UAB
Universitat Autònoma de Barcelona

Dipòsit digital de documents de la UAB

---

This is the **published version** of the bachelor thesis:

Ramos Segarra, Rubén; Benítez Fernández, Yolanda, tut. Generación del diseño de la interfaz gráfica de un programa basado en la arquitectura MVC mediante aprendizaje automático. 2025. (Enginyeria Informàtica)

---

This version is available at https://ddd.uab.cat/record/317561

under the terms of the  license

# Generació del disseny de la interfície gràfica d'un programa basat en l'arquitectura MVC mitjançant aprenentatge automàtic

Rubén Ramos Segarra

**Resum**—Aquest treball presenta una eina que genera interfícies gràfiques d'usuari a partir de la part del Model i Controlador, seguint l'arquitectura Model-Vista-Controlador i aplicant aprenentatge automàtic per construir la Vista. La proposta és innovadora i poc explorada, amb aplicacions reals en el desenvolupament de *software*. Es compara amb metodologies *low-code* i eines com ChatGPT, destacant-ne diferències i avantatges. El sistema es divideix en sis mòduls, cadascun amb una funció específica dins del procés de generació. Tot i les dificultats, especialment en la recollida de dades i ús de llibreries externes com Abstract Syntax Tree i Tkinter, l'eina desenvolupada representa una contribució útil i potencialment innovadora.

**Paraules clau**—Aprenentatge Automàtic, Arquitectura Model-Vista-Controlador, Generador, Interfície Gràfica d'Usuari.

**Abstract**—This project presents a tool that generates graphical user interfaces based on the Model and Controller components, following the Model-View-Controller architecture and applying machine learning to build the View. The proposal is innovative and scarcely explored, with real-world applications in software development. It is compared to low-code methodologies and tools like ChatGPT, highlighting differences and advantages. The system is divided into six modules, each with a specific function in the generation process. Despite challenges, especially in data collection and the use of external libraries such as Abstract Syntax Tree and Tkinter, the developed tool represents a useful and potentially innovative contribution.

**Index Terms**—Generator, Graphical User Interface, Machine Learning, Model-View-Controller Architecture.

## 1 Introducció – Context del Treball

El treball en qüestió es basa en la implementació d'una eina que generi el disseny d'una interfície gràfica d'usuari d'un programa d'acord amb la implementació de la part(s) de Model i la part(s) del Controlador, respectant l'arquitectura Model-Vista-Controlador i aplicant un model d'aprenentatge automàtic, amb la premissa de generar la interfície gràfica que correspon a la part (o parts) de Vista.

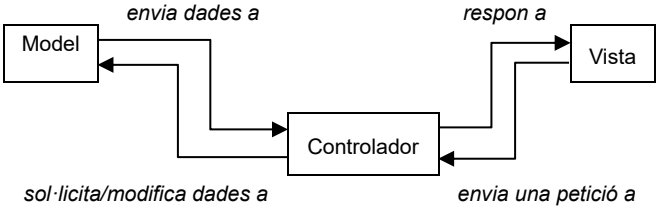


Fig. 1. Diagrama de l'arquitectura Model-Vista-Controlador.

Aquesta està expressament adreçada a desenvolupadors de *software* interessats en desenvolupar aplicacions basades en l'arquitectura anterior. Un dels principis motivacionals per a dur a terme el treball va ser l'originalitat de la proposta, que excepcionalment no havia estat explorada en altres treballs d'acord amb la cerca prèvia de l'autor. Altrament, es tracta d'una potencial solució a problemes de l'àmbit real, sobretot en el desenvolupament de *software* amb interfície gràfica, i que es presenta com una oportunitat d'innovació. Per tant, s'espera que aquest treball esdevingui útil i enriqueixi l'aportació de noves idees en l'àmbit de la informàtica.

Per tal de facilitar la comprensió i oferir una visió estructurada del treball, l'informe es divideix en les següents seccions:

- L'estat de l'art, on s'ofereix una revisió crítica i organitzada del coneixement existent dins del context del treball.
- Els objectius, on s'expressen les descripcions concretes que s'han d'assolir o complir en el treball.
- La metodologia, on s'indiquen les tècniques, els recursos i els actors que intervenen per assolir els objectius proposats anteriorment.
- La planificació, on es manifesta el calendari a seguir durant el transcurs del treball.
- L'anàlisi, on s'examinen aspectes crucials per a l'acompliment dels objectius i el correcte desenvolupament del treball.
- El desenvolupament, on s'explica el gruix de les fases d'implementació que conformen el projecte del treball.
- Les proves (o *tests*), on es posa a prova el funcionament i el rendiment del treball.
- Els resultats, on s'avalua el treball en la seva


- E-mail de contacte: 1496796@uab.cat
- Menció realitzada: Computació.
- Treball tutoritzat per: Yolanda Benítez Fernández (Àrea de Ciències de la Computació i Intel·ligència Artificial).
- Curs 2024/25.

totalitat, interpretant els mateixos seguits d'una discussió/reflexió.
- Les conclusions, on es posa èmfasi sobre els descobriments més importants i quina línia continuista ha de seguir de cara a futurs projectes.
- Les línies obertes que presenta el treball, amb possibilitat de ser explorades en altres projectes i/o estudis.

Addicionalment s'inclou una secció d'apèndix amb material il·lustratiu referent a aquest treball.

## 2 Estat de l'Art

LA contextualització prèvia del treball permet parar atenció a casos d'aplicació en el món real que guardin certa similitud. Un d'aquests es el mètode *low-code*, que permet desenvolupar aplicacions mitjançant una interfície d'usuari (mitjançant *software* de tercers) sense implementar codi manualment "[1]". Això redueix dràsticament el temps de desenvolupament, a més de tenir un alt nivell d'abstracció que permet accelerar la producció. Per tant, es pot dir que el *low-code* es comparable a la proposta del treball, donat que no requereix la interacció directa del desenvolupador a l'hora d'implementar codi, però amb la particularitat de que només s'aplica a la generació de codi de la interfície gràfica. Un altre exemple molt habitual i representatiu d'avui día són les eines d'aprenentatge automàtic com ChatGPT, Copilot o Deepseek, que poden generar codi d'una interfície gràfica fàcilment, encara que no totes ofereixen la mateixa qualitat a nivell resolutiu.

Ambdós casos anteriors són una bona mostra de les eines que fan servir empreses, entitats o institucions en el dia a dia per agilitzar processos i innovar en el desenvolupament de projectes, i que des d'una perspectiva semblant evoquen alguna afinitat amb la proposta d'aquest treball.

## 3 Objectius

ARRAN de les consideracions inicials exposades en el context del treball, es van plantejar uns objectius alineats amb les seves característiques:

- Generar la interfície gràfica que correspon a la part de Vista a partir de les parts de Model i Controlador (crític).
- Accelerar el temps que es triga a dissenyar la interfície gràfica en un 70% o de manera significativa (prioritari).
- Generar dissenys que tinguin coherència (opcional).
- Millorar la precisió dels dissenys que es generen un cop ha finalitzat la fase de construcció en un 10% (opcional).

La definició d'aquests objectius, per tant, estableix inicialment el rerefons o nivell d'assoliment del treball, prioritzant el desenvolupament dels objectius més crítics i, addicionalment, la inclusió d'altres de caire secundari que poden aportar valor afegit al treball.

## 4 Metodologia

EL treball ha de seguir una metodologia específica per assolir una bona gestió i desenvolupament del mateix. S'ha escollit aplicar la metodologia Agile Software Development, basada en l'enfocament eficaç per a projectes que necessiten flexibilitat i adaptabilitat, especialment quan canvia en termes d'abast o requisits "[2]". Aquesta consta dels següents processos "[3]":

1. La fase de recollida de requisits, on es recullen i prioritzen els requisits.
2. La fase de planificació, on es crea un pla per lliurar el programari, incloses les funcions que es lliuraran en cada iteració.
3. La fase de desenvolupament, on el desenvolupador (o desenvolupadors) treballen per crear el programari, utilitzant iteracions freqüents i ràpides.
4. La fase de testejat, on el programari es testeja a fons per garantir que es compleixen els requisits i és d'alta qualitat.
5. La fase de lliurament, on el programari es lliurat.
6. La fase de manteniment, on el programari es manté per garantir que segueix satisfent les necessitats definides des del principi.

