# **الملخص**

استجابةً للتجربة غير الشخصية التي تقدمها منصات الحجز التقليدية في قطاع الضيافة شديد التنافسية، يقدّم هذا المشروع منصة مبتكرة ومتكاملة مصممة لتوحيد حجوزات الفنادق والمطاعم ضمن نظام واحد. يتمحور جوهر النظام حول محرك توصيات متطور يعتمد على التعلم العميق، والذي يزوّد المستخدمين باقتراحات استباقية وشخصية عبر تحليل بيانات متنوعة مثل التفضيلات وسجل الحجوزات والمراجعات. يستخدم هذا المحرك نموذجًا هجينًا يجمع بين التصفية التعاونية والتصفية القائمة على المحتوى، وقد تم بناؤه باستخدام Python و PyTorch. تتكون البنية المعمارية للمنصة من تطبيق خلفي (Backend) تم تطويره باستخدام ASP.NET Core Web API، متبعاً مبادئ البنية النظيفة (Clean Architecture) ومستخدماً أنماط التصميم CQRS، والوسيط (Mediator)، والمستودع (Repository). أما الواجهة الأمامية (Frontend) فتتألف من تطبيق للمستخدمين مبني باستخدام Flutter و مُهيكل وفق نمط MVVM، وتطبيق لمديري النظام مبني باستخدام React. تتمثل المساهمة الأساسية لهذا المشروع في تقديم حل متكامل يعالج فجوة كبيرة في السوق، حيث يوفر تجربة مستخدم سلسة تلغي الحاجة إلى استخدام تطبيقات متعددة ومنفصلة. كما أن استخدام مجموعة تقنيات حديثة وفعالة يوفر أساسًا مرنًا وقابلًا للتطوير لمنصة سوق مفتوح قادرة على المنافسة.

# **Abstract**

In response to the impersonal experience offered by traditional booking platforms in the highly competitive hospitality sector, this project introduces an innovative, integrated platform designed to unify hotel and restaurant reservations within a single system. The system's core is a sophisticated recommendation engine based on deep learning, which provides users with proactive and personalized suggestions by analyzing diverse data such as preferences, booking history, and reviews. This engine employs a hybrid model that combines collaborative filtering and content-based filtering, built using Python and PyTorch.The platform's architecture consists of a backend developed with ASP.NET Core Web API following Clean Architecture principles and utilizing CQRS, Mediator, and Repository design patterns. The frontend comprises a Flutter-based application for users, structured with the MVVM pattern, and a React-based application for system administrators. This project's primary contribution is an integrated solution that addresses a significant gap in the market, offering a seamless user experience that eliminates the need for multiple separate applications. The use of a modern and efficient technology stack further provides a flexible and scalable foundation for a competitive open-market platform.

Table of Contents

[**الملخص** II](#_Toc205041718)

[**Abstract** II](#_Toc205041719)

[الفصل الأول 1](#_Toc205041720)

[1.1 مقدّمة: 1](#_Toc205041721)

[2.1. الهدف من المشروع: 1](#_Toc205041722)

[3.1. المتطلبات الوظيفية: 1](#_Toc205041723)

[**4.1. المتطلبات غير الوظيفية:** 2](#_Toc205041724)

[الفصل الثاني 3](#_Toc205041725)

[**1.2. تمهيد عن تعلم الآلة:** 3](#_Toc205041726)

[**2.2. أنواع تعلّم الآلة:** 3](#_Toc205041727)

[**1.2.2. التعلم الخاضع للإشراف Supervised Learning:** 3](#_Toc205041728)

[**2.2.2. التّعلم غير الخاضع للإشراف Unsupervised Learning:** 3](#_Toc205041729)

[**3.2.2. التّعلم المعزّز Reinforcement Learning:** 4](#_Toc205041730)

[**3.2. التّعلم العميق:** 4](#_Toc205041731)

[**1.3.2. العصبون:** 4](#_Toc205041732)

[**2.3.2. الشّبكة العصبيّة:** 7](#_Toc205041733)

[**3.3.2. مقاييس التقييم Evaluation Metrics :** 11](#_Toc205041734)

[**4.3.2. عمليّة التّعلم:** 12](#_Toc205041735)

[**4.2. نظام التّوصية Recommendation Systems:** 14](#_Toc205041736)

[**1.4.2. أنوع التّوصيات:** 15](#_Toc205041737)

[**2.4.2. مشكلة البداية الباردة Cold Start Problem :** 16](#_Toc205041738)

[**5.2. نموذج اللّغة الكبير (LLM) :** 16](#_Toc205041739)

[**1.5.2. ما هو نموذج اللّغة الكبير ؟** 16](#_Toc205041740)

[**2.5.2. ماذا يمكن أن يفعل نموذج اللّغة الكبير:** 16](#_Toc205041741)

[**6.2. البنية المعماريّة Architecture:** 17](#_Toc205041742)

[**7.2. البنية المعماريّة النّظيفة Clean Architecture:** 17](#_Toc205041743)

[**1.7.2. مفهوم المعماريّة النّظيفة:** 17](#_Toc205041744)

[**2.7.2. قاعدة التبعية:** 18](#_Toc205041745)

[**8.2. أنماط التّصميم Design Patterns :** 19](#_Toc205041746)

[**9.2. فصل الاستعلامات و الأوامر Command Query Segregation (CQS) :** 20](#_Toc205041747)

[**10.2. فصل مسؤولية الاستعلام و الأوامر Command Query Responsibility Segregation:** 20](#_Toc205041748)

[الفصل الثالث 22](#_Toc205041749)

[**1.3. الدراسة المرجعية لنظام التوصية:** 22](#_Toc205041750)

[**1.1.3. لمحة عن مجموعة البيانات المستخدمة:** 22](#_Toc205041751)

[**2.1.3. نماذج التصفية التعاونية:** 24](#_Toc205041752)

[**3.1.3. الاستفادة من نص المراجعة مع نماذج اللغة الكبيرة:** 25](#_Toc205041753)

[**4.1.3. أنظمة التوصية الهجينة:** 25](#_Toc205041754)

[**2.3. الدراسة المرجعية لنظام الحجوزات:** 26](#_Toc205041755)

[**1.2.3. وضع السّوق والمنافسة:** 26](#_Toc205041756)

[**2.2.3. نظرة عامّة على السّوق:** 26](#_Toc205041757)

[**3.2.3. التّحليل التّنافسيّ ونقاط القوّة في المشروع:** 26](#_Toc205041758)

[الفصل الرابع 28](#_Toc205041759)

[**1.4. مخططات حالات الاستخدام:** 28](#_Toc205041760)

[**1.1.4. حالات استخدام الضيف:** 28](#_Toc205041761)

[**2.1.4. حالات استخدام الزبون:** 29](#_Toc205041762)

[**3.1.4. حالات استخدام مدير النظام:** 30](#_Toc205041763)

[**2.4. السرد النصي لحالات الاستخدام:** 31](#_Toc205041764)

[**1.2.4. حالات استخدام الضيف:** 31](#_Toc205041765)

[**2.2.4. حالات استخدام المستخدم:** 33](#_Toc205041766)

[**3.2.4. حالات استخدام مدير النّظام:** 41](#_Toc205041767)

[**2.5. مخططات تسلسل النظام:** 46](#_Toc205041768)

[**1.2.5. الحجز لدى مطعم:** 46](#_Toc205041769)

[**2.2.5. حجز غرفة في فندق:** 47](#_Toc205041770)

[**3.2.5. استعراض الأغراض الموصى بها:** 48](#_Toc205041771)

[**3.5. مخطط كائنات العلاقات ERD في النظام:** 48](#_Toc205041772)

[**1.3.5. مخطط المواقع:** 49](#_Toc205041773)

[**2.3.5. مخطط أنواع العمولة:** 49](#_Toc205041774)

[**3.3.5. مخطط الفنادق:** 50](#_Toc205041775)

[**4.3.5. مخطط المطاعم:** 51](#_Toc205041776)

[**5.3.5. مخطط الأحداث:** 52](#_Toc205041777)

[**6.3.5. مخطط المستخدمين و ارتباطاته مع باقي المخططات:** 53](#_Toc205041778)

[الفصل الخامس 54](#_Toc205041779)

[**1.6. البنية المقترحة:** 54](#_Toc205041780)

[**2.6. تنجيز التطبيق الخلفي Backend:** 55](#_Toc205041781)

[**1.2.6. إطار العمل:** 55](#_Toc205041782)

[**2.2.6. هيكلية الطبقات وتدفق العمليات:** 56](#_Toc205041783)

[**3.6. محرك التوصية Recommendation Engine:** 57](#_Toc205041784)

[**1.3.6. بنية النموذج** 57](#_Toc205041785)

[**2.3.6. التدريب والتقييم:** 57](#_Toc205041786)

[**3.3.6. واجهة برمجة التطبيقات API:** 58](#_Toc205041787)

[**4.6. تنجيز الواجهات الأمامية Frontend:** 58](#_Toc205041788)

[**1.4.6. منهجية العمل:** 58](#_Toc205041789)

[المراجع: 59](#_Toc205041790)

# الفصل الأول

**التّعريف بالمشروع**

*يتضمن هذا الفصل التعريف بالمشروع ومتطلباته.*

## 1.1 مقدّمة:

في ظل المنافسة الشديدة التي يشهدها قطاع الضيافة، أصبحت تجربة المستخدم الشخصية عاملًا حاسمًا. إلا أن منصات الحجز التقليدية للفنادق والمطاعم لا تزال تقدم تجربة عامة وغير مخصصة، مما يحد من فعاليتها. لمواجهة هذا التحدي، قمنا بتطوير منصة حجز مبتكرة ومتكاملة، مدعومة بمحرك توصيات متطور يعتمد على التعلم العميق. يقوم هذا النظام بتحليل بيانات المستخدمين المتنوعة، مثل تفضيلاتهم، سجل حجوزاتهم، وآرائهم في التقييمات، وذلك لتقديم اقتراحات استباقية تتناسب مع أذواقهم واحتياجاتهم بدقة.

## 2.1. الهدف من المشروع:

الهدف الرئيسي من هذا المشروع هو تسهيل عملية الحجز في المطاعم و الفنادق بحيث تصبح متوفرة كبنية متكاملة في نظام واحد إضافةً إلى التّحسين من رضى المستخدم عن طريق عرض توفير توصيات مخصصة له.

## 3.1. المتطلبات الوظيفية:

يجب على النظام أن :

* يسمح للمستخدم ب:
  + تسجيل الدخول باستخدام حسابه و كلمة المرور الخاصة به.
  + إضافة حساب جديد باستخدام حساب غير مسجل سابقاً و كلمة مرور.
  + استعراض المطاعم و الفنادق المتوفرة لدى النظام.
  + الاطلاع على المعلومات الخاصة بمطعم أو فندق معين.
  + إضافة مراجعة لمطعم أو فندق معين.
  + الحجز في مطعم أو فندق مُسجَّل سابقاً.
  + تحديد التاريخ الذي يريد الحجز فيه و عدد الأشخاص الذين سوف يكونون في الحجز معه.
  + تعديل بياناته الشّخصيّة.
* يسمح لمدير النظام:
  + بإضافة مطعم أو فندق جديد مع البيانات الخاصة به.
  + تعديل بيانات مطعم أو فندق موجود سابقاً.
  + حذف مطعم أو فندق من النّظام.
  + إضافة مدير نظام آخر.
* يقوم بتزويد المستخدم بتوصية للمطاعم و الفنادق التي قد تعجبه.

## **4.1. المتطلبات غير الوظيفية:**

* يجب أن تكون استجابة النظام سريعة لطلبات المستخدم أي أن لا تستغرق أكثر من ثلاث ثوان كحد أقصى.
* يجب على النظام أن يوفر واجهة تخاطب سهلة الاستخدام للمستخدم بحيث لا يحتاج المستخدم لأكثر من ساعتين كحد أقصى للتآلف مع التطبيق.
* يجب على النظام أن يوفر لمدير النظام واجهة تخاطب سهلة الاستخدام.
* يجب أن يتم بناء تصميم الواجهة الخلفية Backend للنظام بتقنية ASP.NET Core Web API .
* يجب أن يتم تصميم الواجهة الأمامية Frontend للمستخدم بتقنية Flutter .
* يجب تصميم الواجهة الأمامية لمدير النّظام بتقنية React .
* يجب على النظام أن يكون مرن و قابل للتوسع.

# الفصل الثاني

**الدّراسة النّظرية**

*يقدم هذا الفصل تمهيد عن تعلم الآلة، إضافةً إلى تمهيد عن نظام المقترحات*

## **1.2. تمهيد عن تعلم الآلة:**

التّعلم الآلي هو نوع من الذّكاء الاصطناعي (AI) الذي يسمح للنّظم الحاسوبيّة بالتّعلم من البيانات وتحسين أدائها في مهمّة ما دون الحاجة إلى برمجتها بشكل صريح. بدلاً من اتّباع مجموعة من التّعليمات المكتوبة مسبقاً، تستخدم هذه النّظم خوارزميات لتحديد الأنماط داخل البيانات واتخاذ التّنبؤات أو القرارات. في جوهره، يتعلّق التّعلم الآلي بتمكين الآلة من التّعلم من التّجربة. تأتي هذه "التّجربة" في شكل كمّيات هائلة من البيانات. كلما زادت البيانات التي يعالجها النّظام، كلّما صقل فهمه وازدادت دقّة مخرجاته. هذه العملية مشابهة لكيفية تعلّم البشر، من خلال المراقبة، وتحديد الأنماط، وتعديل فهمهم بناءً على المعلومات الجديدة.

## **2.2. أنواع تعلّم الآلة:**

### **1.2.2. التعلم الخاضع للإشراف Supervised Learning:**

في هذا النّهج، يتم تدريب الآلة على مجموعة بيانات "مُعَلَّمة"، مما يعني أن النّاتج الصّحيح معروف مسبقاً. هدف النّموذج هو تعلّم دالّة التّحويل التي تحوّل بيانات الإدخال إلى النّاتج الصّحيح. مثال شائع هو عامل تصفية (فلتر) البريد الإلكتروني العشوائي، الذي يُدرَّب على رسائل بريد إلكتروني صُنِّفت على أنّها "بريد عشوائي" أو "ليست بريدًا عشوائيا"[1].

