

# • Check list proyecto

# 📁 Proyecto POO Mejoras - SNIES-extractor

## 📊 Análisis de Datos Académicos del SNIES y Generación de Estadísticas

### ✨ Descripción General

El Sistema Nacional de Información de la Educación Superior (SNIES) en recopila una gran cantidad de 📊 sobre las instituciones de educación superior y los programas académicos que ofrecen. El objetivo de este proyecto es mejorar el **\*\*SNIES-extractor\*\***, un programa en 🖥️ C++ que ya fue desarrollado previamente, con el fin de aplicar mejores prácticas de desarrollo, incluyendo principios **\*\*GRASP\*\***, manejo de 🚫 excepciones, validaciones y una mejor organización del código.

Este programa debe permitir la consolidación de 📊 sobre 📝 inscritos, 📝 admitidos, nuevos matriculados, 📊 matriculados y 🎓 graduados de los programas académicos a lo largo de varios 📅 años, soportando variaciones en la estructura de los archivos CSV sin necesidad de conocerlas a priori. Los 📊 serán exportados en formato **\*\*CSV\*\***, **\*\*TXT\*\*** y **\*\*JSON\*\***.



## 🎯 Objetivos del Proyecto






### 🚀 Mejoras Funcionales


- El programa debe procesar archivos CSV descargados del SNIES que pueden tener variaciones en sus columnas a lo largo de los 📅 años. Los 📊 relevantes para analizar, sin importar la variación de columnas, son:

- **\*\*🏠 Datos de la Institución\*\***: Código de la institución, IES padre, institución principal o seccional, ID sector IES, sector IES, ID carácter, carácter IES, código del departamento







(IES), departamento de domicilio de la IES, código del municipio IES, municipio de domicilio de la IES.

- \*\* Datos del Programa Académico\*\*: Código SNIES del programa, programa académico, núcleo básico del conocimiento, ID nivel académico, nivel académico, ID nivel de formación, nivel de formación, ID metodología, metodología, ID área, área de conocimiento, ID núcleo, núcleo básico del conocimiento (NBC), ID CINE campo amplio, descripción CINE campo amplio, ID CINE campo específico, descripción CINE campo específico, ID CINE código detallado, descripción CINE código detallado, código del departamento (programa), departamento de oferta del programa, código del municipio (programa), municipio de oferta del programa.

- \*\* Datos de los Estudiantes\*\*: Inscritos, admitidos, nuevos  matriculados,  graduados, género,  año, semestre.

- Debe soportar cualquier rango de  años y cualquier programa académico registrado en el SNIES.





- El sistema debe generar archivos de salida en formato \*\*CSV\*\*, \*\*TXT\*\* y \*\*JSON\*\*, aplicando polimorfismo y herencia para gestionar los diferentes formatos de salida.




- \*\* Recomendación\*\*: Se sugiere utilizar un \*\*mapa\*\* en lugar de un vector para la relación entre los programas académicos y los consolidados, ya que permite una mejor gestión cuando la cantidad de  varía o hay claves específicas para cada programa y  año. La información entre programas académicos puede variar; por ejemplo, puede haber semestres en los que no hubo aspirantes, nuevos  matriculados o  graduados, y el sistema debe soportar estos casos sin generar errores.

- El delimitador de caracteres podría cambiar, haga que su programa pueda cambiar fácilmente el delimitador de coma por un punto y coma, dos puntos, o un punto.

- Todos los requerimientos de la primera versión del proyecto deben funcionar.

### ### Mejoras de Calidad de Código

- Aplicar los principios \*\*GRASP\*\* ( experto,  creador,  bajo acoplamiento,  alta cohesión) en la estructura de clases para mejorar la modularidad y reducir el acoplamiento entre clases.

- Revisar el proyecto con la lista de chequeo disponible en el archivo de markdown **\*\*CHECKLIST.md\*\*** para asegurar que se cumplan los requerimientos de calidad de código.
- **\*\*🚫 Manejo de excepciones\*\***: El programa debe manejar adecuadamente los errores relacionados con la lectura de archivos, entradas inválidas y otros posibles fallos durante el proceso de consolidación de .
- **\*\*✂ Eliminación de números mágicos\*\***: Reemplazar todos los valores numéricos directos en el código por constantes significativas que mejoren la legibilidad.
- **\*\*✅ Validaciones de datos\*\***: Implementar validaciones sólidas para las entradas del usuario, como el rango de  años, nombres de archivos de salida y verificación de formatos de .
- Usar un **\*\*🔧 linter\*\*** para mejorar el formato y estilo del código, aplicando los principios de **\*\*código limpio\*\*** para asegurar que el código sea fácil de leer y mantener.
- **\*\*🔄 Refactorización\*\***: Como práctica obligatoria, los estudiantes deben analizar la complejidad ciclomática y duplicación de código de la versión inicial utilizando herramientas como **\*\*SonarQube\*\*** o **\*\*CppCheck\*\***. Las áreas problemáticas deben ser refactorizadas para mejorar la eficiencia y organización del código.