L'eina escollida per seguir aquesta metodologia es Trello, degut a la seva flexibilitat, l'àmplia selecció d'elements visuals i la seva gratuïtat "[4]". També proporciona accés multiplataforma.

## 5 Planificació

EN aquesta secció es mostra la planificació del treball, des del començament fins al final, dividida en diferents fases, activitats i tasques. El cicle de vida del treball es divideix en quatre etapes diferenciades "[5]":

- L'inici, on es defineixen els objectius, l'abast i els lliuraments.
- La planificació, on s'estableixen les pautes per garantir la seva finalització. Aquesta pot desenvolupar-se durant tot el cicle de vida.
- L'execució i el control, on es posa en marxa el desenvolupament i posteriorment es fa un anàlisi dels resultats obtinguts.
- El tancament, que assegura la correcta finalització de totes les activitats.

Per aquest treball s'ha dividit el projecte en tres fases generals: la fase inicial, que correspon a les activitats d'iniciació i planificació, la fase intermèdia, equivalent a l'execució i el control amb les activitats d'anàlisi, disseny i desenvolupament, i per últim la fase final, que es basa en els resultats i la finalització del mateix. La Fig. A1 mostra la captura de pantalla de Microsoft Project de la planificació del treball.

## 6 ANÀLISI

A la part d'anàlisi es precisa tot allò que ha esdevingut fonamental en la fase de desenvolupament. Per generar la interfície gràfica de qualsevol implementació, aquesta ha de seguir l'arquitectura Model-Vista-Controlador, constituïda per tres components:

- El Model, que representa el gruix de les dades que fa servir l'aplicació i gestiona les peticions o modificacions realitzades per el Controlador o la Vista.
- La Vista, que s'encarrega de visualitzar les dades de sortida del Model i permet la introducció de dades d'entrada al Controlador.
- El Controlador, que actua com a intermediari entre el Model i la Vista gestionant les dades d'entrada introduïdes per l'usuari, modifica el Model segons convingui i actualitza la Vista per reflectir els canvis en l'anterior component. Aquest conté gran part de la lògica de l'aplicació (com per exemple la transformació de les dades o la validació de les dades d'entrada).

El Model rep peticions i envia dades al Controlador, que rep les dades d'entrada de l'usuari i, segons les modificacions realitzades fins el moment, respon a la Vista actualitzant els valors de sortida. El funcionament bàsic de l'arquitectura Model-Vista-Controlador es pot apreciar a la Fig. 1. En la majoria de casos on s'aplica l'arquitectura Model-Vista-Controlador, independentment del llenguatge de programació que se'n faci ús, els components es representen mitjançant objectes de tipus classe (òbviament ha de seguir el paradigma de programació orientada a objectes). Cada classe haurà d'incloure els atributs i els mètodes pertinents segons el tipus de component que es vol representar en aquesta, incloent a més les relacions amb la resta de components (veure Fig. A2). Cal esmentar que l'arquitectura Model-Vista-Controlador pot ésser constituïda per més d'un d'aquests components i també poden subsistir diverses interrelacions, sempre i quan respectin el model d'arquitectura preestablert.

La implementació s'ha realitzat íntegrament mitjançant el llenguatge de programació Python, degut a la seva relativament poca complexitat, la gran escalabilitat que ofereix a múltiples plataformes i el seu ampli suport comunitari. D'acord amb les necessitats de desenvolupament de treball i la seva rellevància, se n'ha fet ús, principalment, de les següents llibreries:

- Tkinter, *framework* per defecte del paquet de llibreries de Python utilitzat per dissenyar interfícies gràfiques d'usuari "[6]".
- Pandas, eina d'anàlisi i manipulació de dades de codi obert "[7]".
- Abstract Syntax Tree, llibreria que conté funcions que generen l'arbre sintàctic d'un codi. Aquest arbre forma part del procés de compilació de Python, on cada node representa un objecte definit a la seva gramàtica "[8]".
- Scikit-Learn, llibreria que conté eines per a l'anàlisi predictiu de dades "[9]".
- LangID, eina autònoma d'identificació lingüística "[10]".

Pel que fa a la recollida de dades per a el conjunt d'entrenament, existeixen nombroses bases de dades o datasets especialitzats, tot i que de manera exclusiva no se n'ha trobat cap que acompleixi les necessitats del treball. Donat aquest cas particular, s'ha decidit dividir les dades en tres conjunts diferenciats (fitxers .xlsx):

- Dades reals basades en exemples de codi font que segueixen l'arquitectura Model-Vista-Controlador.
- Dades sintètiques generades amb ChatGPT mitjançant un *prompt* personalitzat.
- Dades d'*oversampling* per tal de millorar el desbalanceig de les classes objectiu.

Les dades del conjunt de prova o *test* s'extreuen durant l'escaneig del codi font de la part implementada pel desenvolupador, amb les mateixes característiques que el conjunt de dades d'entrenament.

## 7 DESENVOLUPAMENT

LA següent secció es centra en el desenvolupament íntegre del treball, on es detallaran aspectes com l'estructura principal, la fase d'aprenentatge i els mòduls que componen l'eina de generació.

### 7.1 Estructura principal

La fase de construcció representa el gruix de la part de desenvolupament, que consisteix en la implementació de diversos mòduls que formaran part del programa principal per assolir l'objectiu principal del treball, es a dir, generar una interfície gràfica. La fase es dividirà en sis mòduls:

- El programa principal (main.py), que conté el flux principal del programa, incloent una finestra per monitoritzar la configuració prèvia a la generació de la interfície gràfica.
- La càrrega de fitxers (load.py), que conté les funcions associades a la càrrega de fitxers de Python que inclouen el codi font basat en els components de Mòdul i Controlador i les dades en el format desitjat.
- L'escàner (scanner.py), que conté les funcions que serveixen per escanejar el codi mitjançant un anàlisi sintàctic (mòdul Abstract Syntax Tree) i que permeten l'extracció de les característiques desitjades per a la fase d'aprenentatge (concretament esdevindrà el conjunt de prova o *test*).
- El model d'aprenentatge (model.py), que conté les funcions que permeten entrenar un model específic un cop es tenen a l'abast les dades del conjunt d'entrenament i les de prova o *test* (llibreria Scikit-Learn).

- El mòdul de refinament (refiner.py), que conté funcions addicionals amb la finalitat de millorar els resultats obtinguts durant la fase d'aprenentatge per a posteriorment traduir-los en elements de la interfície gràfica, entre altres matissos.
- El generador (generator.py), que conté les funcions que tradueixen els resultats d'aprenentatge en el disseny de la interfície gràfica mitjançant codi font preestablert (mòdul Tkinter).

Juntament amb els mòduls existeix una carpeta anomenada /data on es trobaran les dades necessàries per a l'aplicació del mòdul d'aprenentatge automàtic, una carpeta /code on es desarà el codi font implementat pel desenvolupador i una carpeta /media on s'inclouen elements gràfics (en la seva majoria icones).

### 7.2 Fase d'aprenentatge

Una part significant del desenvolupament passa per l'aprenentatge que farà el model i que es traduirà en el disseny de la interfície gràfica d'usuari. Per tant, cal que les dades siguin suficientment representatives, explicatives i orientatives per obtenir els resultats desitjats en l'aprenentatge. Com a característiques que es poden extreure, només s'ha d'observar l'aplicació de l'arquitectura Model-Vista-Controlador a nivell d'implementació. Se sap que el Controlador n'és el component que actua com a intermediari entre els component de Model i Vista, i que aquest permet la manipulació de les dades del primer interactuant des de la interfície del segon. Molt sovint, el Controlador inclou mètodes que, quan es criden, realitzen funcions específiques i que es troben en el mateix context d'ús. Se centrarà, per tant, en els mètodes del Controlador.

```
class ControladorEstudiant:
    ...
    def modificar_nom_estudiant(self, nom: str):
        self.model.set_nom(nom)

    def retornar_nom_estudiant(self) -> str:
        return self.model.get_nom()
    ...
```

Fig. 2. Exemple de codi font de dos mètodes de la classe ControladorEstudiant (Python).

