



**TECNOLÓGICO  
DE MONTERREY®**

## **MAESTRÍA EN INTELIGENCIA ARTIFICIAL APLICADA**

### **Materia:**

Proyecto Integrador

### **Avance 4**

### **EQUIPO 29**

Miguel Angel Favela Corella A00818504

Kevin Balderas Sánchez A01795149

José Manuel García Ogarrio A01795147

### **Sponsor del Proyecto**

- **Dr. Juan Arturo Nolazco** – Director del Data Science Hub, Tecnológico de Monterrey.

### **Colaborador Invitado**

- **Dr. Carlos Alberto Brizuela Rodríguez** – Investigador, CICESE (Centro de Investigación

## Contexto del proyecto

El presente proyecto tiene como objetivo aplicar herramientas de inteligencia artificial y biología computacional para el diseño y análisis de proteínas con potencial aplicación biomédica. Nuestro foco principal se centra en el estudio del receptor GLP-1R, una proteína de membrana que desempeña un papel esencial en la regulación de la glucosa y del metabolismo energético.

El GLP-1R es activado por el péptido GLP-1, lo que estimula la secreción de insulina y reduce el apetito. Por esta razón, este receptor constituye una diana terapéutica importante en el tratamiento de la diabetes tipo 2 y la obesidad. Comprender su estructura, sus interacciones y posibles moduladores ofrece oportunidades para el diseño racional de nuevas moléculas bioactivas.

Nuestro enfoque combina distintas plataformas computacionales de vanguardia: RFDiffusion, MPNN, AlphaFold, y próximamente Rosetta y REF15, con el propósito de obtener una visión integral del diseño estructural, la viabilidad funcional y las propiedades bioquímicas de las proteínas generadas.

## Herramientas

- **RFDiffusion**

RFDiffusion es un modelo generativo basado en el principio de difusión probabilística, diseñado para la creación de nuevas estructuras proteicas de forma controlada y coherente. Este sistema aprende patrones estructurales a partir de proteínas reales y posteriormente puede generar diseños que cumplan con restricciones espaciales o funcionales específicas.

En este proyecto, RFDiffusion se emplea para proponer estructuras candidatas que potencialmente interactúen con el receptor GLP-1R. La finalidad es generar proteínas que puedan actuar como ligandos o moduladores del receptor, conservando estabilidad estructural y compatibilidad con el entorno biológico.

- **ProteinMPNN**

ProteinMPNN sirve para procesar grafos moleculares, donde los nodos representan átomos y las aristas los enlaces químicos. Esta arquitectura permite modelar las interacciones locales y globales entre los componentes de una molécula o proteína.

Su función principal es predecir la secuencia de aminoácidos que mejor se ajusta a una estructura tridimensional predefinida. El modelo recibe como entrada una estructura generada por RFDiffusion y, mediante el intercambio de información entre los nodos del grafo (que representan los residuos de aminoácidos), aprende qué combinaciones son más estables y coherentes con la geometría original.

El resultado son una o varias secuencias candidatas que podrían adoptar la forma estructural deseada. Posteriormente, estas secuencias se validan mediante AlphaFold,

que predice si la secuencia realmente se pliega de manera similar a la estructura original.

- **AlphaFold**

AlphaFold, es una herramienta de predicción estructural que utiliza aprendizaje profundo para determinar la conformación tridimensional de proteínas a partir únicamente de su secuencia de aminoácidos.

Su integración en el proyecto tiene como propósito verificar la estabilidad estructural de las secuencias diseñadas con RFDiffusion y ProteinMPNN. Esta herramienta nos permite visualizar las estructuras resultantes y evaluar si adoptan configuraciones energéticamente estables y coherentes con la función prevista. De esta forma, actúa como una herramienta de validación y análisis estructural.

