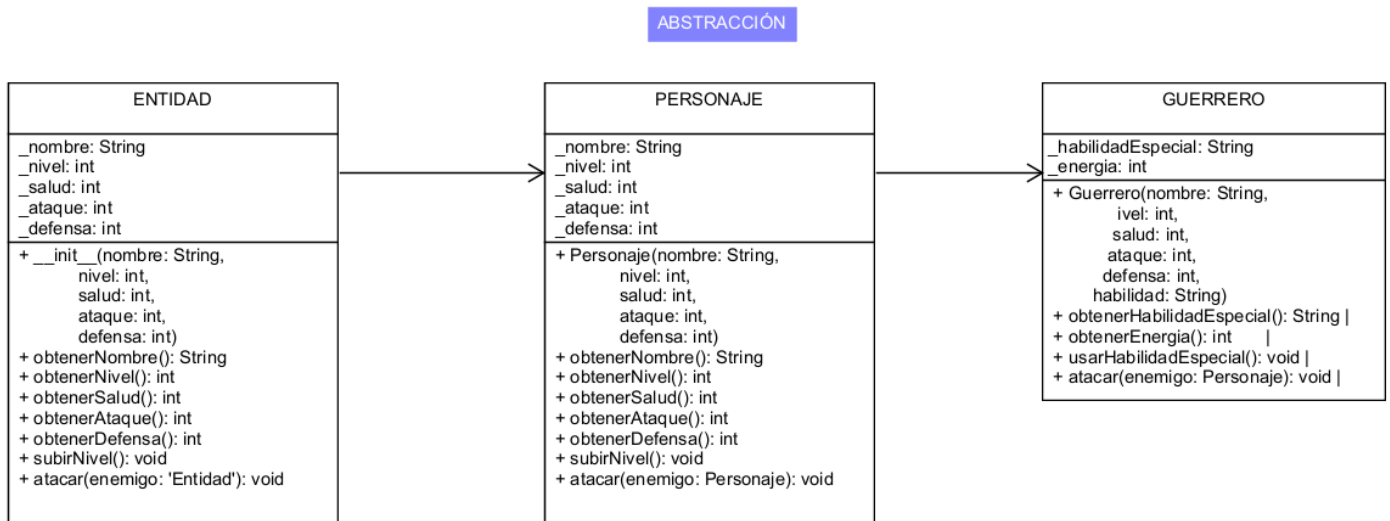
Search:

Zoom:

70%

Mail diagram

Abstracción-{Javier Brega} x



DESCRIPCIÓN EJEMPLO ABSTRACCIÓN

En este ejemplo:

La clase Entidad se define como una clase abstracta utilizando el módulo abc. Contiene la estructura general de un personaje de videojuegos, pero su método atacar es abstracto, lo que significa que debe ser implementado en las clases concretas que heredan de ella.

Las clases Personaje y Guerrero heredan de Entidad y proporcionan implementaciones concretas del método atacar, lo que demuestra la abstracción. Cada clase puede tener su propia implementación de atacar.

La abstracción permite definir una estructura común para las clases relacionadas, como Entidad, y luego crear clases concretas como Personaje y Guerrero que se ajusten a esa estructura pero proporcionen implementaciones específicas según sea necesario. Esto promueve la reutilización de código y la organización de la lógica del programa.