Un exemple de com s'extreuen aquestes dades es mostra a la Fig. 2. Dins la classe ControladorEstudiant podem trobar els mètodes modificar_nom_estudiant() i retornar_nom_estudiant(). Amb la premissa de que aquests mètodes es traslladaràn a la Vista en forma de *widgets*, s'han d'identificar prèviament quines son les entrades, quins són els processos i quines seràn les sortides.

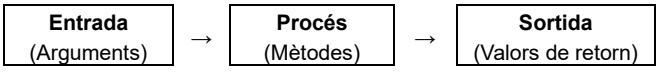


Fig. 3. Diagrama del model Entrada-Procés-Sortida.

Si ens fixem en la Fig. 3, l'observació anterior es molt similar a la que segueix el model Entrada-Procés-Sortida (*Input-Process-Output*) habitual en l'anàlisi de sistemes i l'enginyeria del *software*, a més d'altres àmbits relacionats "[11]". L'entrada equivaldria a els paràmetres que li passem al mètode modificar_nom_estudiant() (nom), el procés correspon a la crida del mètode retornar_nom_estudiant() (per exemple) i finalment la sortida es redueix a el valor de retorn self.model.get_nom() del mètode anterior.

La llista de *widgets* utilitzats per extrapolar el model Entrada-Procés-Sortida són els següents (veure Taula 1).

TAULA 1
ENTRADA, PROCÉS I SORTIDA DELS *WIDGETS* AMB ELS TIPUS DE DADES SUPORTATS

| *Widget* | Tipus de dades | *Input* | *Process* | *Output* |
|---|---|---|---|---|
| Label | int, float, bool, complex, str | No | No | Sí |
| Entry | complex, str | Sí | No | Sí |
| Button | - | No | Sí | No |
| Checkbutton | bool | Sí | No | No |
| Radiobutton | str | Sí | No | No |
| Scale | int, float | Sí | No | No |
| Spinbox | int, float | Sí | No | No |
| Combobox | str | Sí | No | No |
| Treeview | list, tuple, set, dict | Sí | No | Sí |

Sota la base anterior es decideix com es classificarà cada mètode, argument o valor de retorn repartit entre els diferents Controladors, atribuint a cadascun d'aquests una etiqueta associada a un *widget* d'entrada, de procés o de sortida. Com a característiques de les dades, es tindràn en compte les següents (veure Taula 2).

TAULA 2
CONJUNT DE CARACTERÍSTIQUES DE LES DADES D'ENTRENAMENT

| Nom | Descripció | Tipus (dtype) |
|---|---|---|
| Name | Nom del mètode/argument/valor de retorn. | object |
| Type | Tipus de dades del mètode/argument/valor de retorn. Aquest pot ésser int, float, bool, str, complex, list, tuple, set o dict (NaN si no és de cap tipus). | object |
| From | Valor mínim amb el qual es pot assignar un argument de tipus int o float. | float64 |
| To | Valor màxim amb el qual es pot assignar un argument de tipus int o float. | float64 |
| IsAnArgument | Si es True es tracta d'un argument. | bool |

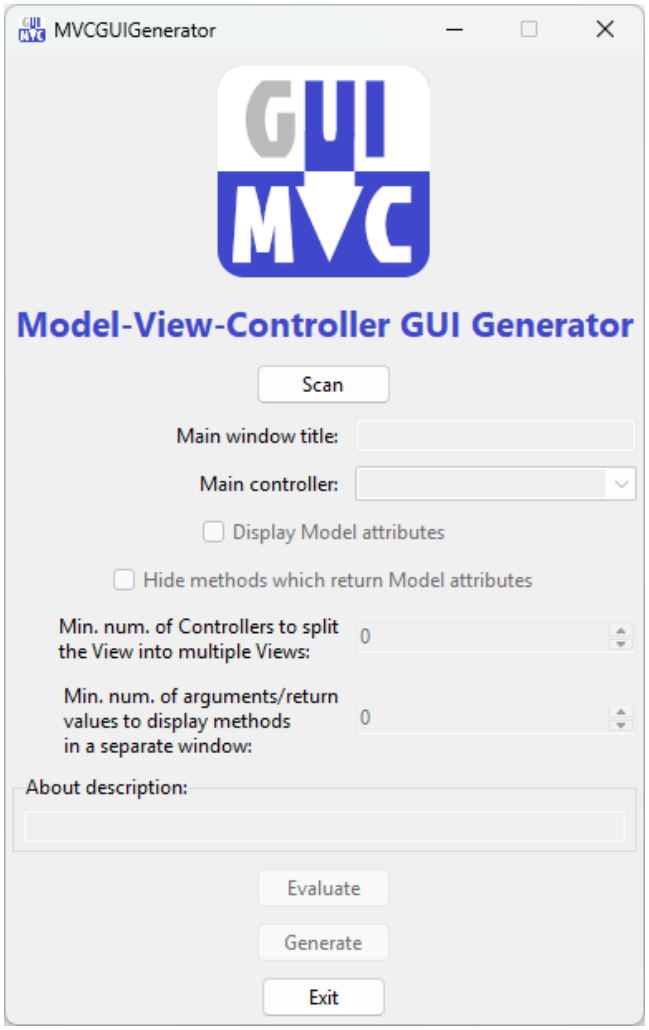


Fig. 4. Captura de pantalla de la finestra que apareix quan s'executa main.py.

| | | |
|---|---|---|
| IsAMethod | Si es True es tracta d'un mètode. | bool |
| IsAReturnValue | Si es True es tracta d'un valor de retorn. | bool |
| Argument-Name1-10 | Nom dels arguments que es passen a un mètode, si escau. | object |
| ArgumentType1-10 | Tipus de dades dels arguments que es passen a un mètode, si escau. | object |
| ReturnValueName1-10 | Nom dels valors de retorn d'un mètode, si escau. | object |
| ReturnValueType1-10 | Tipus de dades dels valors de retorn que es passen a un mètode, si escau. | object |
| DefaultValue | Valor per defecte d'un argument, si escau. | object |
| PossibleValues | Conjunt de valors possibles que pot adoptar un argument de tipus str, si escau. Els valors es troben separats per una coma. | object |
| BelongsTo | Nom del mètode al qual pertany un argument o valor de retorn. | object |
| ClassName | Nom de la classe a la qual pertany un mètode/argument/valor de retorn. | object |
| UsedByView | Si es True es que té comunicació directa amb la vista, en cas contrari es False. | bool |
| Widget | Nom de l'etiqueta del *widget*. | object |

Les implementacions realitzades a Python no requereixen l'especificació del tipus de dades. No obstant, en aquest cas si cal tenir en compte que en els mètodes es defineixen explícitament els tipus de dades mitjançant suggeriments (*type hints*) "[12]". En el cas de que es retornin múltiples valors de retorn (separats per una coma), el tipus de sortida s'ha definir com a tuple i incloure els tipus de dades corresponents a dins.

També s'inclou la possibilitat de definir valors màxims i mínims o possibles valors d'entrada mitjançant l'ús d'assercions associades a els arguments d'un mètode. Això es tindrà en compte en cas de que:

1. Es comparin arguments de tipus int o float amb un valor constant del mateix tipus mitjançant els operadors <, <=, >, >=.
2. Es comparin arguments de tipus str amb un valor constant del mateix tipus mitjançant l'operador ==, i separant múltiples comparacions amb l'operador or.
3. Es comparin arguments de tipus str amb un conjunt de valors constants que es troben a una llista mitjançant l'operador in.