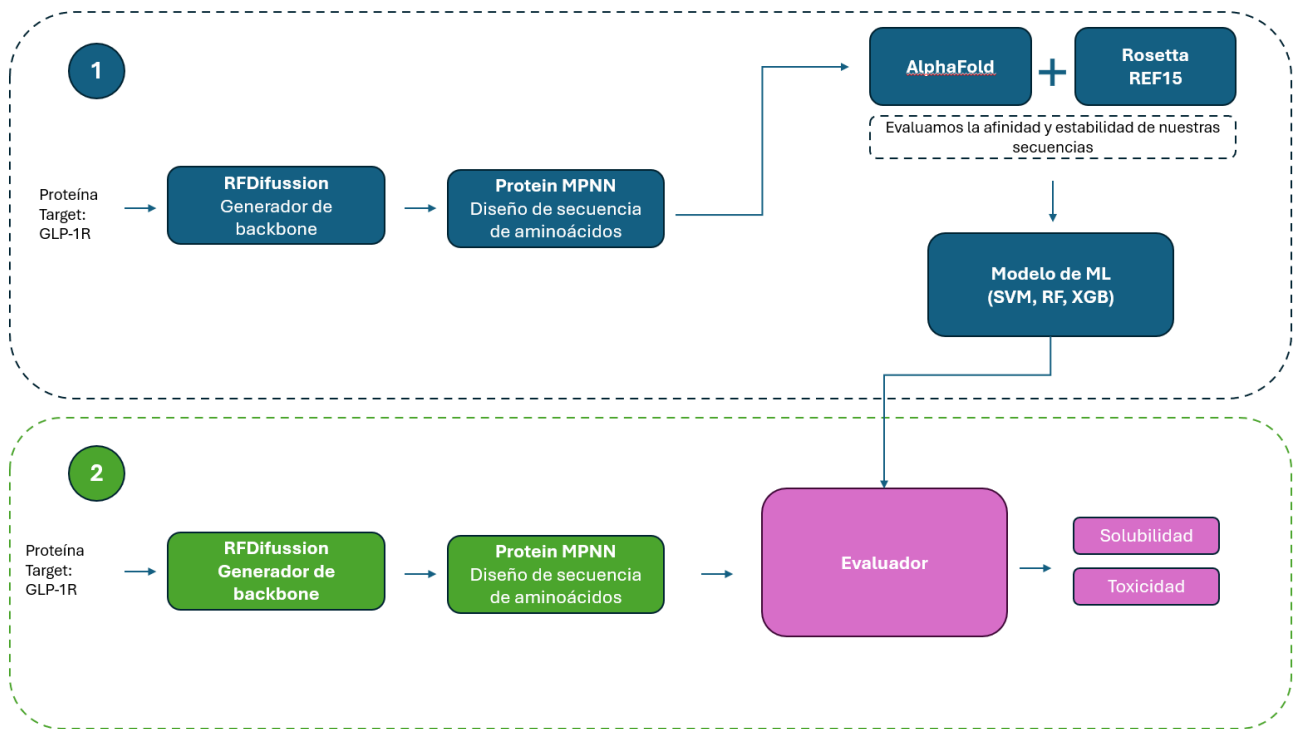
- **Rosetta y REF15**

Rosetta es una de las plataformas computacionales más avanzadas y versátiles para el modelado, diseño y simulación de proteínas. Permite realizar predicciones estructurales, análisis de estabilidad, acoplamiento molecular y optimización energética de biomoléculas. Su funcionamiento se basa en un conjunto de funciones de energía que estiman la estabilidad y la plausibilidad física de una estructura proteica.

Dentro de Rosetta, una de las funciones energéticas más utilizadas es REF15. Esta función integra parámetros físicos y estadísticos que permiten calcular con precisión las energías de interacción, enlaces de hidrógeno, fuerzas hidrofóbicas, interacciones electrostáticas, y energías de solvatación.

En el contexto de este proyecto, Rosetta y REF15 se emplearán conjuntamente para analizar y refinar las proteínas diseñadas mediante RFDiffusion y ProteinMPNN. El objetivo es evaluar no solo la estabilidad estructural y energética, sino también incorporar criterios adicionales relacionados con la toxicidad y la solubilidad de las proteínas generadas.

## Etapas y planificación.



## Descripción del Flujo de Trabajo

El flujo general del proyecto se divide en dos etapas principales, representadas en el diagrama anterior. Ambas tienen como punto de partida la proteína target GLP-1R, sobre la cual se generan y evalúan nuevas secuencias peptídicas con potencial afinidad estructural y funcional.

### Etapa 1: Generación y evaluación inicial de secuencias

En la primera fase del proyecto (azul), se utiliza RFDifussion como generador del backbone proteico, es decir, la estructura tridimensional base sobre la cual se diseñarán nuevas secuencias. Posteriormente, ProteinMPNN predice en este caso como ejemplo ~1,000/5,000 secuencias de aminoácidos que podrían adoptar dicha estructura.

Estas secuencias se someten a una evaluación estructural y energética mediante AlphaFold y Rosetta (REF15), con el fin de determinar propiedades como la afinidad, estabilidad y compatibilidad estructural.

El principal desafío en esta etapa es el alto costo computacional asociado al cálculo detallado de estas propiedades para miles de secuencias, lo que limita la escalabilidad del proceso.

Para superar esta limitación, se propone entrenar un modelo de ML, empleando algoritmos como SVM, Random Forest o XGBoost, usando un subconjunto representativo de las secuencias generadas.

Este modelo actuará como un predictor de afinidad y estabilidad, reduciendo significativamente el tiempo necesario para evaluar grandes volúmenes de datos sin sacrificar precisión.

## **Etapas 2: Evaluación masiva mediante modelo predictivo**

En la segunda fase (indicada en verde y rosa), se repite el proceso de diseño estructural mediante RFDiffusion y ProteinMPNN, pero esta vez a mayor escala, generando muchas secuencias candidatas.

En lugar de evaluar individualmente cada una mediante simulaciones intensivas, las secuencias se procesan con el modelo evaluador desarrollado en la primera etapa. Este evaluador integra el modelo de ML y permite estimar de manera rápida propiedades clave como la solubilidad y toxicidad.

De esta forma, se priorizan las secuencias con mejores características, reduciendo la carga computacional y enfocando los recursos en aquellas con mayor potencial biotecnológico.

## **Anexo de Resultados Visuales y Evidencia Computacional**

### **1. Propósito del anexo**

Este anexo documenta de manera independiente los resultados visuales obtenidos durante la ejecución del pipeline computacional para el diseño de proteínas orientadas al tratamiento de la diabetes. Se presentan las capturas de ejecución, visualizaciones estructurales y métricas derivadas de los modelos RFDiffusion, ProteinMPNN y AlphaFold, con una descripción técnica clara y contextualizada.

### **2. Flujo visual de trabajo**

#### **2.1 Generación de esqueletos proteicos con RFDiffusion**

Imágenes correspondientes a la ejecución local del modelo RFDiffusion, implementado en VSCode mediante notebooks de Jupyter.

**Descripción técnica:** El modelo generó 10 esqueletos (“backbones”) en formato .pdb a partir de parámetros relacionados con péptidos candidatos para diabetes.