### **2.2.2. التّعلم غير الخاضع للإشراف Unsupervised Learning:**

هنا يتم إعطاء الآلة بيانات غير مصنّفة وعليها أن تجد أنماطاً وهياكل بمفردها. و خير مثالٍ على ذلك هو تجميع المقالات الإخباريّة. يمكن لخوارزميّة غير خاضعة للإشراف تحليل الآلاف من المقالات وتصنيفها تلقائيّاً بناءً على الكلمات والمواضيع المتشابهة. وتقوم بذلك دون أي مُعَلَّمات محدّدة مسبقاً، مما يؤدي بشكل فعّال إلى إنشاء فئاتها الخاصّة مثل "التّكنولوجيا"، "الرّياضة"، أو "الصّحة" من النّص نفسه. يُستخدم التّعلم غير الخاضع للإشراف عندما يكون لدينا الكثير من البيانات ونريد اكتشاف الأنماط أو الهياكل المخفيّة فيها، دون أن يكون لدينا أي نتائج مسبقة أي إنّها في الأساس شكل من أشكال استكشاف البيانات الذي يسمح للآلة باكتشاف الضّمنيات بمفردها [1].

### **3.2.2. التّعلم المعزّز Reinforcement Learning:**

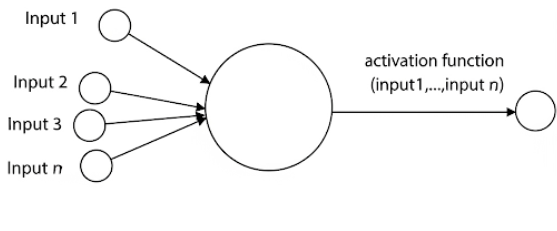
هذا النّوع من التّعلم ينطوي على وكيل يتعلّم من خلال التّفاعل مع بيئته. يتلقى الوكيل مكافآت مقابل أداء إجراءات تؤدي إلى نتيجة مرغوبة وعقوبات على الأفعال غير الصّحيحة. مع مرور الوقت، يتعلم الوكيل أفضل استراتيجية لتحقيق أقصى قدر من المكافآت [1]. هذه هي الطّريقة الأساسية المستخدمة لتدريب السّيارات ذاتيّة القيادة وفي علم الرّوبوتات.

## **3.2. التّعلم العميق:**

التّعلم العميق هو نوع متقدّم من التّعلم الآلي مستوحى من هيكل وعمل الدّماغ البشري. يستخدم الشّبكات العصبية المعقّدة متعدّدة الطّبقات للتّعلم من كمّيات هائلة من البيانات. يشير "العميق" في التّعلم العميق إلى العديد من طبقات هذه الشّبكات العصبيّة. تتعلّم كل طبقة التّعرف على ميّزات أكثر تعقيدًا بشكل متزايد في البيانات. على سبيل المثال، عند تحليل صورة، قد تكشف الطّبقات الأولية عن حواف وألوان بسيطة، وقد تتعرف الطبقات التالية على أشكال وقوام، بينما يمكن أن تتعرف الطبقات النهائية على كائنات كاملة مثل وجه أو سيّارة [2].

### **1.3.2. العصبون:**

العصبون والمعروف أيضاً باسم الخليّة العصبيّة الاصطناعيّة أو العقدة، هو الوحدة الأساسيّة لمعالجة البيانات في الشّبكة العصبيّة. إنه نمذجة رياضيّة مستوحاة من الخلايا العصبيّة البيولوجيّة، مصمّمة لاستقبال ومعالجة وتمرير المعلومات. وعلى الرغم من أنّه أبسط بكثير من نظيره البيولوجيّ، إلا أنّه المكوّن الأساسيّ الذي يسمح لنماذج التّعلم العميق بالتّعلم من البيانات [2].



الشكل : شكل يوضح العصبون مع مدخلاته و خرجه.

#### **1.1.3.2. كيف يعمل العصبون:**

يمكن النّظر إلى العصبون على أنّه مُتّخذ قرارات صغير. خيث أنّه يستقبل المعلومات، ويزن أهمّيتها، ثمّ يقرّر ما إذا كان سينقل إشارة إلى خلايا عصبيّة أخرى. تحتوي هذه العملية على أربعة خطوات رئيسية [2]:

* استقبال المدخلات: يقوم العصبون باستقبال مُدخل واحد أو أكثر. يمكن أن تكون هذه المدخلات الميزات الخام للبيانات (مثل وحدات البكسل في صورة) أو المخرجات من عصبونات أخرى في طبقة سابقة.
* معالجة المدخلات (المجموع الموزَّن): يتم ضرب كل مدخل بوزن. هذا الوزن يمثل أهمية هذا المدخل المحدد بحيث إذا كان الوزن أعلى يكون المدخل له تأثير أكبر على ناتج الخليّة العصبيّة. تقوم الخليّة العصبيّة بعد ذلك بجمع كل هذه المدخلات الموزّنة. كما تضيف انحيازًا، وهو مدخل إضافي واحد يسمح للخلية العصبيّة بتعديل ناتجها بشكلٍ مستقلٍّ عن مدخلاتها، مما يزيد من مرونتها.
* تطبيق تابع التّنشيط: يتم تمرير ناتج المجموع الموزّن عبر دالّة تنشيط. هذه الدّالة هي خطوة غير خطّية حاسمة تحدّد النّاتج النهائي للخليّة العصبيّة. أي إنّها تحدّد أساساً ما إذا كان يجب تنشيط الخليّة العصبيّة وما يجب أن تكون عليه قوّة إشارتها. فيما يلي توابع التّنشيط الشّائعة:
  + Sigmoid: يأخذ أي قيمة حقيقيّة كمدخل ويضغطها إلى قيمة بين 0 و 1. يُستخدم غالباً في طبقة الخرج لمشاكل التّصنيف الثّنائي، حيث الهدف هو التّنبؤ بأحد النّتائج الممكنة من اثنتين (مثل "نعم/لا" أو "قط/كلب"). ومع ذلك، فهو أقلُّ شيوعًا في الطّبقات المخفيّة للشّبكات الحديثة لأنّه يمكن أن يؤّدي إلى مشكلة "تلاشي التّدرج" (Vanishing Gradient)، والتي تُبطئ عملية التّعلم.فيما يلي معادلته [2]:

*حيث:*

* *: قيمة الدّخل.*
* : *رقم أويلر.*
  + ReLU: هو واحد من أكثر توابع التّنشيط الشّعبيّة المستخدمة في الطّبقات المخفيّة للشّبكات العصبيّة العميقة. وصيغته بسيطة بحيث يتم إخراج المدخلات مباشرةً إذا كانت إيجابيّة، ويتم إخراج صفر إذا كانت سلبيّة. تجعل هذه البساطة حساباته فعّالة للغاية وتساعد على منع مشكلة تلاشي التّدرج. ومن العيوب المحتملة مشكلة "ReLU dying "، حيث يمكن أن تصبح الخلايا العصبيّة غير نشطة ويتوقف التّعلم إذا كانت مدخلاتها سلبيّة باستمرار. فيما يلي معادلته [2]:

حيث:

* *: قيمة الدّخل.*
  + Leaky ReLU: يعتبر تنويعًا لـ ReLU مصممًا لحل مشكلة " dying ReLU". بدلاً من إخراج صفر للمدخلات السلبيّة، فهو يخرج قيمة إيجابيّة صغيرة جداً (مثل 0.01 من المدخل). هذا يضمن أنّ الخليّة العصبيّة لديها دائمًا بعض التدرج، ممّا يسمح لها بمواصلة التّعلم حتّى مع المدخلات السلبيّة. إنه بديل جيد لـ ReLU عندما يتم مواجه خلايا عصبيّة ميّتة في الشّبكه. فيما يلي معادلته [2]:

حيث:

* *: قيمة الدّخل.*
* : ثابت موجب صغير يحدد ميل المدخلات السلبية.
  + Softmax: يُستخدم عادةً في الطّبقة النّهائيّة من الشّبكة العصبيّة لمشاكل التّصنيف متعدّد الفئات. يأخذ هذه التّابع متّجهاً من الأعداد الحقيقيّة ويحوّله إلى توزيع احتمالي، حيث تتراوح كل قيمة بين 0 و 1، وجميع القيم يكون مجموعها 1. يسمح ذلك للشّبكة بإخراج احتمال انتماء المدخلات إلى كل من الفئات المحتملة. فيما يلي معادلته في حالة شعاع حيث هو عدد الفئات [2]:

*حيث:*

* *: احتمالية الفئة* i *.*
* *يحقّق التّابع السّابق العلاقة:*
  + Tanh (Hyperbolic Tangent): دالة مشابهة لدالة Sigmoid لكنّها تضغط القيم إلى نطاق يتراوح بين -1 و 1. ناتجها مركزي حول الصفر، مما قد يسهل على النّموذج التّعلم. مثل Sigmoid ، فهي أيضًا معرضة لمشكلة التّدرج المتلاشي، ولكنها غالبًا ما تُفضل على Sigmoid في الطّبقات المخفيّة في أنواع معينة من الشّبكات، مثل الشّبكات العصبيّة المتكرّرة (RNNs). فيما يلي معادلته [2]:

حيث:

* *: قيمة الدّخل.*

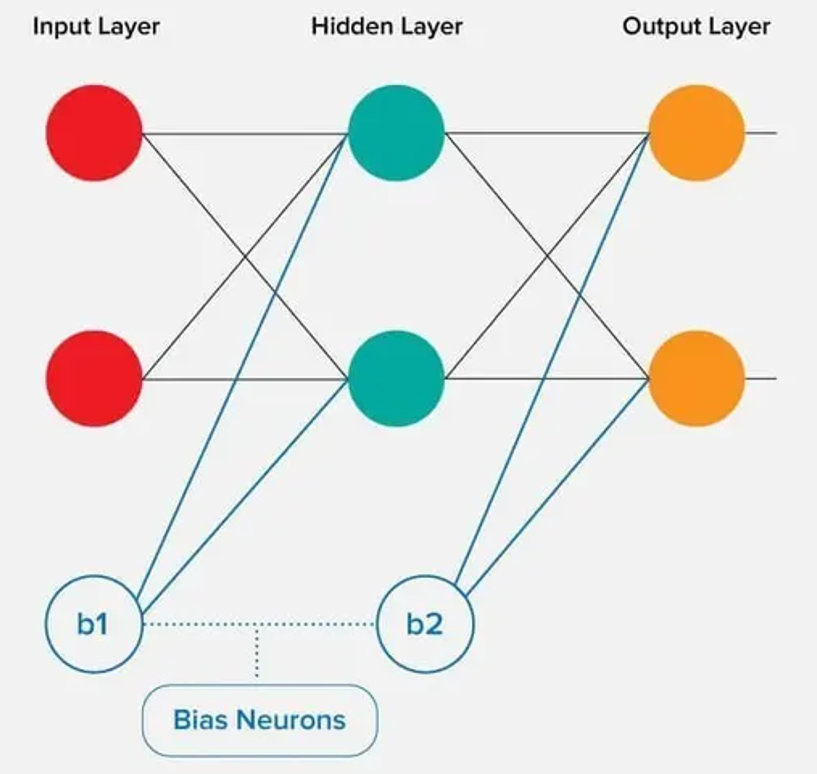
### **2.3.2. الشّبكة العصبيّة:**

الشّبكة العصبيّة هي نموذج حسابيّ مستوحى من الدّماغ البشريّ مصمّم للتّعرف على الأنماط في البيانات. تُعَد الشّبكة العصبيّة الإطار الأساسيّ للتّعلم العميق وتستخدم في مهام مثل التّعرف على الصّور، والتّرجمة اللّغوية، ولعب الألعاب. يُمكن تشبيهها بفريق من صانعي القرار المتخصّصين الذين يعملون معًا لحل مشكلة مُعقّدة [2].

#### **1.2.3.2. بنية الشّبكة العصبيّة:**

يتم تنظيم الشّبكة العصبيّة في طبقات من العقد المترابطة (العصبونات). لكل طبقة وظيفة محددة [2]. فيما يلي نذكر هذه الطبقات:

* طبقة الإدخال: هي الطّبقة الأولى التي تتلقّى البيانات الأوّلية. على سبيل المثال، إذا كنّا تقوم بتحليل صورة، فإنّ كل خليّة عصبيّة في طبقة الدّخل ستمثّل بكسل واحد من تلك الصورة.
* الطّبقات المخفيّة: هي الطّبقات بين الإدخال والإخراج. يمكن أن تحتوي الشّبكة على العديد من الطّبقات المخفيّة، ولهذا يُطلق على التّعلم العميق هذا الاسم. كل طبقة تأخذ الخرج من الطّبقة السّابقة وتحدّد ميزّات أكثر تعقيدًا. على سبيل المثال، قد تكتشف الطّبقة المخفيّة الأولى الحواف البسيطة، بينما قد تحدّد الطّبقة التّالية أشكالًا مثل العيون والأنف، وقد تتعرف طبقة لاحقة على وجه كامل.
* طبقة الخرج: هذه هي الطّبقة النّهائية التي تُعطي نتيجة تحليل الشّبكة. مثلاً في مهمة تصنيف الصّور، ستعطي طبقة الخرج احتمالًا لكل فئة ممكنة (على سبيل المثال، احتمال 85% أنّها "قطّة"، 10% "كلب"، 5% "ثعلب").



الشكل : شكل يوضح بنية الشبكة العصبية.

#### **2.2.3.2. الانحدار التّدريجيّ Gradient descent:**

الانحدار التّدريجيّ هو خوارزميّة التّحسين الأساسيّة المستخدمة في التّعلم الآليّ والتّعلم العميق لتدريب النّماذج. الهدف الرّئيسي منه هو العثور على قيم مُعلّمات النّموذج (مثل الأوزان والإنحيازات) التي تقلّل من دالة الخسارة، والتي تقيس مدى بعد تنبّؤات النّموذج عن القيم الصّحيحة الفعليّة [3].

#### **3.2.3.2. مُحسِّن Adam(Adaptive Moment Estimation):**

يعّد مُحسِّن Adam خوارزميّة تحسين شائعة وفعّالة للغاية في مجال التّعلم العميق والتّعلم الآلي. يرمز هذا المصطلح إلى تقدير الّلحظة التّكيّفيّة، وهو مصمّم للعثور بكفاءة على القيم المثلى لمعلّمات النّموذج (Parameters) عن طريق تحديثها بشكل متكرّر بناءً على بيانات التّدريب. يشتهر مُحسّن آدم بقدرته على الجمع بين أفضل ميزات خوارزميّتين آخرتين معروفتين للتّحسين: Momentum وRMSprop. يسمح هذا المزيج القويّ بمعدّل تعلّم تكيّفي، يتمّ تعديله لكل معلّمة على حدة، مما يؤدي إلى تقارب أسرع وأداء قوي عبر مجموعة واسعة من النماذج ومجموعات البيانات [3].