Donat que el volum de les dades es inferior a les 100 mil mostres i es tracta d'un problema de classificació, s'ha fet servir el classificador de descens del gradient estocàstic (aprenentatge supervisat) "[13]". Durant el preprocessament, s'eliminen les columnes que no són rellevants per a l'aprenentatge, a més d'aplicar *one-hot encoding* a l'atribut Type (això ajuda a diferenciar les variables categòriques en valors binaris distribuïts en nous atributs, generats arran el conjunt de valors possibles en un únic atribut, i evitar problemes d'ordenació i freqüència de valors un cop s'apliqui el classificador de descens del gradient estocàstic) "[14]". Addicionalment, s'escalen totes les dades numèriques (columnes From i To) mitjançant la funció StandardScaler() i es creen característiques addicionals. En el moment d'entrenar el model, s'apliquen els següents hiperparàmetres mostrats a la Taula 3 (prèviament s'ha fet us de la funció get_best_hyperparameters() del mòdul model.py per obtenir els que asseguren un model més òptim).

TAULA 3
HIPERPARÀMETRES DEL MODEL D'APRENENTATGE (SGDCLASSIFIER)

| Hiperparàmetre | Valor |
|---|---|
| alpha | 0.001 |
| eta0 | 0.001 |
| learning_rate | 'optimal' |
| loss | 'log_loss' |
| penalty | 'l2' |
| max_iter | 1000 |

## 7.3 Mòduls

Per implementar l'eina de generació d'interfícies gràfiques d'usuari, la seva implementació es divideix en sis mòduls o fitxers: main.py, load.py, scanner.py, model.py, refiner.py i generator.py. Tot seguit s'explica amb detall el funcionament de cada mòdul.

### *7.3.1 Programa principal*

L'eina de generació ofereix una interfície gràfica per on passa gran part del flux principal. Aquesta permet configurar diferents aspectes que poden afectar el disseny i la visualització final des d'una finestra tal i com es mostra a la Fig. 4. Com a pas previ, l'usuari ha de prémer el botó "Scan" per tal d'escanejar el codi font implementat, fet que habilitarà la modificació de les preferències que es llisten a continuació (per ordre d'aparició a la Fig. 4, de dalt a baix):

- El títol de la finestra principal que apareixerà a la part superior del contenidor.
- La selecció del Controlador principal.
- Una opció que, si ha estat seleccionada, indiqui si els atributs dels Models es mostren a la Vista.
- Una opció que, si ha estat seleccionada, indiqui si s'han d'ocultar els mètodes dels Controladors que retornen atributs dels Models.
- El nombre mínim de controladors per dividir la Vista en diverses Vistes (per defecte el valor es 3).
- El nombre mínim d'arguments/valors de retorn per mostrar els mètodes en una finestra separada (per defecte el valor es 5).
- Descripció breu de l'aplicació.

Un cop l'usuari hagi establert les seves preferències, aquest pot decidir evaluar el model d'aprenentatge clicant el botó "Evaluate", o bé iniciar el procés de generació prement el botó "Generate".

### *7.3.2 Càrrega de fitxers*

Aquest mòdul conté les funcions corresponents a la càrrega dels fitxers de Python que contenen els Mòduls i Controladors implementats pel desenvolupador, a més del conjunt de dades d'entrenament. Es el pas previ a l'escaneig del codi font sencer, on es criden les funcions load_source_data() i load_train_data() per carregar el codi font i les dades d'entrenament, i es desen en dos variables auxiliars.

### *7.3.3 Escàner*

L'obtenció de les dades de prova que complementaran l'aprenentatge amb les dades d'entrenament es realitza a través de les funcions del mòdul scanner.py. Aquest mòdul fa servir el mòdul extern Abstract Syntax Tree, imprescindible per analitzar sintàcticament el codi font implementat pel desenvolupador, tant si es tracta de Models com de Controladors. La major part de les funcions implementades en aquest mòdul accedeixen a diccionaris inclosos a cadascun dels nodes extrets inicialment amb el mètode ast.parse(). Cada clau correspon a una propietat de l'objecte amb el qual correspon el node actual, i el valor es refereix a el significat de l'anterior "[8]". Per tant, cada node pot contenir diferents propietats gramaticals i el seu accés depèn del context amb el qual interessa l'obtenció d'informació útil per a el procés d'aprenentatge.

L'escàner s'executa des del programa principal mitjançant la funció scan(), passant com a argument el codi font carregat anteriorment. Aquesta crida la funció recursiva walk() i recorre l'arbre sintàctic fins trobar els mètodes de cada classe, dels quals se n'extreu la mostra corresponent al mètode amb la funció get_method_sample(). Si es el cas que la mostra del mètode té arguments i/o valors de retorn involucrats, llavors crida els mètodes get_argument_sample() i/o get_return_value_sample() per obtenir les mostres dels respectius arguments i/o els valors de retorn. Un cop finalitzat el recorregut, es retorna un Pandas DataFrame corresponent a totes les mostres obtingudes, tant si es troben en Models com a Controladors.

### *7.3.4 Model d'aprenentatge*

L'aprenentatge basat a partir de les dades d'entrenament i de prova s'efectua a partir de les funcions incloses al mòdul model.py. Altrament, s'inclou també la llibreria Scikit-Learn per a poder executar els mètodes d'aprenentatge pertinents.

Primerament s'executa la funció preprocess_and_split() i, després d'executar la funció anterior aplicada a les dades d'entrenament i de prova, s'executen les funcions fit() i classify(), que creen el classificador de descens de gradient estocàstic, entrena el model i retorna el resultat de la classificació.

### *7.3.5 Refinament*

Abans de generar la interfície gràfica es fa un tractament previ a les dades de prova. El mòdul refiner.py conté funcions encarregades de millorar les característiques de les mostres ajuntant-les, afegint de noves o modificant-les en determinats contexts d'ús. Per facilitar aquest procediment, les mostres del conjunt de prova es divideixen en tres grups (veure Taula 4).

TAULA 4
MOSTRES DEL CONJUNT DE PROVA

| Mostra | Descripció |
|---|---|
| data | Correspon a les mostres que s'han fet servir per a la classificació, juntament amb l'etiquetatge corresponent. |

| | |
|---|---|
| init_data | Correspon a les mostres dels constructors de totes les classes, tant de Models com de Controladors (sense incloure les mostres dels arguments i/o valors de retorn). |
| model_data | Correspon a les mostres dels Models escanejats. |

Algunes de les funcions que es fan servir en aquest mòdul es descriuen a la Taula 5.

TAULA 5
FUNCIONS DEL MÒDUL REFINER.PY

| Funció | Descripció |
|---|---|
| merge_arguments(data) | Uneix mostres d'arguments que tenen el mateix nom, tipus de dades, valors mínims i màxims, valor per defecte, valors possibles, *widget* i on el mètode al qual pertany es troba dins del mateix Controlador. |
| merge_return_values(data) | Uneix mostres de valors de retorn que tenen el mateix nom, tipus de dades, *widget* i on el mètode al qual pertany es troba dins del mateix Controlador. |
| set_languages(data) | Detecta a quin idioma correspon la nomenclatura del mètode. L'idioma és un codi basat en la norma ISO 639-1. |
| set_methods_as_menu_buttons(data, main_controller_name, view_threshold=3, window_threshold=5) | Estableix si els mètodes que es troben a la finestra principal es defineixen com a Menubutton sempre que el nombre d'arguments i/o valors de retorn superi el llindar establert. També defineix l'accés a la finestra de cada Controlador per a cada Vista a través del menú. |
| set_widget_label(data, init_data, model_data) | Crea una nova columna anomenada WidgetLabel que correspon a la nomenclatura utilitzada pel desenvolupador en la implementació, però sense caràcters especials. |
| set_widget_description(data, init_data, model_data) | Crea una nova columna anomenada WidgetDescription que correspon a la nomenclatura utilitzada pel desenvolupador en la implementació, però sense caràcters especials. |