**Tiempo de procesamiento:** ~35 minutos en GPU GeForce GTX 1650.

**Visualizador utilizado:** Mol\* Viewer.

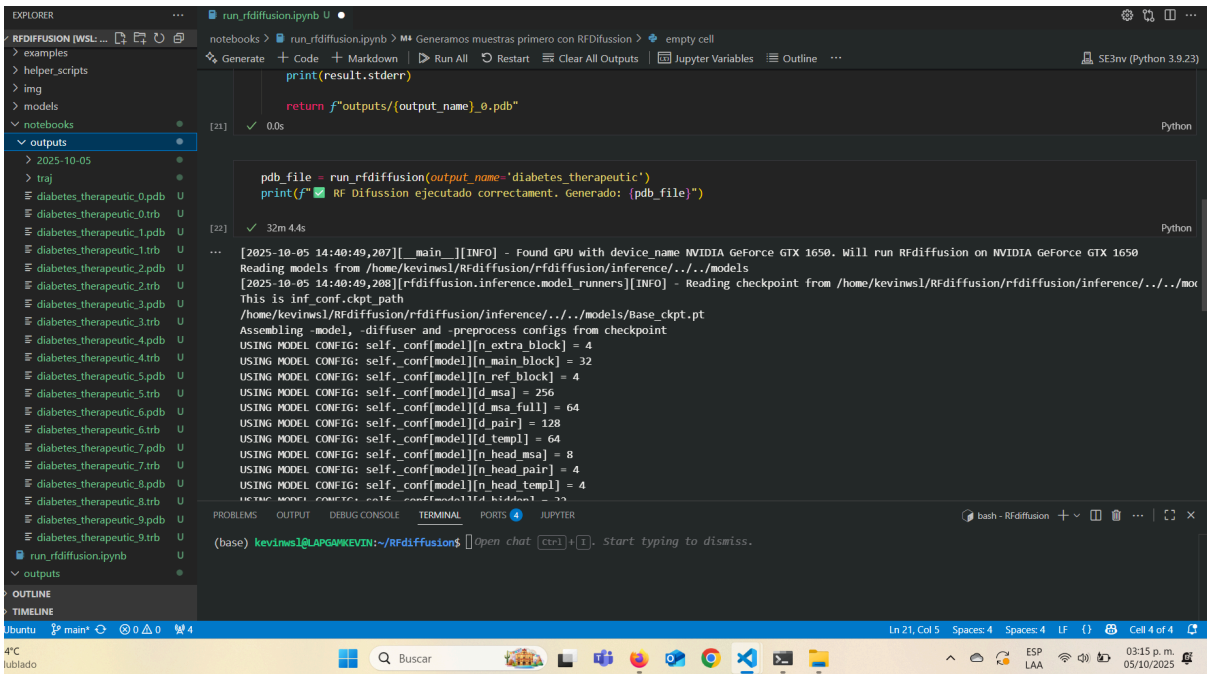


Figura 1. Ejecución del modelo RFdiffusion en entorno local.