#### **4.2.3.2. دالة الخسارة Loss Function :**

تقيس دالّة الخسارة، المعروفة أيضاً باسم دالّة التّكلفة، مدى جودة أداء نموذج التّعلم الآلي. فهو يحدد الفرق بين المخرجات المتوقّعة للنّموذج والقيم المستهدفة الفعليّة. الهدف الأساسيّ أثناء عمليّة التّدريب هو ضبط مُعلّمات النّموذج لتقليل دالّة الخسارة هذه، وبالتالي جعل تنبّؤات النّموذج دقيقة قدر الإمكان. فيما يلي نذكر بعض دوال الخسارة المستخدمة [2]:

* متوسّط الخطأ التّربيعيّ Mean Squared Error (MSE):

متوسّط الخطأ التّربيعيّ هو دالّة الخسارة الأكثر شيوعاً لمسائل الانحدار حيث يكون الهدف هو تقدير قيمة ضمن مجالٍ مستمرٍّ من القيم (على سبيل المثال سعر المنزل، درجة الحرارة ،التّقييم المعطى لمطعم، ...).

* + آليّة عمله: يحسب متوسّط الفروق المربّعة بين القيم المتوقّعة والقيم الفعليّة. يؤدّي تربيع الفرق إلى معاقبة الأخطاء الأكبر بشكلٍ أكبر ويضمن أن تكون النّتيجة إيجابيّة دائماً.
  + معادلته:

حيث:

* + - : هو عدد نقاط البيانات.
    - : هي القيمة الفعلية.
    - : هي القيمة المتوقعة.
* الانتروبيّة الثّنائيّة المتقاطعة Binary Cross-Entropy :

يتم استخدام الانتروبّية الثّنائيّة المتقاطعة لمهام التّصنيف الثّنائيّ، حيث يكون الخرج أحد فئتين (على سبيل المثال "بريد عشوائيّ" أو "ليس بريداً عشوائياً").

* آليّة عمله: يقيس الاختلاف بين الاحتمال المتوقّع وعلامة الفئة الفعليّة (0 أو 1). إنه يعاقب النموذج بشدة عندما يقدم تنبؤًا واثقًا خاطئًا.
* معادلته:

حيث:

* : هو عدد العينات.
* : العلامة الحقيقية (0 أو 1) للعينة رقم i.
* : الاحتمال المتوقع (بين 0 و 1) للعينة رقم i.
* الإنتروبيّة المتقاطعة التّصنيفيّة Categorical Cross-Entropy :

الإنتروبيّة المتقاطعة التّصنيفيّة هي دالّة الخسارة المفضّلة لمشاكل التّصنيف متعدّدة الفئات، حيث يجب تصنيف المدخلات إلى واحدة من ثلاث فئات أو أكثر (على سبيل المثال تحديد ما إذا كان الحيوان الموجود في الصّورة "قطة" أو "كلب" أو "طائر").

* + آليّة عمله: إنه امتداد مباشر للإنتروبية الثنائية المتقاطعة. فهو يقارن توزيع الاحتمالات المتوقعة مع التوزيع الفعلي، حيث يكون احتمال الفئة الصحيحة فقط 1 وجميع الفئات الأخرى 0.
  + معادلته:

حيث:

* + - : عدد العينات.
    - : عدد الفئات.
    - : العلامة الصحيحة (1 إذا كانت العينة i تنتمي إلى الفئة c، وإلا 0).
    - : الاحتمال المتوقع أن العينة i تنتمي إلى الفئة c.

### **3.3.2. مقاييس التقييم Evaluation Metrics :**

#### **1.3.3.2. مقياس الرّبح التراكميّ المخصوم المطبَّع Normalized Discounted Cumulative Gain (NDCG):**

يُعدّ مقياس NDCG أحد أهم مقاييس التقييم المستخدمة في نظم التوصية ومهام استرجاع المعلومات، خاصةً عندما يكون ترتيب النتائج ذو أهمية قصوى. على عكس مقاييس الدقة البسيطة، لا يقيّم NDCG فقط ما إذا كانت العناصر الموصى بها ذات صلة، بل يأخذ في الاعتبار أيضًا موضعها في قائمة التوصيات[4] .

* **آلية عمله:** يقوم المقياس على مبدأين أساسيين:
  + العناصر ذات الصلة العالية أكثر قيمة من العناصر ذات الصلة المنخفضة.
  + العناصر التي تظهر في أعلى القائمة لها تأثير أكبر من تلك التي تظهر في الأسفل.
* لحساب ذلك، يقوم NDCG أولاً بحساب "الربح التراكمي المخصوم " (DCG)، الذي يمنح مكافأة أعلى للعناصر ذات الصلة التي تظهر في المراكز الأولى، مع تقليل قيمة الربح للعناصر التي تظهر في مواضع لاحقة. بعد ذلك، تتم "تطبيع" (Normalization) هذه القيمة بقسمتها على قيمة DCG المثالية (IDCG)، وهي أقصى قيمة ممكنة للربح إذا كانت القائمة مرتبة بشكل مثالي. ينتج عن ذلك درجة تتراوح بين 0 و 1، حيث يمثل 1 الترتيب المثالي. يسمح هذا التطبيع بمقارنة أداء النموذج عبر استعلامات أو قوائم مختلفة بأطوال متفاوتة.

### **4.3.2. عمليّة التّعلم:**

تتعلّم الشّبكة العصبيّة من خلال عمليّة تُسمّى التّدريب. وتتمايز طرق التّدريب بحسب نوع التّعلم المتّبع. فمثلاً في حالة التّعلم الخاضع للإشراف، يتم عرض الآلاف أو الملايين من الأمثلة من البيانات حيث تكون الإجابة الصّحيحة معروفة مسبقاً (على سبيل المثال، صور مُعلَّمة بعبارة "قطّة"). بالنّسبة لكل مثال، تقوم الشّبكة بإيجاد تخمين. ثم تقارن تخمينها بالإجابة الصّحيحة وتحسب الخطأ. يتم استخدام هذا الخطأ بعد ذلك بواسطة خوارزمية تحسين، مثل مُحسِّن Adam الفعال، لتعديل الاتصالات (الأوزان والانحيازات) بين الخلايا العصبيّة بطريقة تجعل تخمينها أكثر دقّة في المرة القادمة.تُكرر هذه العملية، المعروفة باسم الانتشار العكسي (Backpropagation)، مراراً وتكراراً . و من خلال تصحيح نفسها باستمرار بناءً على الملاحظات، تتعلّم الشّبكة تدريجيًا الأنماط الأساسيّة في البيانات، مما يجعلها أكثر دقّة في المهمّة الموكّلة إليها. ويكون الهدف الأساسي هنا هو التّنبؤ بالانتماء إلى مُعلّمة معيّنة. تبدأ بتحديد نقاط مركزيّة عشوائيّة للمجموعات التي تريد اكتشافها [2]. أمّا في التّعلم غير الخاضع للإشراف، يكون الهدف الأساسيّ هو الاكتشاف بدلاً من التّنبؤ. على عكس التّعلم الخاضع للإشراف، يتم تزويد الخوارزميّة ببيانات خام وغير مُعلّمة ويجب عليها أن تجد أنماط البيانات وهياكلها الخاصّة. لتحقيق ذلك، تقوم الخوارزميّة بتحليل البيانات بشكل متكرّر لتنظيمها بناءً على خصائصها الداخلية. وأشهر أساليب عملها هو التّجميع العنقوديّ (Clustering)، حيث تقوم الخوارزميّة بالآتي:

* تُعيّن كل نقطة بيانات إلى أقرب مركز لها، مُشكّلةً بذلك عناقيد أوليّة.
* تُحدّث موقع كل مركز عن طريق حساب متوسّط جميع النقاط في عنقوده.
* تُكرّر الخطوتين السّابقتين حتّى تستقرّ المجموعات ولا تعود للتّغير، كاشفةً بذلك عن الهيكل الطبيعيّ للبيانات.

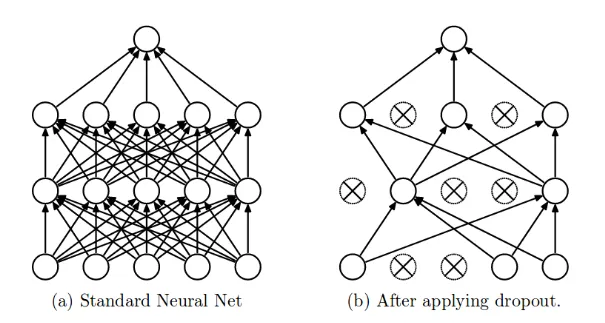
الهدف هنا ليس التّنبؤ بإجابة صحيحة، ولكن استكشاف الهيكل الداخليّ للبيانات واكتشاف مجموعات أو علاقات لم تكن معروفة من قبل [2].

#### **1.4.3.2. التّجهيز الزّائد (Over Fitting):**

يحدث التّجهيز الزّائد عندما يتعلّم نموذج التّعلم الآليّ بيانات التّدريب بشكل جيد للغاية، ويلتقط الضّوضاء والأنماط غير ذات الصّلة بدلاً من التّعميم على البيانات غير المرئيّة. ويؤدي هذا إلى دقّة عالية في بيانات التّدريب ولكن يكون الأداء ضعيفاً في بيانات الاختبار [2].

#### **2.4.3.2. التّسرّب Dropout :**

التسرب هو أسلوب تنظيمي (Regularization) يُستخدم لمنع التّجهيز الزّائد في الشّبكات العصبيّة. وهو يعمل عن طريق إيقاف جزء صغير من الخلايا العصبيّة بشكل عشوائيّ (أي تعطيله مؤقتاً) أثناء كل تكرار تدريبيّ، مما يجبر الشّبكة على تعلّم ميزات أكثر قوّة. يعمل التّسرب على تحسين التّعميم عن طريق تقليل الاعتماد على أيّ خليّة عصبيّة واحدة وتشجيع التّمثيلات الموزّعة [5].



الشكل : شكل يوضح الفرق بين حالة التدريب النمطية و استخدام التسرب.

#### **3.4.3.2. التّضمينات Embeddings:**

التّضمينات هي طريقة لتمثيل بيانات معقّدة وعالية الأبعاد، مثل الكلمات أو ملفّات تعريف المستخدمين، كمتّجهات رقميّة كثيفة في فضاءات ذات أبعاد أقل. الفكرة الأساسيّة وراء التّضمينات هي التقاط المعنى الأساسيّ والعلاقات بين العناصر. فبدلاً من مجرد إعطاء مُعرّف فريد لكل كلمة أو منتج، يضع التّضمين العناصر المتشابهة بالقرب من بعضها البعض في فضاء متعدّد الأبعاد.

#### **4.4.3.2. تحليل المصفوفات Matrix factorization:**

يقوم تحليل المصفوفة بتحليل مصفوفة كبيرة لتقييم المستخدم-المنتج إلى مصفوفتين أصغر وأسهل إدارة. تُمثّل إحدى المصفوفتين المستخدمين وانجذابهم لمجموعة من السّمات الخفيّة أو الكامنة (مثل النّوع أو الأسلوب). وتُمثل المصفوفة الأخرى العناصر ومدى امتلاكها لهذه السّمات نفسها. للتّنبؤ بالتّقييم، يأخذ النّموذج ببساطة حاصل الضّرب النّقطي لمتّجه المستخدم ومتّجه العنصر. و يتعلم النّظام عن طريق التّعديل على القيم في هاتين المصفوفتين الصّغيرتين حتى يُعيد حاصل ضربهما إلى حدّ كبير التّقييمات الأصليّة المعروفة [6].

#### **5.4.3.2. تشابه جيب التّمام Cosine Similarity :**

تشابه جيب التّمام هو مقياس يُستخدم لقياس التّشابه بين متّجهين غير صفريّين في فضاء متعدد الأبعاد. يتمّ حساب جيب تمام الزّاوية بين هذين المتّجهين، ممّا يُحدّد ما إذا كان المتّجهان يشيران إلى نفس الاتجاه تقريباً. وهو مفيد بشكل خاصّ في حالة النّظم التّوصية، لأنّه يُركّز على اتّجاه المتّجهات بدلاً من حجمها أو طولها[7].

## **4.2. نظام التّوصية Recommendation Systems:**

نظام التّوصيات هو نوع من النّظم تصفية المعلومات، يهدف إلى التّنبؤ بتقييم المستخدم أو تفضيله لعنصر ما. وهي التقنية التي تعتمد عليها نتفليكس في اقتراح الأفلام، وأمازون في عرض المنتجات، وسبوتيفاي في إنشاء قوائم تشغيل مخصصة. فيما يلي نعرض الفوائد الأساسيّة التي يقدّمها نظام التّوصية لكل من الشّركات والمستخدمين [8]:

* للشّركات: تُعدّ نظم التّوصية أدوات فعّالة لزيادة المبيعات وتفاعل المستخدمين. إضافةً إلى كونها مكّنت الشركات من زيادة عدد الزيارات لمنصّاتها، والحفاظ على بقاء المستخدمين على منصتها لفترة أطول من خلال عرض منتجات قيّمة للعملاء بحسب تفضيلاتهم الشّخصيّة.
* للمستخدمين: تساعدهم على اكتشاف محتوى جديد وذو صلة ضمن مجال واسع من الخيارات المتاحة. بحيث يُحسّن هذا التّخصيص تجربة المستخدم، موفرًا بذلك الوقت الذي يتمّ قضاؤه ليجدوا ما يعجبهم، إضافةً إلى أنّه يُعرّفهم على عناصر ربما لم يجرّبوها من قبل.

ولتقديم التّوصيات، تعتمد النّظم على تعليقات المستخدمين، والتي يمكن جمعها بطريقتين مختلفتين:

* ردود فعل صريحة Explicit Feedback:

هذه معلومات يُقدّمها المستخدم مباشرةً وبشكل مقصود حول تفضيلاته. وهي واضحة ودقيقة، ولا تترك مجالًا للغموض.

* + ميّزاتها: توفّر بيانات عالية الجودة لا غموض فيها حول ما يفضّله المستخدم وما يكرهه.
  + عيوبها: غالبًا ما يكون من الصّعب جمع المعلومات، حيث لا يقوم العديد من المستخدمين بوضع تقييمات صريحة.
* ردود الفعل الضمنية Implicit Feedback:

هذه معلومات لا يقدمها المستخدم مباشرةً، بل تُجمع من خلال مراقبة سلوكه. الهدف منها هو استنتاج تفضيلاته من خلال أفعاله.