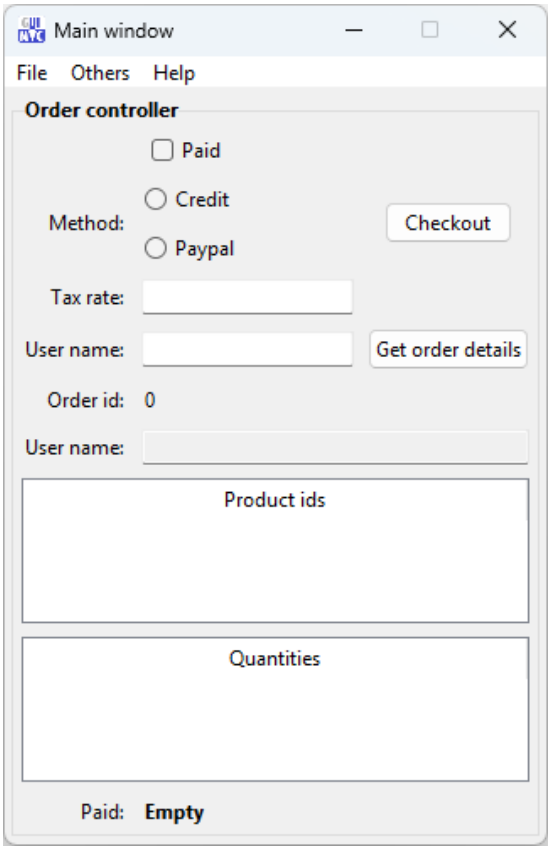


Fig. 5. Captura de pantalla de la finestra principal resultant de la generació de la vista (userProductOrder.py)

| | |
|---|---|
| set_window_methods(data, main_controller_name, window_threshold=5) | Estableix si es pot accedir a través d'una finestra nova als mètodes que es troben en una finestra corresponent a qualsevol vista diferent de la principal. |
| set_hide_show_model_attr(data, init_data, model_data, show_model_attr, hide_model_attr) | Retorna els valors de retorn de la mostra dels models que es mostraran a la interfície gràfica d'usuari. |
| set_attr_description(data) | Crea una nova columna anomenada AttrDescription que correspon a la nomenclatura dels atributs dels Models que es mostren a la interfície gràfica d'usuari, però sense caràcters especials. |

### *7.3.6 Generador*

Finalment, un cop acaba la fase de refinament, la informació de les mostres es troba en condicions de ser processada pel generador. Per tant, el mòdul generator.py genera el disseny de la interfície gràfica d'acord amb les dades especificades anteriorment, i les transforma en el disseny de la

part de Vista, on es faran els ajustaments necessaris per a la seva distribució, col·locació i funcionament.

## 8 Proves

COM a preludi de l'análisi de resultats, s'han realitzat unes proves com a demostració del funcionament de l'eina. Començem suposant que tenim un fitxer anomenat userProductOrder.py que conté 4 Models i 3 Controladors diferents. Si s'executa el mòdul main.py s'obrirà la finestra que es mostra a la Fig. 2, s'escaneja el codi font i generem la interfície gràfica seleccionant com a Controlador principal la classe OrderController i el nombre de Vistes per Controlador a 2. El resultat serà una finestra com la que es mostra a la Fig. 5.

Si es selecciona l'opció "Display Model attributes", es mostrarà, per a cada finestra relativa a la Vista de cada Controlador, els atributs dels Models que s'utilitzen (veure Fig. A3). Alternativament, si volem seleccionar l'opció "Hide methods which return Model attributes", s'amagaràn aquells mètodes que retornen valors corresponents a els atributs dels Models dels quals s'en fa ús (veure Fig. A4). En el cas de que es seleccionin ambdues opcions alhora, només es mostrarà a la interfície aquells atributs dels Models que no es fan servir per part de cap altre mètode que retorni valors provinents dels primers, fent que aquests últims s'amaguin (veure Fig. A5).

Tornant enrere a la configuració per defecte i només modificant el nombre mínim d'arguments/valors de retorn per generar una finestra auxiliar relativa a un mètode (en aquest cas a 3), ens apareixerà una finestra com la de la Fig. A6. En efecte, el menú ha estat subjecte de modificacions i ha inclòs dos botons repartits entre els submenús "View" i "Others". Si cliquem "View" ens apareix un botó anomenat "Get product summary". Al fer clic sobre aquest, una finestra auxiliar s'obrirà i inclourà els *widgets* que visualitzen els seus valors de retorn (veure Fig. A7). El mateix passa si es fa clic a "Get order details" però ubicat al submenú "Others" degut a que el seu mètode associat conté arguments i valors de retorn (Fig. A8).

El generador per defecte declara els objectes relatius a els Models, Controladors i Vistes a main.py, tot i que els atributs que es passen com a arguments també ho són. Això es degut a que es deixa lliure elecció al desenvolupador de definir les dades pertinents segons el seu criteri per a modificar la declaració dels Models (veure Fig. A9).

## 9 Resultats

FINALMENT, per a refutar el funcionament de l'eina de generació d'interfícies gràfiques, s'han obtingut una sèrie de resultats. Aquests es divideixen en les dues seccions on s'aprofundeixen sobre els mateixos.

### 9.1 Avaluació del model

L'avaluació del model són aquells resultats corresponents al rendiment final del model implementat, és a dir, les

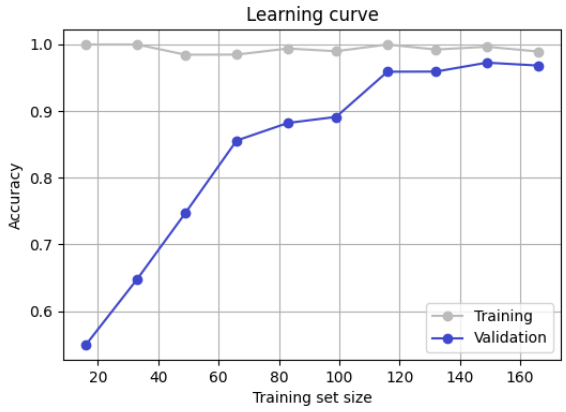


Fig. 6. Gràfica que mostra la corba d'aprenentatge (learning curve) del model entrenat (*k-fold cross-validation* amb k=4).

mesures que permeten valorar fins a quin punt el model compleix amb els objectius establerts. La obtenció d'aquests s'ha realitzat a través de la finestra del programa principal fent clic al botó "Evaluate". Com a primera instància es retornen la mitjana i la desviació estàndard de les mètriques *accuracy*, *precision*, *recall* i *F1-score*. A la Taula 6 es mosta la mitjana equivalent a els tres primers intents efectuats aplicant *k-fold cross-validation* (amb k=4).

TAULA 6
MITJANA DE LES MÈTRIQUES D'AVALUACIÓ DEL MODEL

| Mètrica | Intent 1 | Intent 2 | Intent 3 | $\bar{X}$ |
|---|---|---|---|---|
| Accuracy | 0.986 | 0.995 | 0.995 | 0.992 |
| Precision | 0.983 | 0.993 | 0.986 | 0.987 |
| Recall | 0.979 | 0.997 | 0.997 | 0.991 |
| F1-score | 0.974 | 0.995 | 0.989 | 0.986 |

La primera conclusió a extreure es que el model mostra un rendiment molt robust i equilibrat en la classificació de les dades d'entrenament. Gairebé totes les prediccions són correctes, mostrant un equilibri excel·lent entre les mètriques de *precision* i *recall* (aplicant *macro-average* a cada intent) "[15]". Tot i que s'ha fet servir la penalització L2 per evitar l'*overfitting*, entre altres mètriques, no es pot assegurar que el model es comporti sempre de manera adequada tal i com s'observa en els resultats anteriors.

Com a recurs addicional, l'eina permet generar la corba d'aprenentatge (*learning curve*), que mostra com canvia el rendiment d'un model a mesura que s'entrena amb més dades. La Fig. 6 mostra dues corbes: una pertany al conjunt d'entrenament i un altra per a un conjunt de validació (com el model es comporta per a dades desconegudes). Si ens fixem amb més detall es pot apreciar com la corba blava ascendeix progressivament a mesura que el conjunt de dades d'entrenament augmenta, mitigant el desbalanceig de classes inicial i afavorint un bon *accuracy*.