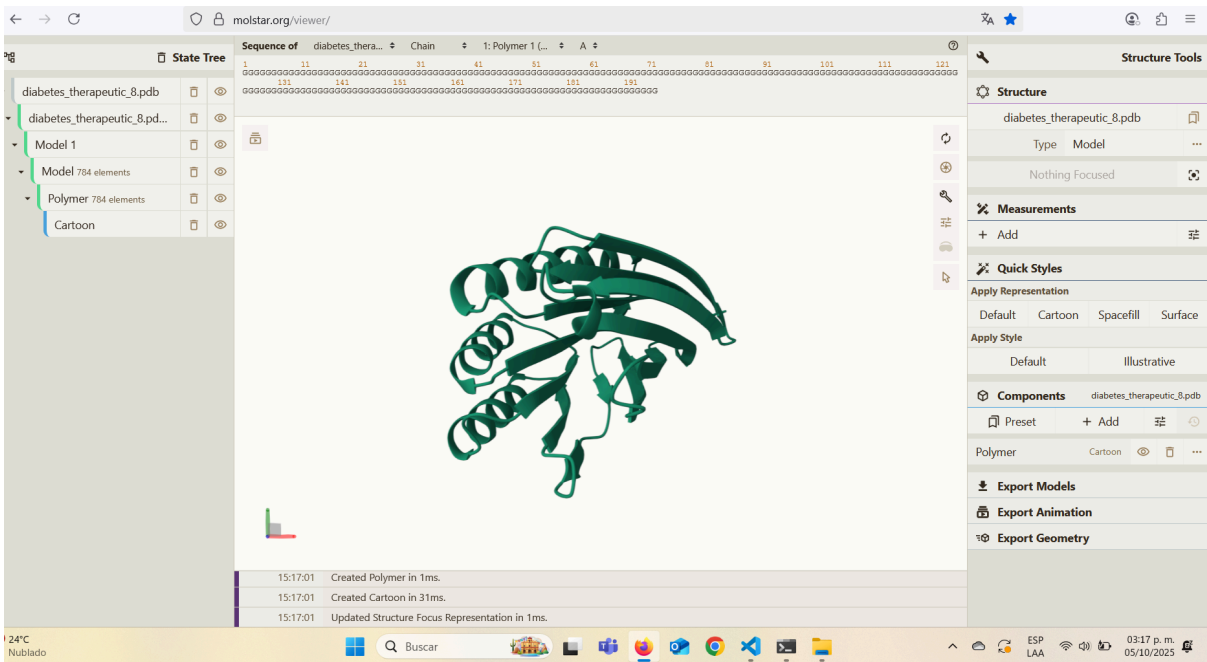
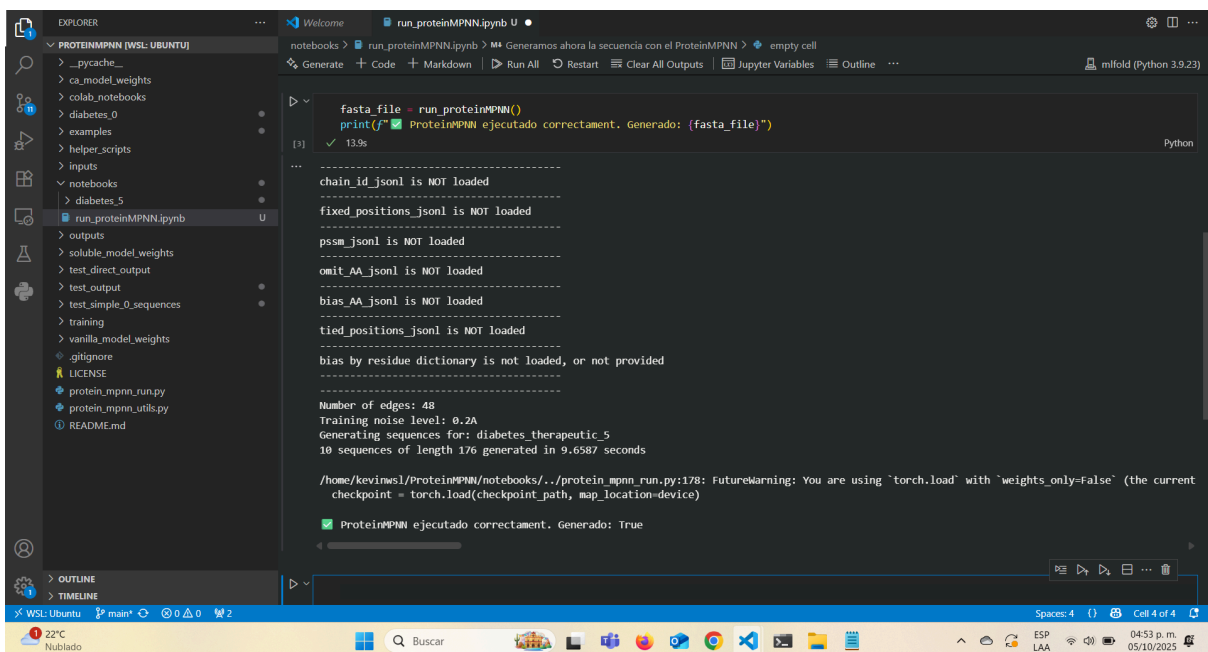


Figura 2. Visualización de estructuras generadas (.pdb) en MolViewer.

## 2.2 Generación de secuencias aminoacídicas con ProteinMPNN

**Descripción técnica:** Se utilizó el script `protein_mpnn_run.py` dentro del entorno Conda configurado. A partir de los archivos .pdb generados por RFdiffusion, se obtuvieron secuencias en formato .FASTA.

**Resultados:** Se generaron 10 secuencias con scores entre 1.009 y 1.2303 (valores de log-probabilidad negativa).



**Figura 3.** Ejecución de ProteinMPNN para obtención de secuencias.

[illegible]

**Figura 4.** Visualización de resultados y scores obtenidos.

### 2.3 Validación estructural con AlphaFold2

**Descripción técnica:** Se empleó colabfold\_batch para validar las secuencias generadas. Para cada muestra, se produjeron cinco modelos estructurales.

### Parámetros de rendimiento:

- Total de combinaciones: 55 validaciones.
- Duración total: ~1h 14min.
- Métricas utilizadas: pLDDT, pTM y RMSD.