## ## 👤 Roles dentro del Equipo

- Un miembro del equipo debe asumir el rol de **\*\*🧑💻 Líder Técnico\*\***. Esta persona tendrá menos tareas en términos de codificación, pero será responsable de revisar la calidad del código fuente, realizar las revisiones de código a través de **\*\*Pull Requests\*\*** y hacer merge con la rama principal. También revisará la ejecución del proyecto en términos de la resolución de las funcionalidades y mejoras pendientes.
- Otro miembro del equipo asumirá el rol de **\*\*📄 Calidad y Documentación\*\***, asegurándose de que el código tenga documentación relevante y pertinente. Se sugiere explorar una herramienta como **\*\*Doxygen\*\***, pero no es indispensable.
- Todos los miembros del equipo deben participar activamente en la codificación y ser responsables de diferentes partes del proyecto.

## ## 📁 Organización del Proyecto en GitHub

- Utilizar el esquema de ramas sugerido por **GitHub Flow**. Cada miembro del equipo debe crear una rama de trabajo por cada funcionalidad o mejora que desee implementar.
- No se deben eliminar las ramas al hacer **merge** con la rama principal, para facilitar la revisión posterior del trabajo realizado.
- Se espera que el proyecto tenga como mínimo **5 pull requests** y **20 issues** que reflejen la gestión de tareas en la ejecución del proyecto.

## ## ⚙️ Integración Continua (CI/CD)

- Implementar **GitHub Actions** para configurar una integración continua (CI) que compile el proyecto automáticamente al realizar un commit o merge en la rama principal. Esto ayudará a verificar que el proyecto se mantenga funcional en todo momento.


## ## 📁 Entregables

📄 **Informe de Refactorización y Análisis de Código**: Documento en formato Markdown donde los estudiantes expliquen las áreas del código que identificaron para refactorizar, con base en el análisis de herramientas como **SonarQube** o **CppCheck**. Este documento debe explicar qué mejoras hicieron, cómo afectaron la complejidad ciclomática, la duplicación y las mejores prácticas, y cómo mejoraron la organización o eficiencia del código.

🎥 **Video de Presentación del Proyecto**: Video de 6 minutos donde se muestre el proyecto funcionando y se explique cómo se abordaron las mejoras implementadas. Cada miembro del equipo debe participar y explicar su contribución. Subir el link del video en los entregables del proyecto.

📊 **Diagrama UML actualizado** como Mermaid dentro del proyecto.


💻 **Código fuente del programa**. Utilizar GitHub Classroom para iniciar el proyecto.


 README en el repositorio\*\* (Manual técnico): Con presentación general del proyecto, imágenes y textos que muestren el cumplimiento de los requerimientos, diagrama de clases UML en formato .MERMAID incluido en el README (uno por equipo).

### Requerimientos Técnicos

- Uso continuo de \*\*Git\*\* para mantener el historial del proyecto con comentarios claros sobre los cambios.



### ## Fechas de Entrega



 Entrega parcial\*\*: 15 de octubre de 2024, que debe incluir avances importantes en las mejoras planteadas.

 Entrega final\*\*: 23 de octubre de 2024, 11:59 pm, vía GitHub.


### ## Condiciones Adicionales

- Los entregables deben ser subidos al repositorio de \*\*GitHub\*\* del equipo.

 Informe de Autoevaluación\*\*: Cada estudiante debe subir un informe individual en PDF a Brightspace, explicando qué problemas enfrentó en el proyecto, qué aprendió, y cómo usó la  IA en el desarrollo del proyecto, incluyendo beneficios y dificultades.

 Sustentación\*\*: Las sustentaciones serán entre el 24 y el 25 de octubre de 2024 de forma virtual, previo acuerdo de cita con la profesora. Es importante tener un  disponible para hacer modificaciones al proyecto durante la sustentación.

- Todos los miembros del equipo deben participar en todas las etapas del desarrollo (diseño, codificación, documentación y pruebas funcionales).

 IA\*\*: Los estudiantes pueden hacer uso de IA en el proceso de desarrollo del proyecto siempre y cuando puedan explicar todo lo que está incluido en la solución y las decisiones de diseño tomadas.