* + ميّزاتها: وفيرة وسهلة التّجميع، حيث يقوم المستخدمون بتوليدها ببساطة عن طريق التّفاعل مع المنصّة.
  + عيوبها: قد تكون مُضَلّلة ويصعب تفسيرها. على سبيل المثال، هل اشترى المستخدم منتجًا كهدية لشخص آخر، أم توقف عن مشاهدة فيلم لأنه لم يُعجبه أو لأنه مُقاطع؟

### **1.4.2. أنوع التّوصيات:**

يتم تصنيف النّظم التّوصية بناءً على كيفيّة تقديم اقتراحاتها. فيما يلي نعرض الأنواع الأساسيّة من نظم التّوصية بناءً على الطّريقة التي تُقّدّم فيها اقتراحاتها [8]:

* نظم التّوصية المعتمدة على التّصفية التّعاونية Collaborative Filtering:

هذا هو النّهج الأكثر شيوعاً، ويعتمد على فكرة أن الأشخاص الذين لديهم اهتمامات متشابهة في الماضي سيكون لديهم اهتمامات متشابهة حاليّاً. يستخدم هذا النهج سلوك مجموعة كبيرة من المستخدمين لتقديم توصيات دون الحاجة إلى معرفة أي شيء عن المنتجات نفسها.

* نظم التّوصية المعتمدة على التصفية القائمة على المحتوى Content-Based Filtering:  
  تستخدم هذه الطريقة سمات أو "محتوى" عنصر ما لتقديم توصيات. يركّز هذا النّوع من نظم التّوصية على خصائص العناصر التي أبدى المستخدم إعجابه بها سابقاً، وتقترح له عناصر أخرى بخصائص مماثلة.
* نظم التّوصية المعتمدة على النماذج الهجينة Hybrid Models:  
  تجمع النّماذج الهجينة بين تقنيات التّصفية التّعاونية والقائمة على المحتوى. والهدف هو الاستفادة من نقاط قوّة كلا النّهجين لإنشاء توصيات أكثر متانة ودّقة و موائمة لتفضيلات المستخدم.

### **2.4.2. مشكلة البداية الباردة Cold Start Problem :**

مشكلة البداية الباردة شائعة في نظم التّوصية، حيث لا يستطيع النّظام استخلاص أيّ استنتاجات للمستخدمين أو العناصر التي لم يجمع عنها معلومات كافية بعد. هذا النّقص في البيانات يُصعّب تقديم توصيات دقيقة ومُخصّصة، إذ لا يمتلك النّظام تفاعلات سابقة ليتعلّم منها [8].

## **5.2. نموذج اللّغة الكبير (LLM) :**

نموذج اللّغة الكبير هو نوع متقدّم من الذّكاء الاصطناعيّ المدرّب على كميّات هائلة من البيانات النّصية، ممّا يسمح له بفهم اللّغة البشريّة وتوليدها والتّفاعل معها بطريقة متطوّرة.

### **1.5.2. ما هو نموذج اللّغة الكبير ؟**

في جوهره، يُعدّ نموذج اللّغة الكبير شبكةً عصبيّةً ضخمة، غالباً ما تحتوي على مليارات المتغيّرات الدّاخلية التي تُسمى المعلّمات Parameters. وهو مبنيٌّ على بنية تُسمّى "المحوِّل Transformer"، وهي فعّالةٌ بشكلٍ خاصّ في تحديد العلاقات المعقّدة والسّياق داخل البيانات المتسلسلة، مثل النّصوص. ومن خلال معالجة مجموعات بياناتٍ هائلة، يتعلّم النّموذج القواعدَ والحقائقَ وقدراتِ التّفكيرِ المنّطقيّ وأساليبَ التّواصلِ المختلفة.

### **2.5.2. ماذا يمكن أن يفعل نموذج اللّغة الكبير:**

إنّ الفهم العميق للّغة يمنح نماذج اللّغة الكبير قدرات واسعة، ممّا يجعلهم متعدّدي المهارات بشكل لا يُصدق. بحيث يمكنهم:

* إنشاء المحتوى: كتابة نصّ أصليّ، من رسائل البريد الإلكترونيّ والمقالات إلى الشعر و الرّمازات البرمجيّة.
* الإجابة على الأسئلة: تقديم إجابات مفصّلة لأسئلة معقّدة حول مجموعة واسعة من المواضيع.
* تلخيص المعلومات: يمكنهم تحويل المستندات أو المقالات الطويلة إلى ملخّصات موجزة.
* ترجمة اللّغات: تحويل النّص من لغة إلى أخرى بدقّة عالية.
* المشاركة في المحادثة: اللّب الأساسيّ لروبوتات المحادثة والمساعدين الافتراضييّن الذين يمكنهم إجراء محادثات دقيقة ومفيدة.

## **6.2. البنية المعماريّة Architecture:**

البنية المعمارية هي هيكل وتنظيم عال المستوى للنظم البرمجية تشتمل على قرارات التصميم الأساسية التي تشكل مكونات النظام وعلاقاتها وكيفية تفاعلها مع بعضها البعض، حيث يساعد بناء هندسة معمارية جيدة على تصميم نظام مرن قابل للصيانة والتكيف مع التغييرات بمرور الوقت[9].

## **7.2. البنية المعماريّة النّظيفة Clean Architecture:**

## **1.7.2. مفهوم المعماريّة النّظيفة:**

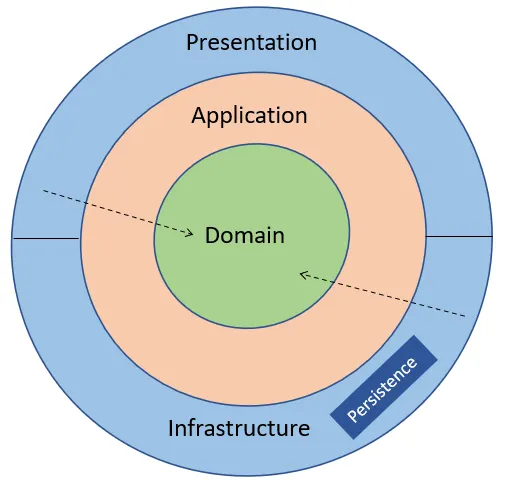
هي فلسفة لتصميم البرمجيات ونمط تصميمي معماري، يهدف إلى إنشاء نظام برمجي قابل للصيانة ومستقل عن تفاصيل التنفيذ. يعزز هذا النهج استقلالية المكونات المختلفة للنّظام، مثل واجهات المستخدم ومنطق الأعمال وتخزين البيانات وغيرها، مما يسمح لهذه المكّونات بالتطور بشكل مستقل دون التأثير على النظام بأكمله،كما توفّر هذه المنهجيّة قاعدة رماز (codebase) قابلة للاختبار، مما يسهل إدارة وتوسيع النظام البرنامجي.[9]

ينتج عن استخدام البنية النظيفة نظام يحمل الخواص التالية:[ 9]

* **مستقل عن أطر العمل (Independence of Frameworks):** لايرتبط منطق العمل الأساسي والمكونات
* الأساسية للنظام بأي إطار خارجي أو مكتبة أو تقنية معينة، الأمر الذي يسمح باستبدال أو تحديث هذه التبعيات الخارجية دون التأثير على الوظائف الأساسيّة للنظام.
* **استقلالية واجهات المستخدم (Independence of User Interface):** تكونواجهة المستخدم منفصلة عن بقية النظام، مما يسمح بإجراء تغييرات على واجهة المستخدم دون التأثير على منطق الأعمال الأساسي، بالإضافة إلى إمكانية وجود واجهات مستخدم متعددة (مثل الويب و الهاتف المحمول) تستخدم منطق الأعمال الأساسي نفسه.
* **استقلالية تخزين البيانات (Independence of Data Storage ):** يمكن تغيير آلية تخزين المعطيات دون التأثير على بقية أجزاء النظام.
* **نظام قابل للاختبار (Testable):** إمكانيةاختبار قواعد العمل بدون واجهة المستخدم أو قاعدة البيانات أو خادم الويب أو أي عنصر خارجي آخر.

## **2.7.2. قاعدة التبعية:**

تعد قاعدة التبعية مبدأ أساسيا يوّجه كيفية تنظيم المكونات والوحدات داخل النظام البرمجي وتفاعلها مع بعضها البعض في البنية النظيفة. تنص قاعدة التبعية على أن التبعيات يجب أن تشير دائما إلى الداخل نحو جوهر النظام، مما يعني أن المكونات ذات المستوى الأعلى والأكثر تجريدا وثباتا يجب أّلا تعتمد على المكونات ذات المستوى الأدنى، بدلا من ذلك، يجب أن تتدفق التبعيات من الطبقات الخارجية نحو الطبقات الداخلية للبنية [9].



الشكل : شكل يوضح المعمارية النظيفة مع قاعدة الاعتمادية.

## **8.2. أنماط التّصميم Design Patterns :**

أنماط التّصميم هي حلول مثبتة لمشاكل تصميم البرامج الشّائعة، حيث توفر قوالب قابلة لإعادة الاستخدام لكتابة أكواد نظيفة وقابلة للصّيانة وقابلة للتّطوير. وهي تنقسم إلى ثلاث فئات رئيسية [10]:

* الأنماط الإبداعية Creational Patterns : الهدف منها هو التّحكم في كيفيّة إنشاء الكائنات لتحسين المرونة وإعادة الاستخدام. نذكر منها:
  + Singleton
  + Factory
  + Abstract Factory
  + Builder
  + Prototype
* الأنماط الهيكلية Structural Patterns : تُركّز أنماط التّصميم الهيكليّة على كيفيّة تكوين الكائنات لتشكيل هياكل أكبر مع الحفاظ عليها مرنة وفعّالة ومنفصلة. الهدف الأساسي من أنماط التّصميم الهيكليّة هو تبسيط النّظم المعقدة، تمكين التوافق، إضافة وظيفة بشكلٍ ديناميكيّ، إدارة علاقات الكائنات و تحسين استخدام الموارد. نذكر منها :
  + Adapter
  + Decorator
  + Facade
  + Composite
  + Proxy
* الأنماط السلوكية Behavioral Patterns : تركّز الأنماط السّلوكية على كيفيّة تواصل الأشياء وتفاعلها مع بعضها البعض، مما يضمن المرونة والانفصال وتوزيع المسؤولية بكفاءة. الهدف الأساسي من أنماط التّصميم الهيكليّة هو فصل المرسلين والمستقبلين، تغليف الخوارزميّات، إدارة سير الأعمال المعقّدة و تبسيط التّعاون بين الكائنات. نذكر منها :
  + Observer
  + Strategy
  + Command
  + Mediator
  + Chain of Responsibility

## **9.2. فصل الاستعلامات و الأوامر Command Query Segregation (CQS) :**

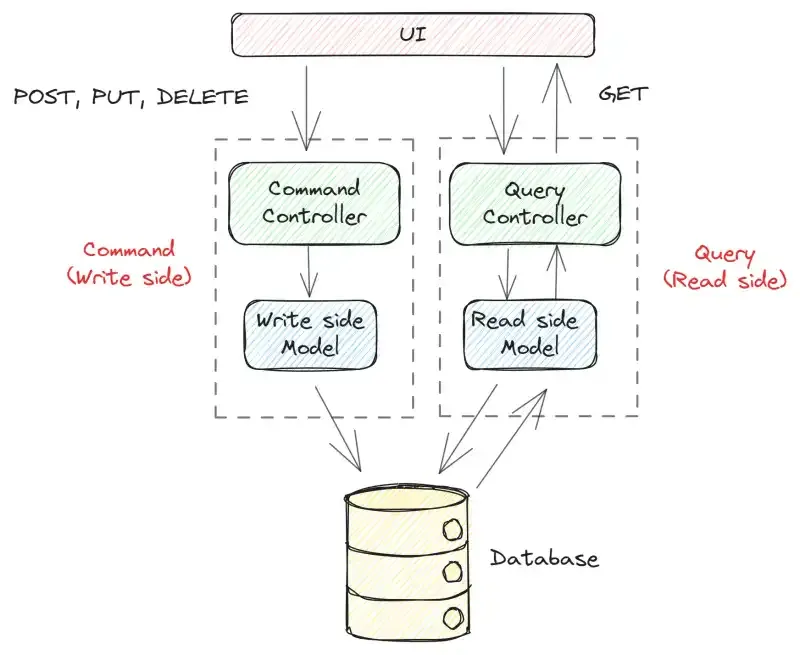
هو نمط تصميمي يهدف إلى فصل التوابع (Methods) إلى فئتين ضمن الغرض أو الصف (object class)، طرق مسؤولة عن الاستعلام عن المعلومات ضمن النظام (Queries) و أخرى مسؤولة عن إرسال أوامر للنظام لتغيير حالته الداخلية من تعديل وحذف وإدخال. يسّهل استخدام CQS تحديد مسؤوليّة كل طريقة ضمن النظام، حيث أن مستخدم النظام يستطيع طلب استعلام أو إرسال أمر من/إلى النظام لكن ليس كلاهما في نفس الوقت. ظهر مفهوم CQS لأول مرة في عام 1988عندما اقترح Meyer أنه ينبغي تقسيم الأساليب إلى نوعين؛ الوظائف والإجراءات. الوظائف تعطي نتائج لكنها لا تؤثر على حالة النظام أما الإجراءات تؤثر صراحة على الحالة ولكنها لا تقدم نتيجة [11].

## **10.2. فصل مسؤولية الاستعلام و الأوامر Command Query Responsibility Segregation:**

ظهر النمط التصميمي CQRS انطلاقا من النمط CQS، مع اختلاف بسيط، يتمثّل في فصل مسؤوليات النظام لأوامر واستعلامات على المستوى المعماري بدلا من مستوى الغرض (object) حيث يتّم إنشاء وحدات (modules) مسؤولة عن معالجة الاستعلامات، وأخرى مسؤولة عن معالجة الأوامر الواردة إلى النظام [11].

تتلخّص أهميّة استخدام CQRS في:

* **تحسين الأداء**: من خلال وجود وحدات منفصلة لمعالجة الأوامر والاستعلامات، يمكن العمل على كل منها بشكل منفصل وتحسين كل وحدة بشكل مستقل.
* **المرونة**: يمكن من خلال تطبيق CQRS استخدام بنى معطيات ومخازن للبيانات في وحدات القراءة مختلفة عن وحدات الكتابة.
* إدارة التعقيد ضمن النظام: يساعد CQRS في فصل التعقيد الناجم عن معالجة الاستعلامات عن ذلك الناجم عن معالجة الأوامر، الأمر الذي يساعد في إنتاج رماز نظيف وقابل للصيانة



الشكل : شكل يوضح الفصل إلى استعلامات و أوامر في حالة ال CQRS.

# الفصل الثالث

**الدّراسة المرجعيّة**

*يوضح هذا الفصل عملية تحليل النظام ودراسة متطلباته وصولا لتصميم النظام*

## **1.3. الدراسة المرجعية لنظام التوصية:**

تستعرض هذه الدراسة الأبحاث الحالية حول نظم توصية المطاعم المُصممة باستخدام قاعدة بيانات Yelp. وينصبّ التركيز على وضع التقنيات المُستخدمة في هذا المشروع في سياقها الصحيح من خلال دراسة الأعمال الأكاديمية المُعتمدة في ثلاثة مجالات رئيسية: التصفية التعاونية، وتحليل المحتوى باستخدام نماذج اللغة الكبيرة (LLMs)، والأنظمة الهجينة التي تجمع بين كلا النهجين.