### Resultados destacados:

- RMSD inicial: 9.16 Å (muestra compleja).
- RMSD optimizado en monómero simple: 0.31 Å.

```
kevin@LAPGAMKEVIN: ~  
2025-10-06 00:37:42,335 alphafold2_ptm_model_3_seed_000 recycle=2 pLDDT=78.6 pTM=0.653 tol=1  
2025-10-06 00:38:01,057 alphafold2_ptm_model_3_seed_000 recycle=3 pLDDT=76.8 pTM=0.64 tol=0.278  
2025-10-06 00:38:01,057 alphafold2_ptm_model_3_seed_000 took 76.2s (3 recycles)  
2025-10-06 00:38:21,357 alphafold2_ptm_model_4_seed_000 recycle=0 pLDDT=72.2 pTM=0.585  
2025-10-06 00:38:40,316 alphafold2_ptm_model_4_seed_000 recycle=1 pLDDT=78.8 pTM=0.669 tol=1.07  
2025-10-06 00:38:59,645 alphafold2_ptm_model_4_seed_000 recycle=2 pLDDT=76.9 pTM=0.639 tol=0.427  
2025-10-06 00:39:20,699 alphafold2_ptm_model_4_seed_000 recycle=3 pLDDT=77.8 pTM=0.633 tol=0.221  
2025-10-06 00:39:20,700 alphafold2_ptm_model_4_seed_000 took 79.6s (3 recycles)  
2025-10-06 00:39:40,135 alphafold2_ptm_model_5_seed_000 recycle=0 pLDDT=82.1 pTM=0.72  
2025-10-06 00:40:00,743 alphafold2_ptm_model_5_seed_000 recycle=1 pLDDT=80.9 pTM=0.696 tol=0.287  
2025-10-06 00:40:20,067 alphafold2_ptm_model_5_seed_000 recycle=2 pLDDT=78.3 pTM=0.674 tol=0.157  
2025-10-06 00:40:39,467 alphafold2_ptm_model_5_seed_000 recycle=3 pLDDT=78.7 pTM=0.674 tol=0.168  
2025-10-06 00:40:39,467 alphafold2_ptm_model_5_seed_000 took 78.7s (3 recycles)  
2025-10-06 00:40:39,495 reranking models by 'plddt' metric  
2025-10-06 00:40:39,495 rank_001_alphafold2_ptm_model_1_seed_000 pLDDT=78.9 pTM=0.65  
2025-10-06 00:40:39,495 rank_002_alphafold2_ptm_model_5_seed_000 pLDDT=78.7 pTM=0.674  
2025-10-06 00:40:39,496 rank_003_alphafold2_ptm_model_2_seed_000 pLDDT=78 pTM=0.688  
2025-10-06 00:40:39,496 rank_004_alphafold2_ptm_model_4_seed_000 pLDDT=77.8 pTM=0.633  
2025-10-06 00:40:39,497 rank_005_alphafold2_ptm_model_3_seed_000 pLDDT=76.8 pTM=0.64  
2025-10-06 00:40:40,780 Query 7/11: T_0.1__sample_6__score_1.1905__global_score_1.1905__seq_recovery_0.0057 (length 176)  
2025-10-06 00:40:41,341 Sleeping for 5s. Reason: PENDING  
2025-10-06 00:40:46,787 Sleeping for 7s. Reason: RUNNING  
COMPLETE: 100% | 150/150 [elapsed: 00:13 remaining: 00:00]  
2025-10-06 00:41:14,367 alphafold2_ptm_model_1_seed_000 recycle=0 pLDDT=77.8 pTM=0.685  
2025-10-06 00:41:32,808 alphafold2_ptm_model_1_seed_000 recycle=1 pLDDT=81.8 pTM=0.743 tol=1  
2025-10-06 00:41:52,854 alphafold2_ptm_model_1_seed_000 recycle=2 pLDDT=81.1 pTM=0.739 tol=0.406  
2025-10-06 00:42:11,625 alphafold2_ptm_model_1_seed_000 recycle=3 pLDDT=81.5 pTM=0.738 tol=0.0987  
2025-10-06 00:42:11,625 alphafold2_ptm_model_1_seed_000 took 75.9s (3 recycles)  
2025-10-06 00:42:32,183 alphafold2_ptm_model_2_seed_000 recycle=0 pLDDT=78.1 pTM=0.713  
2025-10-06 00:42:51,044 alphafold2_ptm_model_2_seed_000 recycle=1 pLDDT=81.8 pTM=0.76 tol=1.23  
2025-10-06 00:43:10,007 alphafold2_ptm_model_2_seed_000 recycle=2 pLDDT=82.4 pTM=0.766 tol=0.456  
2025-10-06 00:43:30,670 alphafold2_ptm_model_2_seed_000 recycle=3 pLDDT=82.6 pTM=0.769 tol=0.24  
2025-10-06 00:43:30,671 alphafold2_ptm_model_2_seed_000 took 79.0s (3 recycles)  
2025-10-06 00:43:49,771 alphafold2_ptm_model_3_seed_000 recycle=0 pLDDT=77.8 pTM=0.719  
2025-10-06 00:44:10,156 alphafold2_ptm_model_3_seed_000 recycle=1 pLDDT=81.4 pTM=0.756 tol=0.705  
2025-10-06 00:44:30,268 alphafold2_ptm_model_3_seed_000 recycle=2 pLDDT=82.1 pTM=0.76 tol=0.179  
2025-10-06 00:44:49,046 alphafold2_ptm_model_3_seed_000 recycle=3 pLDDT=82.7 pTM=0.768 tol=0.117  
2025-10-06 00:44:49,047 alphafold2_ptm_model_3_seed_000 took 78.3s (3 recycles)  
2025-10-06 00:45:08,094 alphafold2_ptm_model_4_seed_000 recycle=0 pLDDT=80.5 pTM=0.74  
2025-10-06 00:45:28,461 alphafold2_ptm_model_4_seed_000 recycle=1 pLDDT=82.8 pTM=0.773 tol=1.69
```

Figura 5. Ejecución de AlphaFold y generación de modelos.



Figura 6. Superposición de estructuras generadas (RFdiffusion vs AlphaFold).

## 2.4 Integración de pipeline

Flujo de integración logrado:

RFdiffusion → ProteinMPNN → AlphaFold2

**Conclusión visual:** Se demuestra la viabilidad del pipeline completo para la generación y validación de candidatos peptídicos, dejando como etapa futura la optimización de parámetros y el entrenamiento con Rosetta.



## 2.5 Obtención energética con Rosetta

**Descripción técnica:** Se empleó el módulo de PyRosetta nativo de Python para evaluar las nuevas secuencias (archivos pdb) generados previamente por AlphaFold. El script `calculate_stability.py` crea una función de energía y luego la calcula en base a nuestro .pdb, obteniendo métricas útiles. Pero el segundo script `calculate_stability_completed.py` aplicó la FastRelax para así obtener métricas importantes como el RMSD que servirán para nuestros modelos futuros evaluadores.