### 9.2 Generació dels fitxers

En el moment que es clica el botó "Generate", l'eina genera tres fitxers addicionals a banda de la implementació

corresponent a els Models i els Controladors. Aquests són:

- El fitxer main.py, on es declaren els Models, els Controladors i les Vistes pertinents i des d'on l'usuari pot iniciar el flux complet de la implementació.
- El fitxer view.py, on s'inclouen les definicions de les classes View i les que l'acompanyen.
- El fitxer utilities.py, que conté funcions auxiliars que serveixen com a suport a la vista (veure Taula 7).

TAULA 7
FUNCIONS DEL FITXER UTILITIES.PY

| Funció | Descripció |
|---|---|
| convert_str(value) | Converteix qualsevol valor de tipus str en un altre valor d'un altre tipus de dada (int, float, bool o complex). |
| get_treeview_items(treeview, _type) | Obté els elements inserits a un *widget* Treeview i els retorna en funció de la seva estructura en una llista, una tupla, un *set* o un diccionari. |
| set_treeview_items(treeview, return_value, language) | Insereix els elements que es troben en una llista, una tupla, un *set* o un diccionari a un *widget* Treeview. |
| get_boolean_str(value, language) | Tradueix el valor d'un booleà en format str en el llenguatge especificat. |
| get_boolean_fg(value) | Retorna el color del text en el qual es mostrarà el text traduït d'un booleà. |

El temps de generació es troba al voltant d'un segon, indicant que el procés de generació dels fitxers de la interfície gràfica ha estat força ràpid.

## 10 CONCLUSIONS

EN definitiva, l'assoliment d'aquest treball ha suposat un gran repte, sobretot pel que fa a la recollida de dades, donat que no es disposava d'una font externa que reunís les característiques necessàries per assegurar un bon model d'aprenentatge. Els objectius, en la seva majoria, s'acompleixen, tot i que el temps que triga a generar la interfície gràfica d'usuari depèn del model d'aprenentatge i de la implementació dels diferents mòduls, a més de que la precisió dels dissenys ve donada per l'abstracció de les dades cap a els elements gràfics dels quals es disposa. També l'ús extraordinari d'algunes llibreries externes de Python, com es el cas d'Abstract Syntax Tree o Tkinter, ha requerit l'aprenentatge sobre el seu ús de cara a poder completar el desenvolupament del treball de la manera mes satisfactòria possible. Tampoc es poden obviar desviacions puntuals sobre la planificació, com es el cas de la fase de desenvolupament, que ha requerit més dies per a la seva conclusió per tal d'oferir una versió més completa de l'eina. Tanmateix, la consecució del treball pot suposar una eina útil per facilitar el desenvolupament d'interficies gràfiques en determinats projectes, a més d'estalviar temps de producció de *software*.

## 11 LÍNIES OBERTES

COM a puntualització addicional a les conclusions anteriors, es considera que es deixen diverses vies obertes que podrien ser explorades en futurs treballs. Algunes d'aquestes s'expressen a continuació:

- La dinamització de la disposició dels elements gràfics o *widgets* sobre la interfície gràfica, oferint una experiència més personalitzada a els usuaris (segons l'àmbit d'ús).
- La retroalimentació de les dades d'aprenentatge en funció del comportament o interacció de l'usuari amb la interfície gràfica (partint d'una analogia semblant a els suggeriments de les pàgines web de botigues online), on es poden simplificar determinades accions per facilitar-ne l'ús o la seva comoditat.
- El desenvolupament íntegre d'una eina similar en altres llenguatges/plataformes que ofereixin major rendiment, portabilitat i escalabilitat, sobretot pel que fa a disseny.

Aquestes línies obertes no només reflecteixen algunes de les limitacions del present treball, sinó que també posen de manifest el potencial per continuar generant coneixement i debat en futurs treballs acadèmics.

## AGRAÏMENTS

VULL dedicar unes paraules d'agraïment, en primer lloc, a la meva tutora Yolanda Benítez Fernández per haver-me ajudat durant el transcurs del treball, haver estat sempre disponible a les reunions de cada setmana i haver oferit els seus coneixements, consells i anècdotes que segur enriquiràn la meva experiència professional. També vull fer una menció especial a la coordinació, tant en Jorge Bernal del Nozal, coordinador de la Menció de Computació, com a en Jordi Pons Aróztegui, coordinador del Grau d'Enginyeria Informàtica, per haver acceptat aquesta proposta original i assessorar-me davant els dubtes i procediments que han estat realitzats durant el treball, a més de tots aquells/es que, malgrat que no ha estat gaire exitosa, han dedicat part del seu temps en realitzar l'enquesta.

## BIBLIOGRAFIA


[1] Anònim, "Low-code development platform", *Wikipedia*, 2025, https://en.wikipedia.org/wiki/Low-code_development_platform

[2] Bradon Matthews, "Top 5 Types of Project Management Methodologies", *TechnologyAdvice*, 2025, https://technologyadvice.com/blog/project-management/project-management-methodologies/#:~:text=The%20top%20five%20most-used,Kanban%2C%20Scrum%2C%20and%20Lean

[3] Anònim, "Agile Software Development – Software Engineering", *GeeksForGeeks*, 2024, https://www.geeksforgeeks.org/software-engineering-agile-software-development/

[4] Vanessa Rosselló Villán, "13 herramientas agile para la gestión ágil de proyectos", *IEBS*, 2024, https://www.iebschool.com/blog/herramientas-gestion-agil-proyectos-agile-agile-scrum/

[5] Marc Talló Sendra, "Cicle de vida del projecte", *Gestió de projectes (UAB)*, 2023, pàg. 3-27, document PDF publicat al Campus Virtual de l'assignatura Gestió de Projectes.

[6] Martin Fitzpatrick, "Which Python GUI library should you use?", *PythonGUIs*, 2025, https://www.pythonguis.com/faq/which-python-gui-library/#tkinter

[7] Anònim, "pandes", *Pandas*, 2025, https://pandas.pydata.org/

[8] Shanshan, "Intro to Python ast Module", *Medium*, 2022, https://medium.com/@wshanshan/intro-to-python-ast-module-bbd22cd505f7

[9] Anònim, "scikit-learn", *Scikit-Learn*, data no disponible, https://scikit-learn.org/stable/

[10] Anònim, "langid 1.1.6", *Pypi*, 2016, https://pypi.org/project/langid/

[11] Anònim, "IPO model", *Wikipedia*, 2025, https://en.wikipedia.org/wiki/IPO_model

[12] Anònim, "Explicitly define datatype in a Python function", *GeeksForGeeks*, 2025, https://www.geeksforgeeks.org/explicitly-define-datatype-in-a-python-function/

[13] Scikit-Learn developers, "12. Choosing the right estimator", *Scikit-Learn*, 2007, https://scikit-learn.org/stable/machine_learning_map.html

[14] Anònim, "One Hot Encoding in Machine Learning", *GeeksForGeeks*, 2025, https://www.geeksforgeeks.org/ml-one-hot-encoding/

[15] Anònim, "Accuracy, precision, and recall in multi-class classification", *Evidently AI*, 2025, https://www.evidentlyai.com/classification-metrics/multi-class-metrics

