

**Licenciatura en Sistemas de Información**

**Trabajo Práctico N°3**

**Materia**: Ingeniería de Software II

**Alumnos**: Putallaz Gian

**Profesor**: Dr. Pedro E. Colla

**Ayudante**: Hernán Sanchez

1. <https://github.com/gianquito/UADER_IS2_Putallaz_TP_1/commit/8626fabbfa9834d1d15bc302b4f4e3e0f9a76bf9>
2. <https://github.com/gianquito/UADER_IS2_Putallaz_TP_1/commit/285b70d78df750eeca70b274deb80566a9acbe8d>
3. <https://github.com/gianquito/UADER_IS2_Putallaz_TP_1/commit/c076e2fcc940b9df5f35bc1f9d18e6cc36aee310>
4. <https://github.com/gianquito/UADER_IS2_Putallaz_TP_1/commit/daf92840576453a795fedf78476d5fb123d28b10>
5. <https://github.com/gianquito/UADER_IS2_Putallaz_TP_1/commit/2b3d20d9d4583decd4122daf1237392d18a7cf64>
6. <https://github.com/gianquito/UADER_IS2_Putallaz_TP_1/commit/5df26956b738015400e6caf760af0e92b5edf5db>
7. El uso del patrón abstract factory puede ser útil en el desarrollo de videojuegos.

Una empresa crea juegos para diferentes plataformas como PC, PlayStation y Xbox. La empresa necesita desarrollar un motor de videojuegos que pueda manejar los diferentes sistemas de entrada (teclado y mouse, control de PlayStation, control de Xbox), así como los diferentes sistemas gráficos y de sonido

El patrón abstract factory se utilizará para manejar las diferencias entre plataformas de manera eficiente.

Se crea una interfaz "AbstractFactory" que define los métodos para crear objetos abstractos de sistemas de entrada, gráficos y sonido. Luego, se implementan varias factory concretas, una para cada plataforma (PCFactory, PlayStationFactory, XboxFactory), que producen objetos específicos para cada plataforma.

Cuando el motor de videojuegos necesita crear un objeto de sistema de entrada, gráficos o sonido, simplemente solicita a la fábrica concreta correspondiente que lo haga. Esto permite que el motor se adapte a las diferentes plataformas, ya que solo necesita cambiar la fábrica concreta para cambiar el comportamiento de la creación de objetos.