Antes se presenta el pdb utilizado para esta prueba:

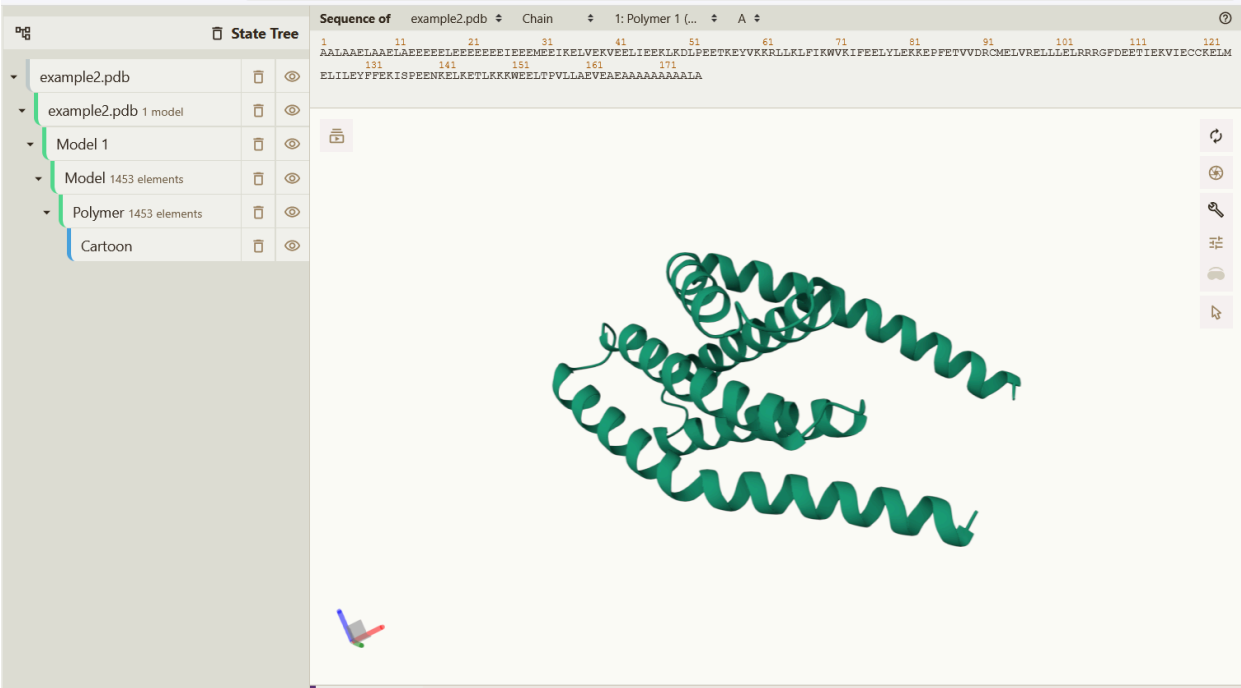
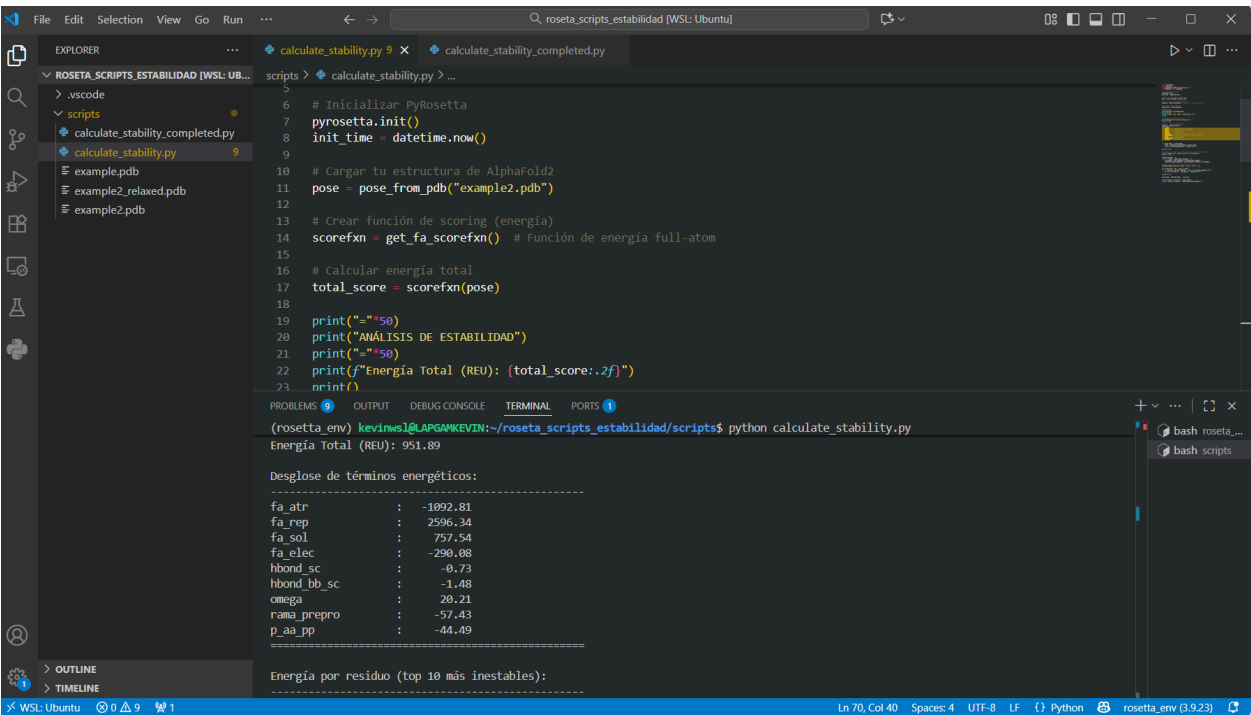