### **1.1.3. لمحة عن مجموعة البيانات المستخدمة:**

مجموعة بيانات Yelp هي مجموعة بيانات عامة ومشهورة تُستخدم على نطاق واسع في الأوساط الأكاديمية والتعليمية. تحتوي على معلومات حقيقية ومجهولة المصدر من موقع Yelp، وتشمل مراجعات المستخدمين، وبيانات الشركات والمتاجر، وملفات تعريف المستخدمين من عدة مناطق حضرية حول العالم. تعد هذه المجموعة مصدرًا غنيًا لتحليل البيانات، ومعالجة اللغات الطبيعية، وتعلم الآلة.

#### **1.1.1.3. شرح ملفات البيانات:**

فيما يلي وصف لكل ملف من ملفات JSON التي تتكون منها مجموعة البيانات [12]:

1. business.json: يحتوي هذا الملف على بيانات الشركات والمتاجر، بما في ذلك مواقعها وخصائصها وفئاتها. يحتوي هذا الملف على الحقول التالية:

* business\_id: معرّف فريد خاص بالمتجر.
* name: اسم المتجر أو الشركة.
* address, city, state, postal\_code: عنوان المتجر (الشارع، المدينة، الولاية، الرمز البريدي).
* latitude, longitude: إحداثيات الموقع الجغرافي للمتجر.
* stars: متوسط تقييم المتجر بالنجوم (من 1.0 إلى 5.0).
* review\_count: إجمالي عدد المراجعات التي تلقاها المتجر.
* is\_open: علامة توضح إذا كان المتجر مفتوحاً حالياً (1) أم مغلقاً (0).
* attributes: يحتوي على خصائص متنوعة للمتجر (مثال: {"GoodForKids": "True"}).
* categories: نص يحتوي على فئات المتجر مفصولة بفواصل (مثال: "Restaurants, Food, Delis").
* hours: يوضح ساعات العمل لكل يوم من أيام الأسبوع.

2. review.json: يحتوي على النصوص الكاملة للمراجعات التي كتبها المستخدمون. يحتوي هذا الملف على الحقول التالية:

* review\_id: معرّف فريد خاص بالمراجعة.
* user\_id: معرّف المستخدم الذي كتب المراجعة.
* business\_id: معرّف المتجر الذي تمت مراجعته.
* stars: تقييم النجوم الذي منحه المستخدم في هذه المراجعة.
* date: تاريخ ووقت كتابة المراجعة.
* text: النص الكامل للمراجعة.
* useful, funny, cool: عدد الأصوات التي تلقتها المراجعة لكل فئة (مفيد، مضحك، رائع).

3. user.json: يحتوي على بيانات ملفات تعريف المستخدمين. يحتوي هذا الملف على الحقول التالية:

* user\_id: معرّف فريد خاص بالمستخدم.
* name: الاسم الأول للمستخدم.
* review\_count: إجمالي عدد المراجعات التي كتبها هذا المستخدم.
* yelping\_since: تاريخ انضمام المستخدم إلى Yelp.
* friends: قائمة بمعرفات المستخدمين الأصدقاء، مفصولة بفواصل.
* useful, funny, cool: إجمالي عدد الأصوات التي تلقتها جميع مراجعات المستخدم.
* fans: عدد المعجبين لدى المستخدم.
* average\_stars: متوسط تقييم النجوم الذي يمنحه المستخدم في مراجعاته.
* compliment\_\*: حقول متنوعة تحسب أنواع الإطراءات التي تلقاها المستخدم (مثل compliment\_hot).

4. checkin.json: يحتوي على بيانات تسجيل الحضور (check-in) في الشركات. يحتوي هذا الملف على الحقول التالية:

* business\_id: معرّف المتجر الذي تم تسجيل الحضور فيه.
* date: نص يحتوي على الطوابع الزمنية لتسجيلات الحضور، مفصولة بفواصل.

5. tip.json: يحتوي على نصائح قصيرة وموجزة يتركها المستخدمون للشركات. يحتوي هذا الملف على الحقول التالية:

* text: محتوى النصيحة.
* date: تاريخ ووقت كتابة النصيحة.
* compliment\_count: عدد الإطراءات التي تلقتها هذه النصيحة.
* business\_id: معرّف المتجر الخاص بالنصيحة.
* user\_id: معرّف المستخدم الذي كتب النصيحة.

6. photo.json: يحتوي على معلومات وصور خاصة بالشركات. يحتوي هذا الملف على الحقول التالية:

* photo\_id: معرّف فريد للصورة.
* business\_id: معرّف المتجر المرتبطة به الصورة.
* caption: التعليق النصي الذي أضافه المستخدم للصورة.
* label: تسمية تصنف الصورة (مثال: "طعام"، "شراب"، "داخل المتجر").

### **2.1.3. نماذج التصفية التعاونية:**

تُعدّ التصفية التعاونية نهجًا أساسيًا في أنظمة التوصية، حيث تستفيد من تفاعلات المستخدم مع العناصر. غالبًا ما تستخدم التطبيقات الحديثة أطر التعلم العميق لتعلّم الميزات الكامنة للمستخدمين والعناصر.

اقترحت دراسة أُجريت عام ٢٠٢٢ نظام توصية قائم على التعلم العميق باستخدام التصفية التعاونية على مجموعة بيانات Yelp. بُني نموذجهم باستخدام Keras وTensorFlow، متضمنًا طبقات تضمين، وطبقة تسلسل، وطبقة كثيفة. قُيّم أداء هذا النموذج باستخدام مقياس جذر متوسط مربع الخطأ (RMSE). بعد التدريب على مدار ٥٠ حقبة، حقق النظام قيمة RMSE قدرها ٠.٢٠، وهي نتيجة تُعتبر تحسنًا ملحوظًا مقارنةً بالنماذج الحالية آنذاك. يُثبت هذا صحة استخدام الشبكات العصبية لتطبيق تحليل العوامل المصفوفية للتنبؤ بتقييمات المستخدمين [13].

### **3.1.3. الاستفادة من نص المراجعة مع نماذج اللغة الكبيرة:**

رغم أهمية التقييمات، تحتوي المراجعات التي يُنشئها المستخدمون على معلومات دلالية ثرية تُحسّن جودة التوصيات بشكل كبير. تُركز الأبحاث الحديثة على استخدام نماذج لغوية مُدرّبة مسبقًا لاستخراج هذه البيانات النصية وفهمها.

على سبيل المثال، قدّمت ورقة بحثية نُشرت عام ٢٠٢٣ نظام "A-Seof"، وهو نظام قائم على التركيز على توصيات المطاعم. يتعلّم هذا النهج تمثيلاتٍ واعيةً للدلالات للمستخدمين والمطاعم من خلال استخدام نموذج BERT مُدرّب مسبقًا لتوليد تضمينات سياقية من نصوص المراجعات. من خلال التقييم على مجموعة بيانات Yelp، أظهر نموذج A-Seof أداءً متفوقًا مقارنةً بالطرق الأساسية، محققًا تحسنًا متوسطًا بنسبة ٤.٣٪ من حيث Recall@10 وNDCG@10. يُبرز هذا العمل قيمة استخدام التضمينات المُولّدة من LLM لالتقاط الفروق الدقيقة في نصوص المراجعات، ويؤكد أن NDCG@10 هو مقياس قياسي لتقييم مهام التصنيف هذه [14].

### **4.1.3. أنظمة التوصية الهجينة:**

تهدف الأنظمة الهجينة إلى تحقيق أداء متفوق من خلال الجمع بين مزايا التصفية التعاونية والنماذج القائمة على المحتوى. ويُعد هذا فعالاً بشكل خاص في مجالات مثل توصيات المطاعم، حيث تتوافر بيانات التقييم والمراجعات النصية بكثرة.

اقترحت دراسة أُجريت عام ٢٠٢٠ نموذج تصفية تعاونية هجينة قائمة على BERT (B-HCF)، والذي تم اختباره خصيصًا على مجموعة بيانات Yelp. تضمنت منهجيتهم إنشاء نموذج قائم على المحتوى باستخدام خصائص مستخرجة من النص عبر نموذج BERT مُدرّب مسبقًا، ودمجه مع نموذج تصفية تعاونية. تم توليد التنبؤ النهائي بالتقييم بإضافة مخرجات النموذج القائم على المحتوى كميزة إدخال لمكون التصفية التعاونية. حققت هذه الاستراتيجية الهجينة معامل خطأ تربيعي متوسط (RMSE) قدره ١٫٠٧٧، وهو ما يمثل تحسنًا بنسبة ٤٫١٪ مقارنةً بأفضل نموذج أساسي أداءً. يُظهر هذا أن الجمع التآزري بين بيانات التقييم والنص يمكن أن يؤدي إلى توصيات أكثر دقة وفعالية[15].

## **2.3. الدراسة المرجعية لنظام الحجوزات:**

### **1.2.3. وضع السّوق والمنافسة:**

يُعد سوق الضّيافة الرّقميّة سوقاً ناضجاً وعالي التّنافسيّة، تهيمن عليه منصّات كبيرة ومتخّصصّة. تكشف النّظرة العامّة عن تجزئة كبيرة بين نظم حجز أماكن الإقامة والمطاعم. تُحلّل هذه الدّراسة موقع المشروع مقارنةً بروّاد السّوق الرّاسخين، وتُبرز مزاياه الاستراتيجيّة الرّئيسيّة.

### **2.2.3. نظرة عامّة على السّوق:**

يتميّز قطّاع السّفر والحجز عبر الإنترنت بالتّخصّص العمودي. يتفاعل المستهلكون عادةً مع منصّات منفصلة لتلبية احتياجاتهم المختلفة:

* حجز أماكن الإقامة: يُعدّ Booking.com رائد السّوق، وهو منصّة عالميّة توفّر مجموعة واسعة من الفنادق والشّقق وخيارات الإقامة الأخرى. يتمحور نموذج أعمالها حول تجميع عدد كبير من العقارات وتبسيط عمليّة الحجز للمسافرين.
* حجوزات المطاعم: يقود هذا القطّاع منصّات مثل OpenTable و TheFork و هي شركة تابعة لـ TripAdvisor. يركّز هذان القطّاعان حصرياً على تجربة تناول الطّعام، مما يتيح للمستخدمين إمكانيّة اكتشاف مجموعة واسعة من المطاعم وتقييّمها وحجز طاولات فيها.
* حلول الأنظمة المغلقة: تُقدّم سلاسل الفنادق الكبرى، مثل ماريوت (بونفوي) وهيلتون (أونورز)، تطبيقاتها الخاصّة للهواتف المحمولة. تُوفّر هذه التّطبيقات تجربة شاملة للنّزلاء، تشمل حجز الغرف والمطاعم في الفندق، ولكنّها تقتصر على مجموعة علاماتها التّجارية الخاصّة.

### **3.2.3. التّحليل التّنافسيّ ونقاط القوّة في المشروع:**

بالمقارنة مع النّظرة العامّة، يوضّح المشروع عدّة نقاط رئيسيّة للتّمييز والقوة التنافسية و هي:

* **ميزّة النّظام المتكامل:**

يكمن الابتكار الأساسيّ للمشروع في بنيته الموحّدة لحجوزات الفنادق والمطاعم. بخلاف نهج المنصّتين حيث يتعيّن على المستخدم التّبديل بين تطبيق مثل Booking.com وآخر مثل OpenTable، يوفّر هذا النّظام رحلة متكاملة. يمكن للمستخدم حجز إقامته ووجباته في سير عمل واحد، مما يُعالج مشكلة واضحة في السّوق الحاليّة. هذا يُقدّم قيمة مضافة قويّة للمستهلكين الباحثين عن الرّاحة وللشركات (مثل الفنادق التي تضمّ مطاعم عامّة) التي ترغب في تقديم خدمة مُبسّطة.

* مجموعة التكنولوجيا الحديثة والفعالة المُستخدمة: يوفّر الأساس التّقني للمشروع ميّزة كبيرة من حيث المرونة والفعّاليّة من حيث التّكلفة.
  + يتيح استخدام إطار العمل Flutter للواجهة الأماميّة Frontend نشر تطبيق واحد عالي الجودة عبر نظاميّ iOS و Android من قاعدة بيانات واحدة. هذا يُقلّل بشكل كبير من تكاليف التّطوير والصّيانة مقارنةً بالمنافسين الذين غالباً ما يحتاجون إلى دعم تطبيقات أصليّة منفصلة.
  + الواجهة الخلفيّة Backend، المبنية باستخدام ASP.NET Core، هي إطار عمل قويّ وعالي الأداء قادر على دعم واجهة برمجة تطبيقات API آمنة و قابلة للتّطوير. كما يضمن التزام المشروع ببنية نظيفة قابليّة صيانة النّظام وجاهزيّته للتّوسع مستقبلاً، مضاهياً للمعايير الهندسيّة في شركات التّكنولوجيا الرّائدة.
  + نموذج عمل مرن ومبتكر: صُمّمت المنصّة لتكون سوقاً مفتوحاً للشّركات المستقلّة، وليست نظاماً مغلقاً. ممّا يجعلها حلاً شاملاً للفنادق والمطاعم التي تعتمد حاليّاً على خدمات متعدّدة ومنفصلة لتواجدها الإلكتروني. من خلال توفير نظام حجز متكامل، يمكن للمشروع أن يوفّر بديلاً أكثر فعّاليّة من حيث التّكلفة وأبسط من النّاحية التّشغيليّة لأصحاب الأعمال.