# APÈNDIX

| Id | Modo de tarea | Nombre de tarea | Duración | Comienzo | Fin | Predecesoras |
|---|---|---|---|---|---|---|
| 1 | | **Generació del disseny de la interfície gràfica d'un programa basat en l'arquitectura MVC mitjançant** | **142 días** | **lun 10/02/25** | **mar 01/07/25** | |
| 2 | | **Fase inicial** | **28 días** | **lun 10/02/25** | **dom 09/03/25** | |
| 3 | | **Iniciació** | **7 días** | **lun 10/02/25** | **dom 16/02/25** | |
| 4 | | Definició d'objectius | 3 días | lun 10/02/25 | mié 12/02/25 | |
| 5 | | Realització de l'estudi de viabilitat | 4 días | jue 13/02/25 | dom 16/02/25 | 4 |
| 6 | | **Planificació** | **21 días** | **lun 17/02/25** | **dom 09/03/25** | **3** |
| 7 | | Definició de WBS (Work Breakdown Structure) | 3 días | lun 17/02/25 | mié 19/02/25 | |
| 8 | | Definició dels recursos | 3 días | jue 20/02/25 | sáb 22/02/25 | 7 |
| 9 | | Definició del calendari | 3 días | dom 23/02/25 | mar 25/02/25 | 8 |
| 10 | | Avaluació de riscos | 6 días | mié 26/02/25 | lun 03/03/25 | 9 |
| 11 | | Conclusions del pla de projecte | 6 días | mar 04/03/25 | dom 09/03/25 | 10 |
| 12 | | Lliurament informe inicial | 1 día | dom 09/03/25 | dom 09/03/25 | |
| 13 | | **Fase intermèdia** | **77 días** | **lun 10/03/25** | **dom 25/05/25** | **2** |
| 14 | | **Anàlisi i disseny** | **14 días** | **lun 10/03/25** | **dom 23/03/25** | |
| 15 | | Fase d'anàlisi | 7 días | lun 10/03/25 | dom 16/03/25 | |
| 16 | | Fase de disseny | 7 días | lun 17/03/25 | dom 23/03/25 | 15 |
| 17 | | **Desenvolupament** | **63 días** | **lun 24/03/25** | **dom 25/05/25** | **14** |
| 18 | | Fase de construcció | 49 días | lun 24/03/25 | dom 11/05/25 | |
| 19 | | Lliurament informe de progrés (I) | 1 día | dom 13/04/25 | dom 13/04/25 | |
| 20 | | Control de qualitat | 7 días | lun 12/05/25 | dom 18/05/25 | 18 |
| 21 | | Fase de documentació | 7 días | lun 19/05/25 | dom 25/05/25 | 20 |
| 22 | | Lliurament informe de progrés (II) | 1 día | dom 25/05/25 | dom 25/05/25 | |
| 23 | | **Fase final** | **37 días** | **lun 26/05/25** | **mar 01/07/25** | **13** |
| 24 | | Resultats | 21 días | lun 26/05/25 | dom 15/06/25 | |
| 25 | | Lliurament informe final | 1 día | dom 15/06/25 | dom 15/06/25 | |
| 26 | | Finalització | 16 días | lun 16/06/25 | mar 01/07/25 | 24 |
| 27 | | Lliurament presentació | 1 día | dom 29/06/25 | dom 29/06/25 | |

Fig. A1. Captura de pantalla de l'ordre de les fases, activitats i tasques del treball (Microsoft Project).

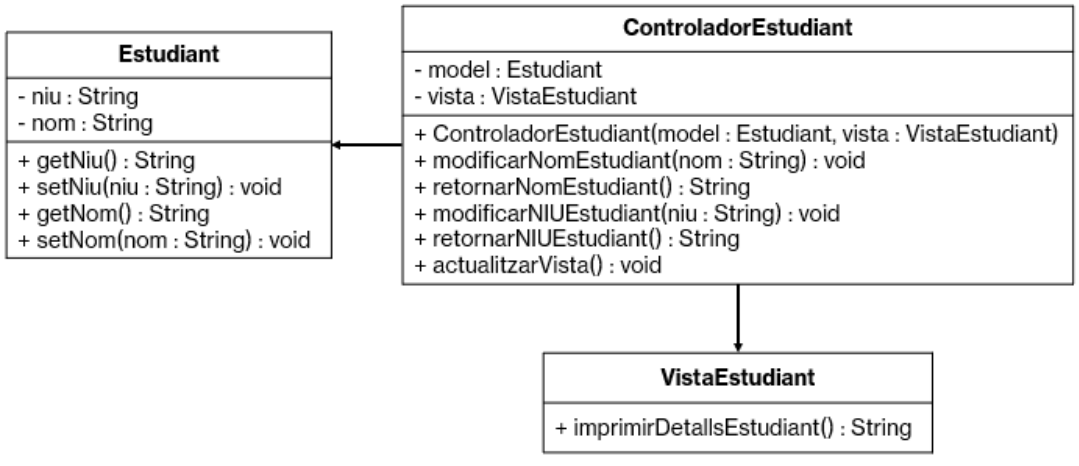


Fig. A2. Exemple d'arquitectura Model-Vista-Controlador (diagrama de classes).

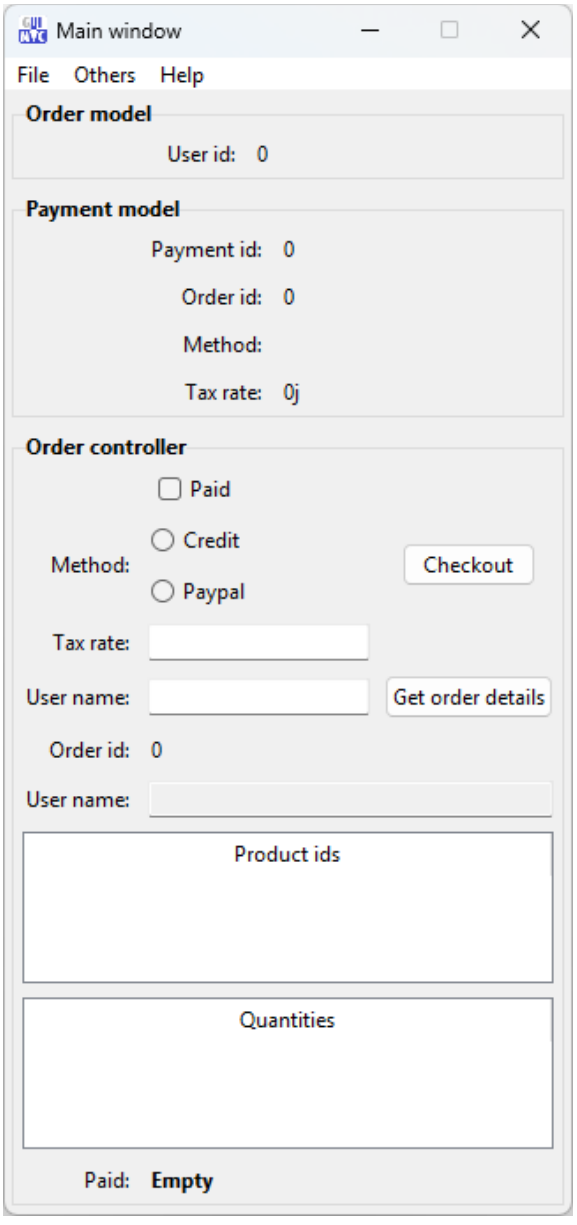


Fig. A3. Captura de pantalla de la finestra principal mostrant el contingut dels atributs (userProductOrder.py).

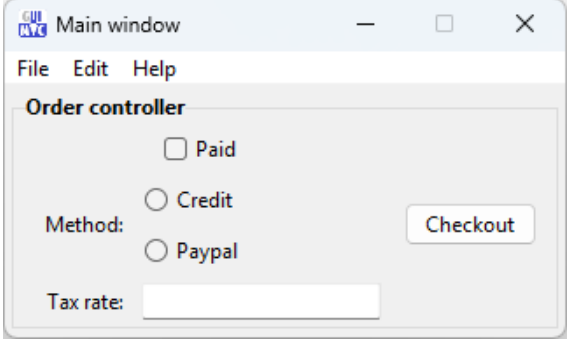


Fig. A4. Captura de pantalla de la finestra principal amagant els mètodes que retornen valors corresponents a els atributs dels Models (userProductOrder.py)

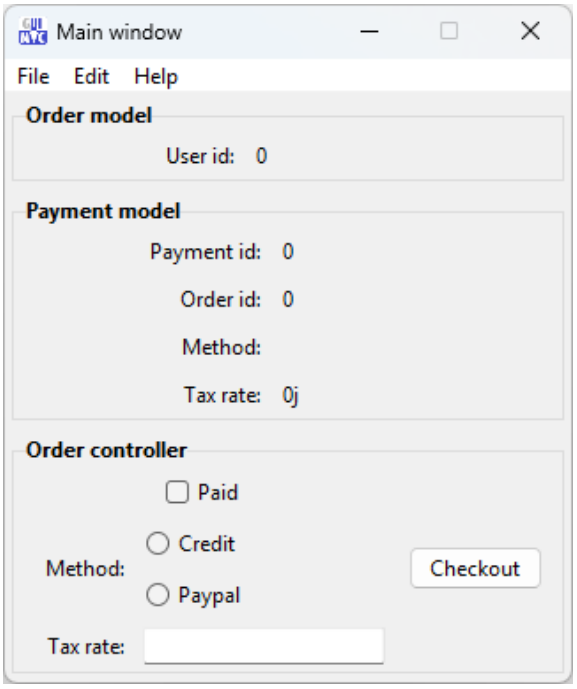


Fig. A5. Captura de pantalla de la finestra principal mostrant aquells atributs dels Models que no es fan servir per part de cap altre mètode que retorni valors provinents dels primers (userProductOrder.py).