## Autoevaluación y Coevaluación

Cada estudiante debe completar un informe de autoevaluación y coevaluación, calificando a sus compañeros en los siguientes elementos:

### Colaboración y trabajo en equipo

- **\*\*Excelente (5)\*\***: El estudiante participa activamente y mejora la dinámica del equipo mediante una colaboración efectiva, mostrando respeto y apertura hacia las ideas de los demás.
- **\*\*Bueno (4)\*\***: Participa regularmente y colabora bien, aunque en ocasiones puede ser pasivo. Respetuoso con los compañeros la mayoría del tiempo.
- **\*\*Adecuado (3)\*\***: Participa sin tomar un rol activo, realiza lo mínimo necesario para no obstaculizar el equipo. Puede mejorar en respeto y colaboración.
- **\*\*Insuficiente (2)\*\***: Participación mínima o negativa, a menudo no colabora o desestima a los demás, afectando negativamente la dinámica del equipo.

### Responsabilidad y compromiso

- **\*\*Excelente (5)\*\***: Cumple con todas las tareas asignadas a tiempo, mostrando un alto nivel de compromiso y capacidad para tomar iniciativas adicionales.
- **\*\*Bueno (4)\*\***: Generalmente cumple con las tareas en los plazos establecidos y muestra un buen nivel de compromiso.
- **\*\*Adecuado (3)\*\***: Cumple con las tareas básicas, pero raramente toma iniciativas adicionales y a veces no cumple con los plazos.
- **\*\*Insuficiente (2)\*\***: Falta frecuentemente a sus responsabilidades, mostrando poco compromiso y afectando el rendimiento del equipo.

### Contribución al Desarrollo del Trabajo

- **\*\*Excelente (5)\*\***: Aporta ideas y soluciones que son esenciales para el proyecto. Participa de manera activa en la planificación y ejecución de tareas clave.
- **\*\*Bueno (4)\*\***: Realiza contribuciones que benefician al proyecto. Participa en la planificación y ejecución de tareas.
- **\*\*Adecuado (3)\*\***: Realiza contribuciones que cumplen con los requisitos básicos del proyecto. Participación regular en la planificación y ejecución de tareas.
- **\*\*Insuficiente (2)\*\***: Realiza pocas o ninguna contribución al desarrollo del trabajo. Participa poco o nada en la planificación y ejecución de tareas.

#### Uso de asistentes de IA en el desarrollo del proyecto (Usarlo solo en la autoevaluación)

- **\*\*Excelente (5)\*\***: Utiliza el asistente de codificación de manera estratégica, integrándolo efectivamente en el flujo de trabajo para mejorar significativamente la productividad y la calidad del código. Demuestra una comprensión profunda del código generado, realizando adaptaciones precisas que mejoran el proyecto. Muestra un juicio crítico excelente, seleccionando cuándo y cómo utilizar el asistente para optimizar resultados.
- **\*\*Bueno (4)\*\***: Emplea el asistente de codificación regularmente con una integración generalmente efectiva, contribuyendo positivamente al desarrollo del proyecto. Comprende bien el código generado y realiza ajustes necesarios para alinear el código con los objetivos del proyecto. Demuestra un buen juicio al evaluar la utilidad y precisión del código sugerido, aunque hay espacio para una evaluación más crítica en algunas situaciones.
- **\*\*Adecuado (3)\*\***: Utiliza el asistente de codificación de forma inconsistente, integrándolo solo parcialmente en el proyecto. Entiende en términos generales el código generado, pero no siempre realiza las modificaciones necesarias para que se ajuste completamente a las necesidades del proyecto. A veces depende demasiado de las sugerencias del asistente sin una evaluación crítica suficiente, afectando la optimización del desarrollo.
- **\*\*Insuficiente (2)\*\***: Hace un uso esporádico e inefectivo del asistente de codificación, con poca evidencia de que su uso contribuya al proyecto. Incorpora código sin entenderlo adecuadamente, resultando en problemas de integración y funcionalidad. Falta un criterio crítico adecuado, aceptando sugerencias del asistente sin una evaluación pertinente de su aplicabilidad o corrección.

- \*\*🤝 Colaboración y trabajo en equipo\*\*
- \*\*📋 Responsabilidad y compromiso\*\*
- \*\*🔧 Contribución al desarrollo del trabajo\*\*
- \*\*🤖 Uso de asistentes de IA en el desarrollo del proyecto\*\*

La evaluación se basará en una rúbrica que considera la participación activa, el respeto hacia los demás, la responsabilidad, y el uso adecuado de asistentes de IA.