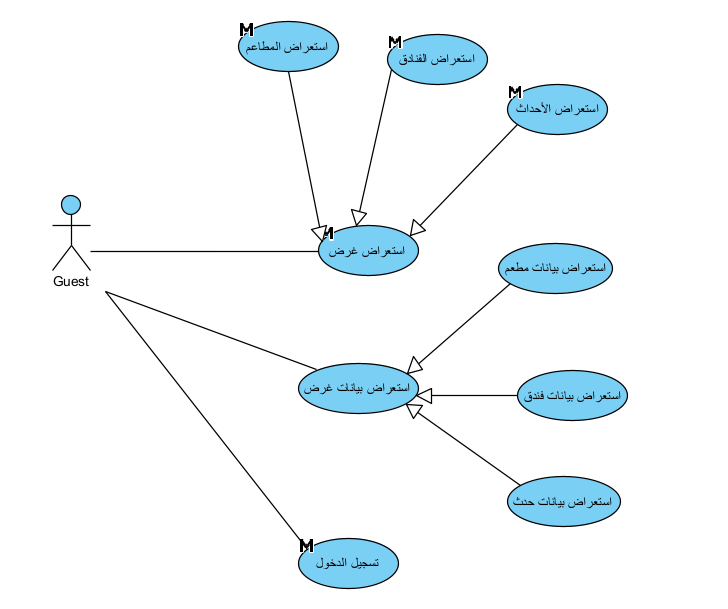
# الفصل الرابع

**الدراسة التحليلية**

*يوضح هذا الفصل عملية تحليل النظام ودراسة متطلباته وصولا لتصميم النظام*

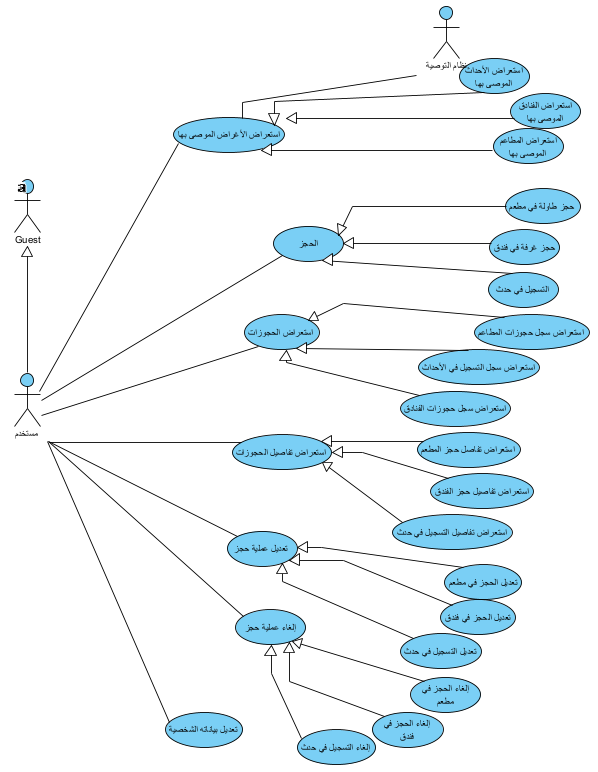
## **1.4. مخططات حالات الاستخدام:**

### **1.1.4. حالات استخدام الضيف:**



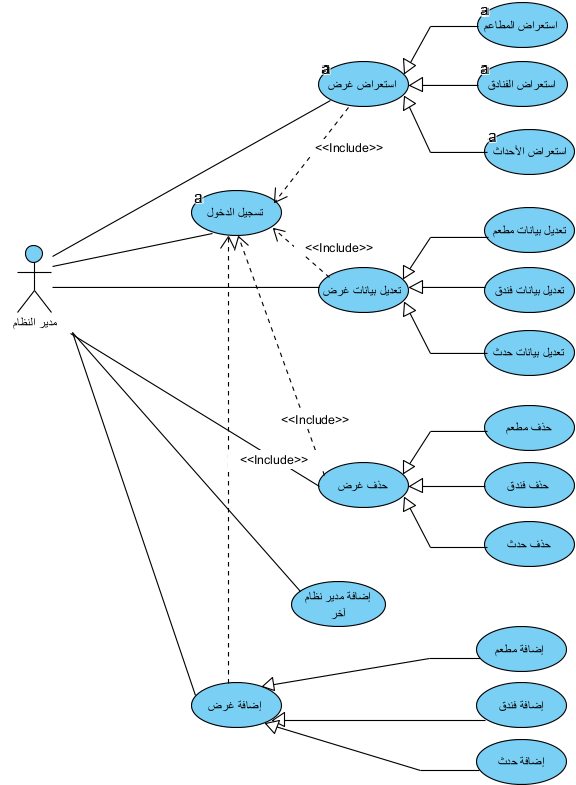
الشكل : شكل يوضح حالات استخدام الضيف.

### **2.1.4. حالات استخدام الزبون:**



الشكل : شكل يوضح حالات استخدام الزبون.

### **3.1.4. حالات استخدام مدير النظام:**



الشكل : شكل يوضح حالات استخدام مدير النظام.

## **2.4. السرد النصي لحالات الاستخدام:**

### **1.2.4. حالات استخدام الضيف:**

#### **1.1.2.4. استعراض الأغراض:**

الجدول حالة استعراض الأغراض (مطاعم/ فنادق/ أحداث).

|  |  |
| --- | --- |
| اسم الحالة: استعراض الأغراض. | |
| الوصف Description | يقوم الضيف باستعراض الأغراض (مطاعم/فنادق/ أحداث) المتوفّرة لدى النّظام. |
| الفاعلون Actors | الضيف. |
| الشّروط السّابقة Preconditions | لا يوجد. |
| الشّروط اللاحقة Postconditions | تمَّ عرض مجموعة من الأغراض. |

**سير الأحداث:**

**السيناريو الناجح الأساسي:**

الجدول السيناريو الناجح لحالة استعراض الأغراض (مطاعم/ فنادق/ أحداث).

|  |  |
| --- | --- |
| الضيف | نظام حجز المطاعم و الفنادق |
| 1. يقوم الضيف بطلب استعراض الأغراض. |  |
|  | 2. يقوم النظام بعرض مجموعة من الأغراض. |

**المسارات البديلة:**

* لا يوجد.

**مسارات الأخطاء:**

* لا يوجد.

#### **2.1.2.4. استعراض غرض ما:**

الجدول حالة استعراض غرض ما (مطاعم/ فنادق/ أحداث).

|  |  |
| --- | --- |
| اسم الحالة: استعراض غرض ما. | |
| الوصف Description: | تسمح للمستخدم بالحصول على بيانات غرض ما (مطعم/ فندق/ حدث). |
| الفاعلون Actors: | الضيف. |
| الشّروط السّابقة Preconditions: | الغرض المطلوب موجود لدى النظام. |
| الشّروط اللاحقة Postconditions: | تم عرض بيانات غرض ما. |

**سير الأحداث:**

**السيناريو الناجح الأساسي:**

الجدول السيناريو الناجح لحالة استعراض غرض ما (مطاعم/ فنادق/ أحداث).

|  |  |
| --- | --- |
| الضيف | نظام حجز المطاعم و الفنادق |
| 1. يقوم الضيف بطلب الاستعلام عن بيانات غرض ما (مطعم /فندق/ حدث). |  |
|  | 2. يقوم النظام بالتحقق من وجود الغرض لديه. |
|  | 3. يقوم النظام بعرض بيانات الغرض المطلوب. |

**المسارات البديلة:**

* لا يوجد.

**مسارات الأخطاء:**

* في حال كان الغرض غير موجود، يعرض النظام ذلك للضيف.

#### **3.1.2.4. تسجيل الدخول:**

الجدول لحالة تسجيل الدخول.

|  |  |
| --- | --- |
| اسم الحالة: تسجيل الدخول. | |
| الوصف Description: | تسمح للمستخدم بالوصول إلى حسابه الشخصي باستخدام اسم المستخدم و كلمة المرور الخاصة به. |
| الفاعلون Actors: | الضيف. |
| الشّروط السّابقة Preconditions: | أن يمتلك المستخدم حساباً مسبقاً في النظام. |
| الشّروط اللاحقة Postconditions: | تم تسجيل دخول المستخدم بنجاح إلى النظام. |

**سير الأحداث:**

**السيناريو الناجح الأساسي:**

الجدول السيناريو الناجح لحالة تسجيل الدخول.

|  |  |
| --- | --- |
| الضيف | نظام حجز المطاعم و الفنادق |
| 1. يقوم الضيف بإدخال اسم المستخدم وكلمة المرور ثم يطلب تسجيل الدخول. |  |
|  | 2. يقوم النظام بالتحقق من صحة البيانات المُدخلة. |
|  | 3. يسمح النظام للضيف بالدخول إلى حسابه. |

**المسارات البديلة:**

* لا يوجد.

**مسارات الأخطاء:**

* في حال كانت البيانات المدخلة غير صحيحة، يعرض النظام ذلك للمستخدم.

### **2.2.4. حالات استخدام المستخدم:**

#### **1.2.2.4. استعراض الأغراض الموصى بها:**

الجدول لحالة استعراض الأغراض الموصى بها (مطاعم/ فنادق/ أحداث).

|  |  |
| --- | --- |
| اسم الحالة: استعراض الأغراض الموصى بها. | |
| الوصف Description: | يقوم المستخدم باستعراض الأغراض الموصى بها المتوفّرة لدى النّظام. |
| الفاعلون Actors: | المستخدم. |
| الشّروط السّابقة Preconditions: | يجب أن يكون المستخدم قد قام بتسجيل الدخول. |
| الشّروط اللاحقة Postconditions: | تمَّ عرض مجموعة من الفنادق. |

**سير الأحداث:**

**السيناريو الناجح الأساسي:**

الجدول السيناريو الناجح لحالة استعراض الأغراض الموصى بها (مطاعم/ فنادق/ أحداث).

|  |  |
| --- | --- |
| المستخدم | نظام حجز المطاعم و الفنادق |
| 1. يقوم المستخدم بطلب استعراض الأغراض الموصى بها. |  |
|  | 2. يقوم النظام بالتواصل مع نظام التوصية للحصول على الأغراض الموصى بها. |
|  | 3. يقوم النظام بعض مجموعة الأغراض الموصى بها. |

**المسارات البديلة:**

* لا يوجد.

**مسارات الأخطاء:**

* لا يوجد.

#### **2.2.2.4. تعديل البيانات الشخصية:**

الجدول لحالة تعديل البيانات الشخصية.

|  |  |
| --- | --- |
| اسم الحالة: تعديل البيانات الشخصية. | |
| الوصف Description: | تسمح للمستخدم بتعديل وتحديث بياناته الشخصية المسجلة في ملفه الشخصي. |
| الفاعلون Actors: | المستخدم. |
| الشّروط السّابقة Preconditions: | يجب أن يكون المستخدم قد قام بتسجيل الدخول إلى النظام. |
| الشّروط اللاحقة Postconditions: | تم حفظ البيانات الشخصية الجديدة للمستخدم. |

**سير الأحداث:**

**السيناريو الناجح الأساسي:**

الجدول السيناريو الناجح لحالة تعديل البيانات الشخصية.

|  |  |
| --- | --- |
| المستخدم | نظام حجز المطاعم و الفنادق |
| 1. يطلب المستخدم تعديل بياناته الشخصية. |  |
|  | 2. يطلب النظام البيانات الجديدة للتعديل. |
| 3. يقوم المستخدم بإدخال البيانات الجديدة ثم يطلب حفظها. |  |
|  | 4. يقوم النظام بالتحقق من صحة البيانات المُدخلة وحفظها. |

**المسارات البديلة:**

* لا يوجد.

**مسارات الأخطاء:**

* في حال كانت البيانات المُدخلة غير صالحة يعرض النظام ذلك.

#### **3.2.2.4. الحجز:**

الجدول لحالة الحجز.

|  |  |
| --- | --- |
| اسم الحالة: الحجز. | |
| الوصف Description: | يقوم المستخدم بإجراء حجز في مطعم أو فندق ، مع تحديد البيانات اللازمة للحجز. |
| الفاعلون Actors: | المستخدم. |
| الشّروط السّابقة Preconditions: | يجب أن يكون المستخدم قد قام بتسجيل الدخول ، وأن يكون قد اختار مطعماً أو فندقاً لإجراء الحجز فيه. |
| الشّروط اللاحقة Postconditions: | تم إنشاء حجز جديد للمستخدم في النظام. |

**سير الأحداث:**

**السيناريو الناجح الأساسي:**

الجدول السيناريو الناجح لحالة الحجز.

|  |  |
| --- | --- |
| المستخدم | نظام حجز المطاعم و الفنادق |
| 1. يختار المستخدم فندق أو مطعم و يطلب الحجز فيه. |  |
|  | 2. يطلب النظام من المستخدم إدخال تفاصيل الحجز. |
| 3. يقوم المستخدم بإدخال التفاصيل وتأكيد الحجز. |  |
|  | 3. يتحقق النظام من توفر الخدمة (طاولة في حالة المطعم أو غرفة في حالة الفندق) حسب تفاصيل الحجز و يقوم بتثبيت الحجز. |

**المسارات البديلة:**

* لا يوجد.

**مسارات الأخطاء:**

* إذا كانت الخدمة المطلوبة غير متاحة في بحسب تفاصيل الحجز، يعرض النظام ذلك للمستخدم.

#### **4.2.2.4. إضافة مراجعة:**

الجدول لحالة إضافة مراجعة.

|  |  |
| --- | --- |
| اسم الحالة: إضافة مراجعة. | |
| الوصف Description: | تسمح للمستخدم بإضافة تعليق أو مراجعة لفندق أو مطعم معين. |
| الفاعلون Actors: | المستخدم. |
| الشّروط السّابقة Preconditions: | يجب أن يكون المستخدم قد قام بتسجيل الدخول. |
| الشّروط اللاحقة Postconditions: | تم حفظ التعليق الجديد وربطه بالجهة المعنية (المطعم أو الفندق). |

**سير الأحداث:**

**السيناريو الناجح الأساسي:**

الجدول السيناريو الناجح لحالة إضافة مراجعة.

|  |  |
| --- | --- |
| المستخدم | نظام حجز المطاعم و الفنادق |
| 1. يختار المستخدم مطعماً أو فندقاً لإضافة مراجعة له. |  |
|  | 2. يطلب النظام من المستخدم إدخال بيانات المراجعة. |
| 3. يقوم المستخدم بإدخال بيانات المراجعة المطلوبة. |  |
|  | 4. يقوم النظام بحفظ المراجعة. |

**المسارات البديلة:**

* لا يوجد.

**مسارات الأخطاء:**

* في حال كانت بيانات المراجعة غير صالحة يعرض النظام ذلك و يطلب من المستخدم تعديلها.

#### **5.2.2.4. استعراض سجل الحجوزات:**

الجدول لحالة استعراض سجل الحجوزات.

|  |  |
| --- | --- |
| اسم الحالة: استعراض سجل الحجوزات. | |
| الوصف Description: | يقوم المستخدم بالاطلاع على قائمة حجوزاته السابقة والقادمة سواء لدى المطاعم أو الفنادق. |
| الفاعلون Actors: | المستخدم. |
| الشّروط السّابقة Preconditions: | يجب أن يكون المستخدم قد قام بتسجيل الدخول إلى النظام. |
| الشّروط اللاحقة Postconditions: | تم عرض قائمة تحتوي على جميع حجوزات المستخدم لدى لمطاعم أو الفنادق. |

**سير الأحداث:**

**السيناريو الناجح الأساسي:**

الجدول السيناريو الناجح لحالة استعراض سجل الحجوزات.

|  |  |
| --- | --- |
| المستخدم | نظام حجز المطاعم و الفنادق |
| 1. يطلب المستخدم عرض سجل حجوزاته. |  |
|  | 2. يقوم النظام باسترداد جميع الحجوزات المرتبطة بحساب المستخدم وعرضها. |

**المسارات البديلة:**

* يمكن للمستخدم اختيار حجز معين من القائمة لعرض تفاصيله الكاملة.