Figura 7. Archivo pdb utilizado para el proceso de Rosetta.



```
File Edit Selection View Go Run ...
roseta_scripts_estabilidad [WSL: Ubuntu]
EXPLORER
  ROSETTA_SCRIPTS_ESTABILIDAD [WSL: UB...
    .vscode
    scripts
      calculate_stability_completed.py
      calculate_stability.py
      example.pdb
      example2_relaxed.pdb
      example2.pdb
  ...
  calculate_stability.py x calculate_stability_completed.py
  scripts > calculate_stability.py > ...
5
6 # Inicializar PyRosetta
7 pyrosetta.init()
8 init_time = datetime.now()
9
10 # Cargar tu estructura de AlphaFold2
11 pose = pose_from_pdb("example2.pdb")
12
13 # Crear función de scoring (energía)
14 scorefxn = get_fa_scorefxn() # Función de energía full-atom
15
16 # Calcular energía total
17 total_score = scorefxn(pose)
18
19 print(f"50")
20 print("ANÁLISIS DE ESTABILIDAD")
21 print(f"50")
22 print(f"Energía Total (REU): {total_score:.2f}")
23 print()
24
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
(rosetta_env) kevin@LAPGAMKEVIN:~/roseta_scripts_estabilidad/scripts$ python calculate_stability.py
Energía Total (REU): 951.89

Desglose de términos energéticos:
-----
fa_atr      : -1092.81
fa_rep      : 2596.34
fa_sol      : 757.54
fa_elec     : -290.08
hbond_sc    : -0.73
hbond_bb_sc : -1.48
omega       : 20.21
rama_prepro : -57.43
p_aa_pp     : -44.49
-----

Energía por residuo (top 10 más inestables):
```

Figura 8. Ejecución del script que calcula función de energía full-atom

```
=====
Energía por residuo (top 10 más inestables):
-----
Residuo#  Tipo  Energía (REU)
      65  LYS      174.40
      22  GLU      173.90
      29  GLU      98.55
      72  LYS      96.83
     108  PHE      73.63
      55  THR      70.00
     106  ARG      61.14
      58  TYR      58.10
     132  PHE      40.43
     137  PRO      28.23
=====
Tiempo de ejecución: 0:00:00.697646
Tiempo en segundos: 0.697646
(rosetta env) kevinwsl@LAPGAMKEVIN:~/roseta_scripts_estabilidad/scripts$
```

Figura 9. Detalles técnicos de la función de energía full-atom

**Parámetros de rendimiento:**

- PDB utilizado: Archivo de 171 cadenas
- Duración total: 0.69 segundos
- Métricas utilizadas: Atracción de van der Waals , Solvatación, Electrostatica

**Resultados destacados:**

- Resumen de métricas a nivel global o general del pdb utilizado
- Análisis de tiempo utilizado durante su ejecución

Ahora se presenta el segundo script creado donde ya usamos Fast-Relax y obtenemos más métricas importantes.

```
(rosetta_env) kevinws1@LAPGAMKEVIN:~/roseta_scripts_estabilidad/scripts$ python calculate_stability_completed.py
(C) Copyright Rosetta Commons Member Institutions

NOTE: USE OF PyRosetta FOR COMMERCIAL PURPOSES REQUIRES PURCHASE OF A LICENSE
See LICENSE.PyRosetta.md or email license@uw.edu for details

PyRosetta-4 2025 [Rosetta PyRosetta4.Release.python39.ubuntu.cxx11thread.serialization 2025.39+release.c389ea27189ed431c5f30718cf7d86843f
eeab8e 2025-09-24T10:26:10] retrieved from: http://www.pyrosetta.org
Ejecutando FastRelax para optimizar estructura...

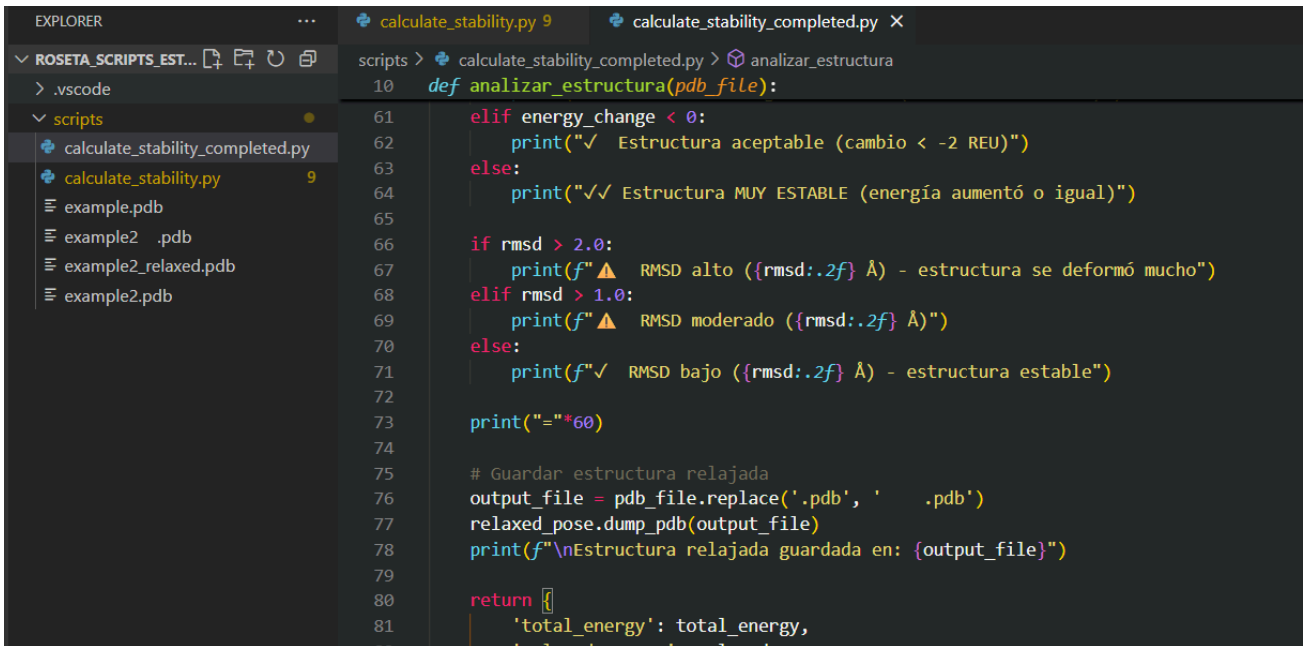
=====
RESUMEN DE ESTABILIDAD
=====
Archivo: example2.pdb
Número de residuos: 176

Energía inicial:      951.89 REU
Energía post-relax:   -556.77 REU
Cambio energético:    -1508.66 REU
Energía/residuo:      5.41 REU/res
RMSD (Cα):           2.22 Å

INTERPRETACIÓN:
-----
⚠ Estructura MUY INESTABLE (cambio > -5 REU)
⚠ RMSD alto (2.22 Å) - estructura se deformó mucho
=====

Estructura relajada guardada en: example2.pdb
Tiempo de ejecución: 0:01:11.347443
Tiempo en segundos: 71.347443
(rosetta_env) kevinws1@LAPGAMKEVIN:~/roseta_scripts_estabilidad/scripts$
```