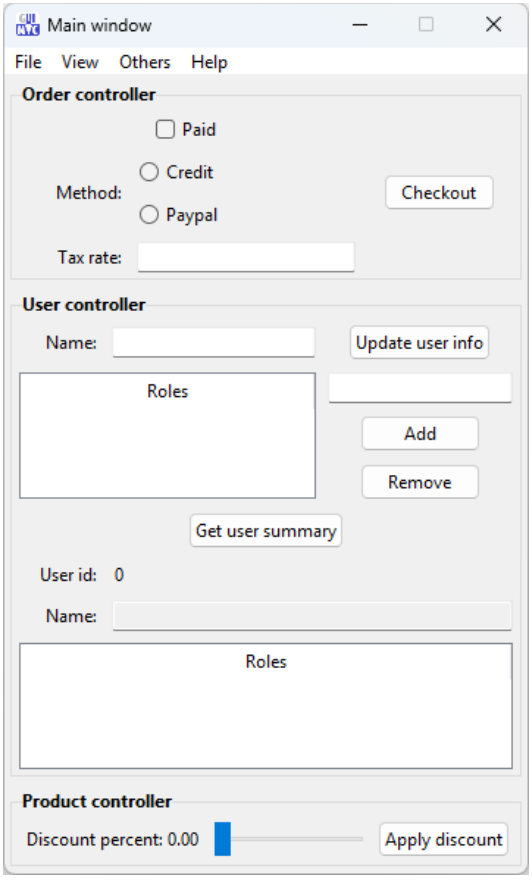


Fig. A6. Captura de pantalla de la finestra principal mostrant tots els Controladors (userProductOrder.py).

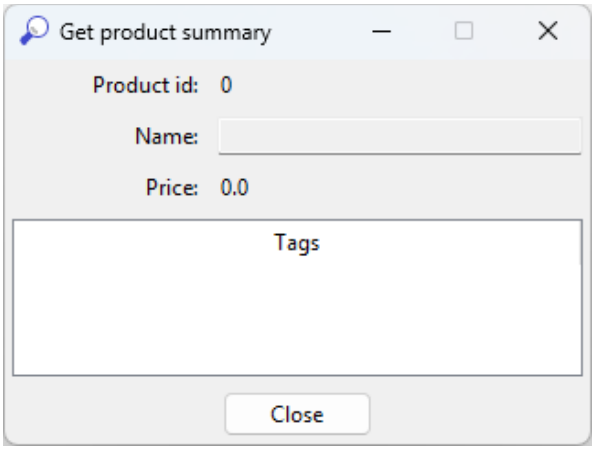


Fig. A7. Captura de pantalla d'una finestra que inclou els *widgets* que visualitzen els valors de retorn del mètode get_product_summary() (userProductOrder.py).

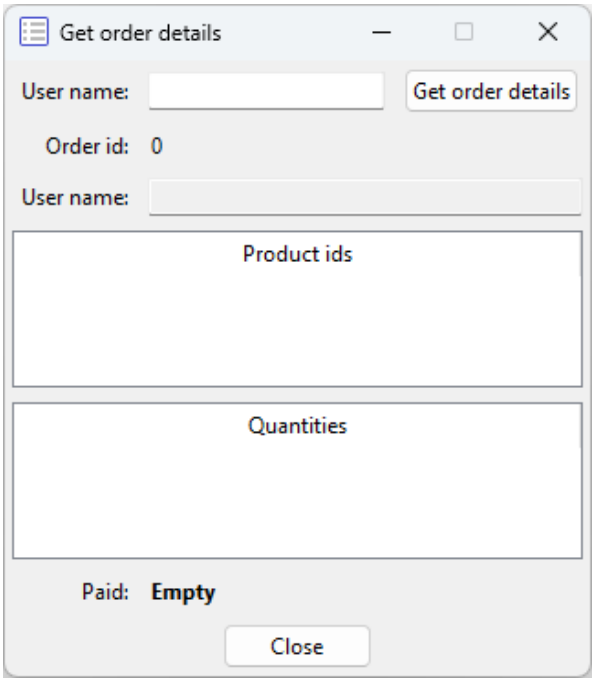


Fig. A8. Captura de pantalla d'una finestra que inclou els *widgets* que visualitzen els arguments i els valors de retorn del mètode get_order_details() (userProductOrder.py).

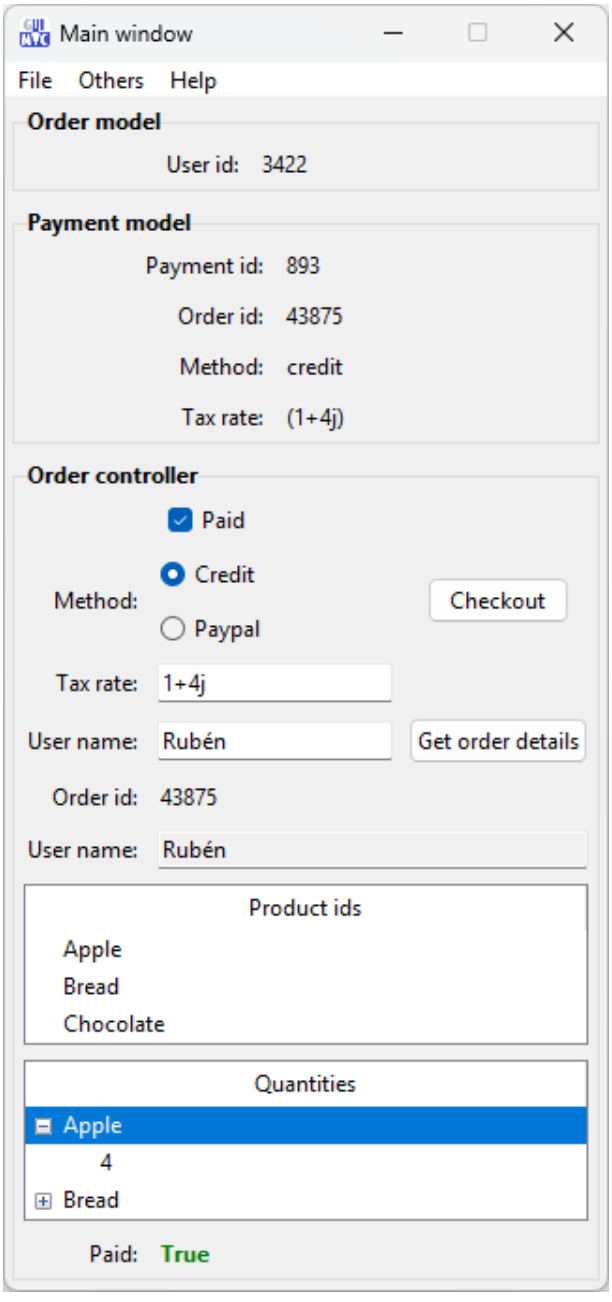


Fig. A9. Captura de pantalla de la finestra principal amb els valors dels atributs modificats a la declaració dels Models a main.py (userProductOrder.py).