**مسارات الأخطاء:**

* لا يوجد.

#### **6.2.2.4. تعديل الحجز:**

الجدول لحالة تعديل الحجز.

|  |  |
| --- | --- |
| اسم الحالة: تعديل الحجز. | |
| الوصف Description: | تسمح للمستخدم بتعديل تفاصيل حجز قائم في فندق أو مطعم. |
| الفاعلون Actors: | المستخدم. |
| الشّروط السّابقة Preconditions: | يجب أن يكون المستخدم قد قام بتسجيل الدخول ولديه حجز قابل للتعديل. |
| الشّروط اللاحقة Postconditions: | تم تحديث تفاصيل الحجز في النظام. |

**سير الأحداث:**

**السيناريو الناجح الأساسي:**

الجدول السيناريو الناجح لحالة تعديل الحجز.

|  |  |
| --- | --- |
| المستخدم | نظام حجز المطاعم و الفنادق |
| 1. يختار المستخدم حجزاً من سجله ويطلب تعديله. |  |
|  | 2. يعرض النظام تفاصيل الحجز الحالية مع إمكانية تعديلها. |
| 3. يقوم المستخدم بإدخال التعديلات الجديدة و تأكيدها. |  |
|  | يتحقق النظام من إمكانية إجراء التعديل و يقوم بحفظ التغييرات. |

**المسارات البديلة:**

* لا يوجد.

**مسارات الأخطاء:**

* إذا كان التعديل المطلوب غير ممكن، يعرض النظام ذلك مع توضح سبب الرفض.

#### **7.2.2.4. إلغاء الحجز:**

الجدول لحالة إلغاء الحجز.

|  |  |
| --- | --- |
| اسم الحالة: إلغاء الحجز. | |
| الوصف Description: | تسمح للمستخدم بإلغاء حجز قائم في فندق أو مطعم. |
| الفاعلون Actors: | المستخدم. |
| الشّروط السّابقة Preconditions: | يجب أن يكون المستخدم قد قام بتسجيل الدخول ولديه حجز قابل للإلغاء. |
| الشّروط اللاحقة Postconditions: | تم إلغاء الحجز. |

**سير الأحداث:**

**السيناريو الناجح الأساسي:**

الجدول السيناريو الناجح لحالة إلغاء الحجز.

|  |  |
| --- | --- |
| المستخدم | نظام حجز المطاعم و الفنادق |
| 1. يختار المستخدم حجزاً من سجله ويطلب إلغاءه. |  |
|  | 2. يطلب النظام من المستخدم تأكيد إلغاء الحجز. |
| 3. يقوم المستخدم بتأكيد الإلغاء. |  |
|  | 4. يقوم النظام بإلغاء الحجز وتحديث حالته. |

**المسارات البديلة:**

* لا يوجد.

**مسارات الأخطاء:**

* إذا كان الحجز غير قابل للإلغاء بسبب تجاوز المهلة المسموحة، يعرض النظام ذلك.

### **3.2.4. حالات استخدام مدير النّظام:**

#### **1.3.2.4. تسجيل الدخول:**

الجدول لحالة تسجيل الدخول.

|  |  |
| --- | --- |
| اسم الحالة: تسجيل الدخول. | |
| الوصف Description: | يقوم مدير النظام بالولوج إلى لوحة التحكم الخاصة به باستخدام بيانات اعتماده. هذه الحالة هي شرط أساسي لتنفيذ جميع الوظائف الإدارية الأخرى. |
| الفاعلون Actors: | مدير النظام. |
| الشّروط السّابقة Preconditions: | أن يمتلك المدير حساباً مسبقاً بصلاحيات إدارية. |
| الشّروط اللاحقة Postconditions: | تم تسجيل دخول مدير النظام بنجاح إلى لوحة التحكم الإدارية. |

**سير الأحداث:**

**السيناريو الناجح الأساسي:**

الجدول السيناريو الناجح لحالة تسجيل الدخول.

|  |  |
| --- | --- |
| مدير النظام | نظام حجز المطاعم و الفنادق |
|  | 1. يطلب النظام اسم المستخدم و كلمة المرور. |
| 2. يقوم بإدخال اسم المستخدم وكلمة المرور الخاصة به. |  |
|  | 3. يقوم النظام بالتحقق من صحة البيانات وصلاحيات الحساب. |
|  | 4. يسمح النظام بالدخول. |

**المسارات البديلة:**

* لا يوجد.

**مسارات الأخطاء:**

* في حال كانت بيانات الاعتماد غير صحيحة، يعرض النظام ذلك.

#### **2.3.2.4. إضافة (مطعم/فندق/حدث):**

الجدول لحالة إضافة (مطعم/فندق/حدث).

|  |  |
| --- | --- |
| اسم الحالة: إضافة مطعم أو فندق أو حدث جديد. | |
| الوصف Description: | تسمح لمدير النظام بإضافة مطعم أو فندق أو حدث جديد إلى النظام مع كافة البيانات الخاصة به. |
| الفاعلون Actors: | مدير النظام. |
| الشّروط السّابقة Preconditions: | يجب أن يكون مدير النظام قد قام بتسجيل الدخول. |
| الشّروط اللاحقة Postconditions: | تم إنشاء سجل جديد للمطعم أو الفندق أو حدث في قاعدة بيانات النظام. |

**سير الأحداث:**

**السيناريو الناجح الأساسي:**

الجدول السيناريو الناجح لحالة إضافة (مطعم/فندق/حدث).

|  |  |
| --- | --- |
| مدير النظام | نظام حجز المطاعم و الفنادق |
| 1. يطلب إضافة مطعم/فندق/حدث جديد. |  |
|  | 2. يطلب إدخال بيانات المطعم/الفندق/الحدث. |
| 3. يقوم بإدخال البيانات المطلوبة. |  |
|  | 4. يقوم النظام بالتحقق من صحة البيانات و حفظها. |

**المسارات البديلة:**

* لا يوجد.

**مسارات الأخطاء:**

* في حال وجود بيانات ناقصة أو غير صالحة، يعرض النظام ذلك.

#### **3.3.2.4. تعديل بيانات (مطعم/فندق/حدث):**

الجدول لحالة تعديل بيانات (مطعم/فندق/حدث).

|  |  |
| --- | --- |
| اسم الحالة: تعديل بيانات غرض. | |
| الوصف Description: | يسمح لمدير النظام بتعديل بيانات مطعم أو فندق أو حدث موجود مسبقاً في النظام. |
| الفاعلون Actors: | مدير النظام. |
| الشّروط السّابقة Preconditions: | يجب أن يكون مدير النظام قد قام بتسجيل الدخول. |
| الشّروط اللاحقة Postconditions: | تم تحديث بيانات العنصر المحدد في النظام. |

**سير الأحداث:**

**السيناريو الناجح الأساسي:**

الجدول السيناريو الناجح لحالة تعديل بيانات (مطعم/فندق/حدث).

|  |  |
| --- | --- |
| مدير النظام | نظام حجز المطاعم و الفنادق |
| 1. يختار عنصراً (مطاعم/فنادق/حدث) لتعديله. |  |
|  | 2. يعرض البيانات الحالية للعنصر و يطلب المعلومات الجديدة. |
| 3. يقوم بإجراء التعديلات المطلوبة و يطلب حفظها. |  |
|  | 4. يقوم النظام بحفظ التغييرات وتحديث بيانات العنصر. |

**المسارات البديلة:**

* لا يوجد.

**مسارات الأخطاء:**

* في حال إدخال بيانات غير صالحة، يعرض النظام ذلك.

#### **4.3.2.4. حذف غرض(مطعم/فندق/حدث):**

الجدول لحالة حذف غرض (مطعم/فندق/حدث).

|  |  |
| --- | --- |
| اسم الحالة: حذف غرض. | |
| الوصف Description: | يسمح لمدير النظام بحذف مطعم أو فندق أو حدث من النظام بشكل نهائي. |
| الفاعلون Actors: | مدير النظام. |
| الشّروط السّابقة Preconditions: | يجب أن يكون مدير النظام قد قام بتسجيل الدخول. |
| الشّروط اللاحقة Postconditions: | تم حذف العنصر المحدد من قاعدة بيانات النظام. |

**سير الأحداث:**

**السيناريو الناجح الأساسي:**

الجدول السيناريو الناجح لحالة حذف غرض (مطعم/فندق/حدث).

|  |  |
| --- | --- |
| مدير النظام | نظام حجز المطاعم و الفنادق |
| 1. يستعرض العناصر (مطاعم/فنادق/أحداث) و يختار عنصراً لحذفه. |  |
|  | 2. يعرض النظام رسالة لتأكيد عملية الحذف. |
| 3. يؤكد رغبته في الحذف. |  |
|  | 4. يقوم النظام بحذف العنصر. |

**المسارات البديلة:**

* يمكن لمدير النظام إلغاء عملية الحذف قبل تأكيدها.

**مسارات الأخطاء:**

* لا يوجد.

#### **5.3.2.4. إضافة مدير نظام آخر:**

الجدول لحالة إضافة مدير نظام آخر.

|  |  |
| --- | --- |
| اسم الحالة: إضافة مدير نظام آخر. | |
| الوصف Description: | يسمح لمدير النظام الحالي بإنشاء حساب جديد لمدير نظام آخر ومنحه الصلاحيات اللازمة. |
| الفاعلون Actors: | مدير النظام. |
| الشّروط السّابقة Preconditions: | يجب أن يكون مدير النظام قد قام بتسجيل الدخول. |
| الشّروط اللاحقة Postconditions: | تم إنشاء حساب مدير نظام جديد في النظام. |

**سير الأحداث:**

**السيناريو الناجح الأساسي:**

الجدول السيناريو الناجح لحالة إضافة مدير نظام آخر.

|  |  |
| --- | --- |
| مدير النظام | نظام حجز المطاعم و الفنادق |
| 1. يطلب إضافة مدير نظام جديد. |  |
|  | 2. يطلب النظام إدخال بيانات المدير الجديد. |
| 3. يقوم بإدخال البيانات المطلوبة يطلب حفظها. |  |
|  | 4. يقوم النظام بإنشاء الحساب الجديد وإضافته إلى قائمة مدراء النظام. |

**المسارات البديلة:**

* لا يوجد.

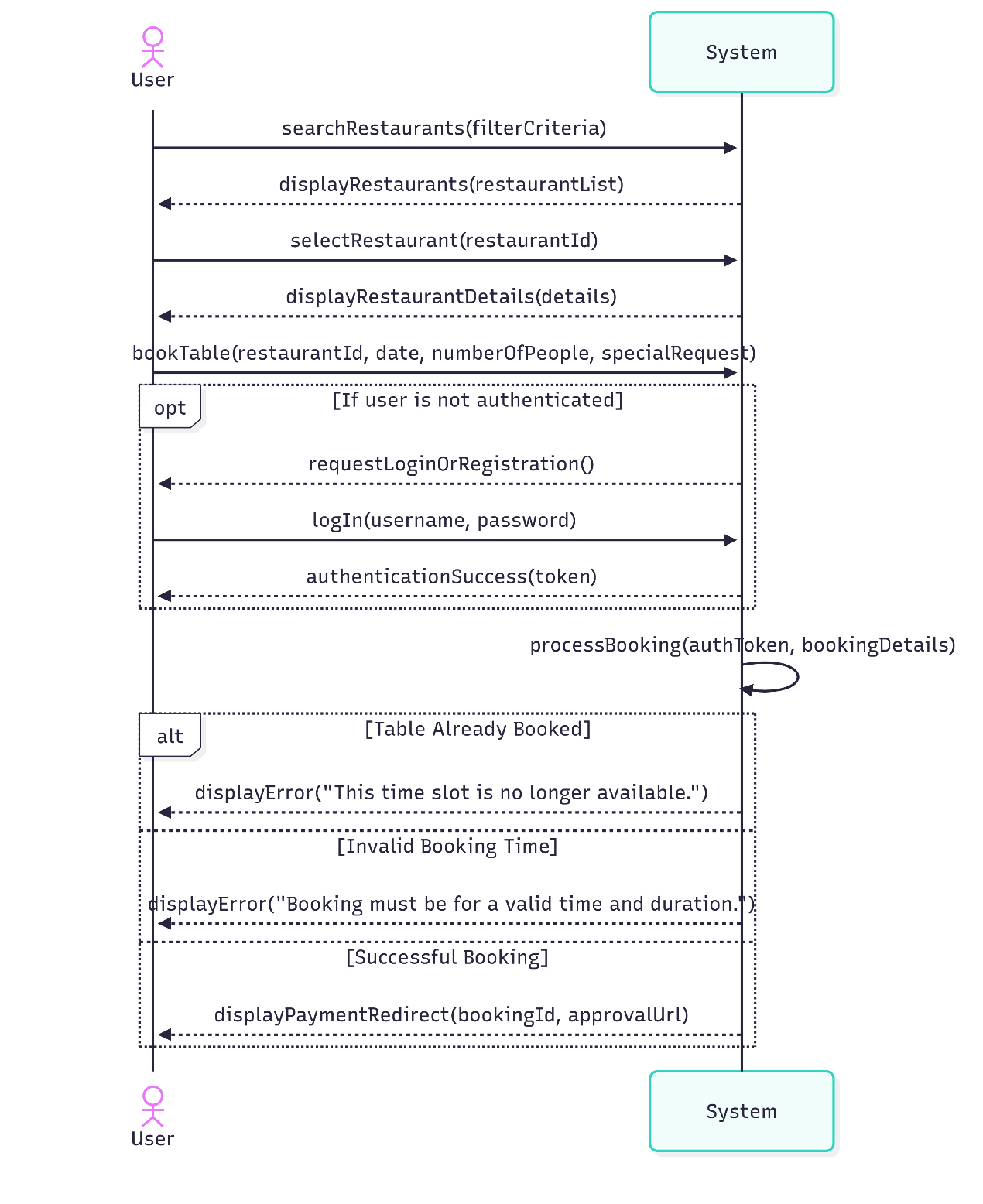
**مسارات الأخطاء:**

* إذا كان اسم المستخدم أو البريد الإلكتروني مستخدماً بالفعل، يعرض النظام ذلك.

## **2.5. مخططات تسلسل النظام:**

نورد في هذا القسم مخططات تسلسل النّظام لحالات الاستخدام الأساسية في النّظام.