Figura 10. Resultados luego de aplicar Fast-Relax.



The image shows a VS Code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'ROSETTA\_SCRIPTS\_EST...' with a 'scripts' folder containing 'calculate\_stability\_completed.py', 'calculate\_stability.py', and several PDB files. The code editor shows the 'calculate\_stability\_completed.py' script, which defines a function 'analizar\_estructura(pdb\_file)'. The function checks the energy change and RMSD of a structure. If the energy change is less than 0, it prints 'Estructura aceptable'. If the energy change is greater than or equal to 0, it prints 'Estructura MUY ESTABLE'. It also checks the RMSD: if it's greater than 2.0, it prints 'RMSD alto'; if it's greater than 1.0, it prints 'RMSD moderado'; otherwise, it prints 'RMSD bajo'. The function also saves the relaxed structure to a new PDB file.

```
def analizar_estructura(pdb_file):
    61     elif energy_change < 0:
    62         print("✓ Estructura aceptable (cambio < -2 REU)")
    63     else:
    64         print("✓ Estructura MUY ESTABLE (energía aumentó o igual)")
    65
    66     if rmsd > 2.0:
    67         print(f"⚠ RMSD alto ({rmsd:.2f} Å) - estructura se deformó mucho")
    68     elif rmsd > 1.0:
    69         print(f"⚠ RMSD moderado ({rmsd:.2f} Å)")
    70     else:
    71         print(f"✓ RMSD bajo ({rmsd:.2f} Å) - estructura estable")
    72
    73     print("="*60)
    74
    75     # Guardar estructura relajada
    76     output_file = pdb_file.replace('.pdb', 'relaxed.pdb')
    77     relaxed_pose.dump_pdb(output_file)
    78     print(f"\nEstructura relajada guardada en: {output_file}")
    79
    80     return {
    81         'total_energy': total_energy,
```

Figura 11. Estructura de la etapa de Rosetta.

**Parámetros de rendimiento:**

- PDB utilizado: Archivo de 171 cadenas.
- Duración total: 1.11 minutos.
- Método utilizado: Fast-Relax

**Resultados destacados:**

- Energía inicial, Energía Post-Relax, Cambio Energético, RMSD
- Análisis de tiempo utilizado durante su ejecución

Por último , en esta etapa se puede concluir que PyRosseta es un módulo muy completo para calcular métricas relacionadas con la energía, así como el método Fast-Relax, el cual es un protocolo que optimiza la geometría molecular. Es decir, luego de los resultados obtenidos mediante AlphaFold, puede existir detalles en su geometría de enlace (no óptima) o energías locales subóptimas. Así, podemos decir que tenemos ya el siguiente flujo logrado:

#### **Flujo de integración logrado:**

RFdiffusion → ProteinMPNN → AlphaFold2 → Rosseta

Meta para la próxima semana:

- Empezar a proponer y ver datos para alimentar los modelos predictores
- Decidir si serán 1 o 2 modelos
- Ver posible tratamiento de datos
- Examinar datos de entrada y número aproximado de muestras para el modelo.

## **Bibliografía**

- Jumper, J., Evans, R., Pritzel, A., et al. (2021). *Highly accurate protein structure prediction with AlphaFold*. Nature, 596(7873), 583–589.
- Watson, J. L., Juergens, D., Bennett, N. R., et al. (2023). *De novo design of protein structure and function with RFdiffusion*. bioRxiv. <https://doi.org/10.1101/2023.03.20.533421>
- Anand, N., & Achim, T. (2022). *ProteinMPNN: A deep learning model for protein sequence design*. bioRxiv. <https://doi.org/10.1101/2022.07.20.500902>