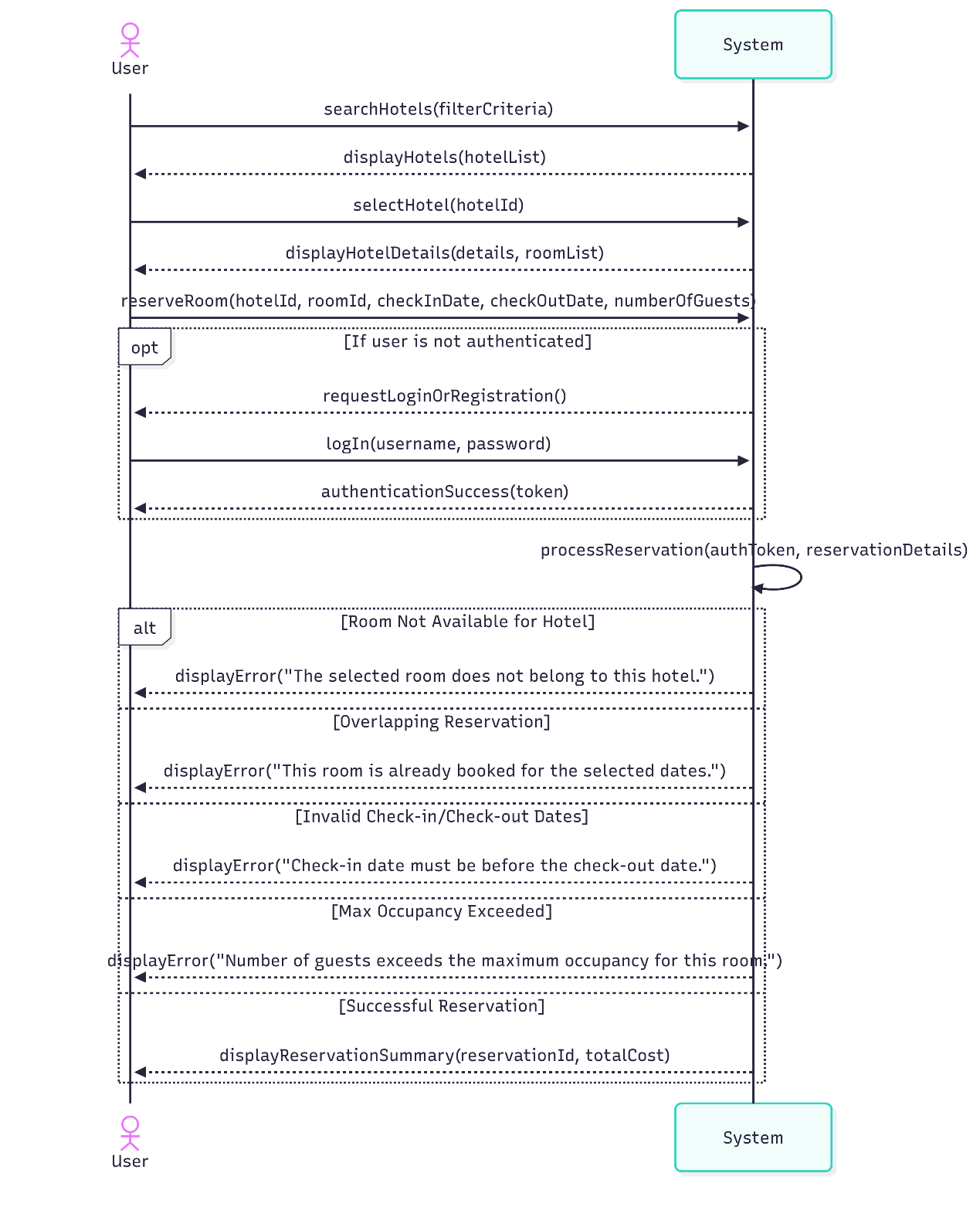
### **1.2.5. الحجز لدى مطعم:**



الشكل :

شكل يوضح مخطط تسلسل النظام لحالة استخدام الحجز لدى مطعم.

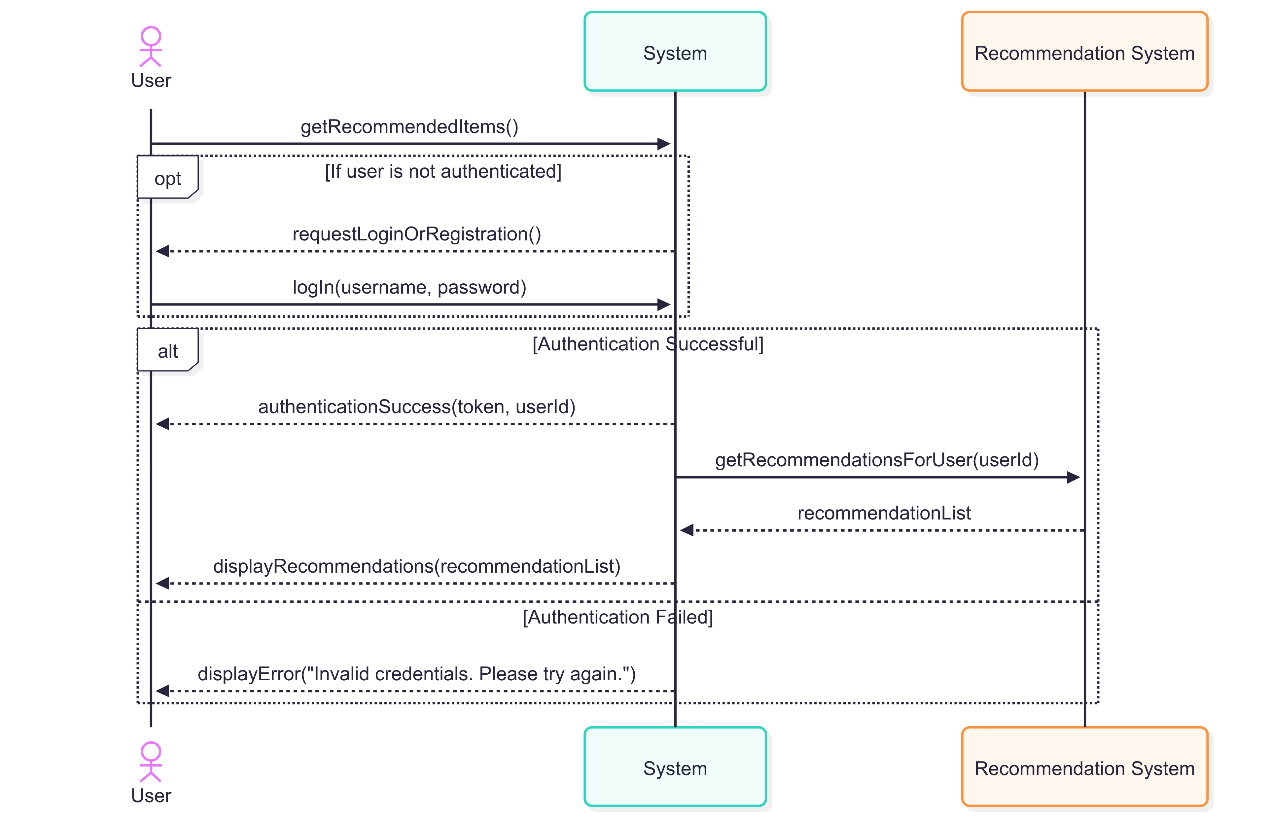
### **2.2.5. حجز غرفة في فندق:**



الشكل :

شكل يوضح مخطط تسلسل النظام لحالة استخدام حجز غرفة في فندق.

### **3.2.5. استعراض الأغراض الموصى بها:**



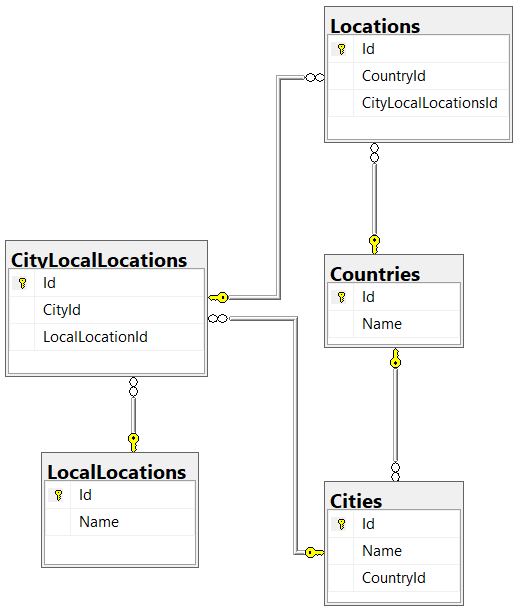
الشكل :

شكل يوضح مخطط تسلسل النظام لحالة استخدام استعراض الأغراض الموصى بها.

## **3.5. مخطط كائنات العلاقات ERD في النظام:**

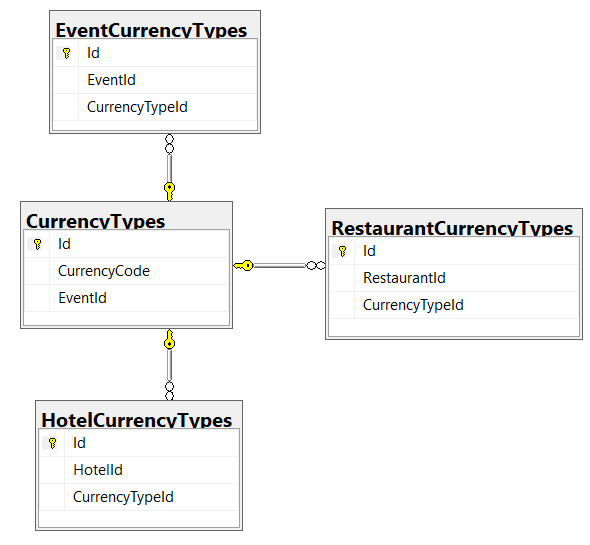
فيما يلي نوضح ست مخططات كل منها يخص مجال معين وهي المطاعم، الفنادق و الأحداث و أنواع العمولة و المواقع و المخطط الأخير يوضح جدول المستخدمين و الجداول التي يرتبط بها مع باقي المجالات.

### **1.3.5. مخطط المواقع:**

****

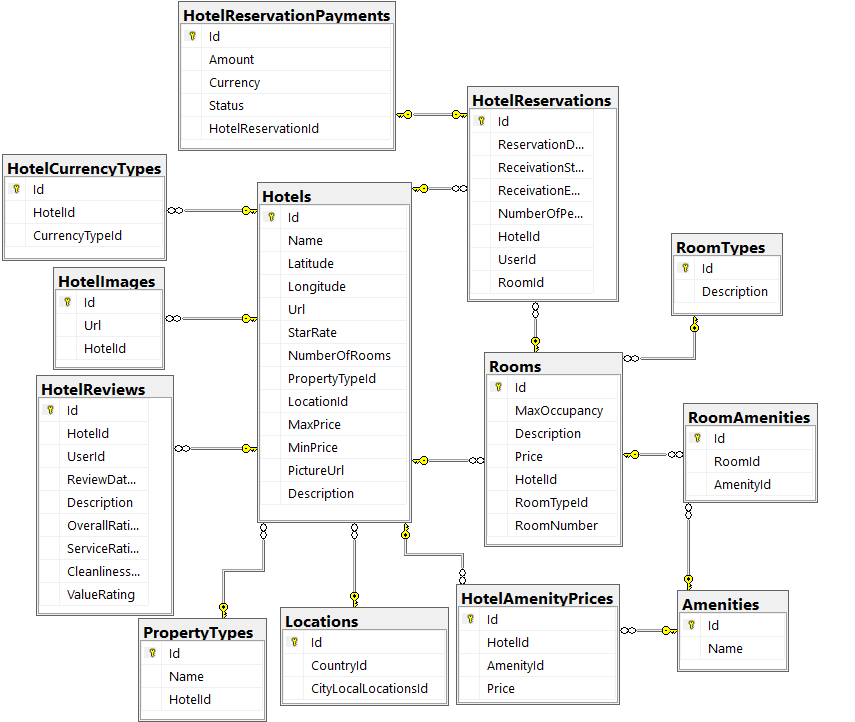
الشكل : شكل يوضح جدول المواقع و الجداو المرتبطة به.

### **2.3.5. مخطط أنواع العمولة:**

****

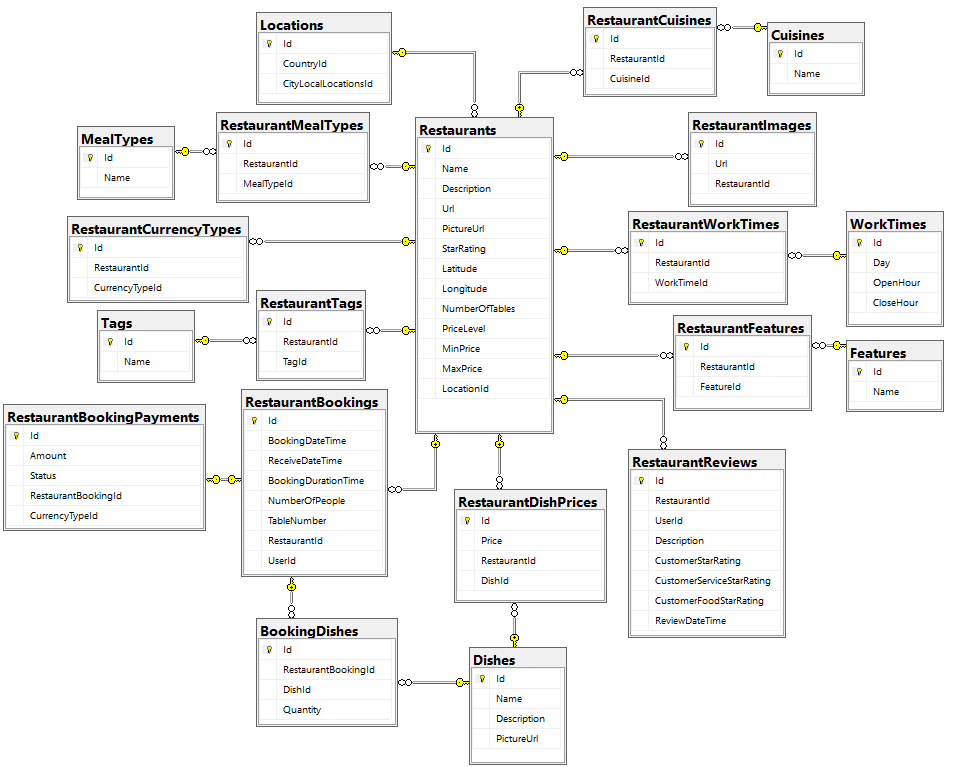
الشكل : شكل يوضح جدول أنواع العمولة و الجداو المرتبطة به.

### **3.3.5. مخطط الفنادق:**

****

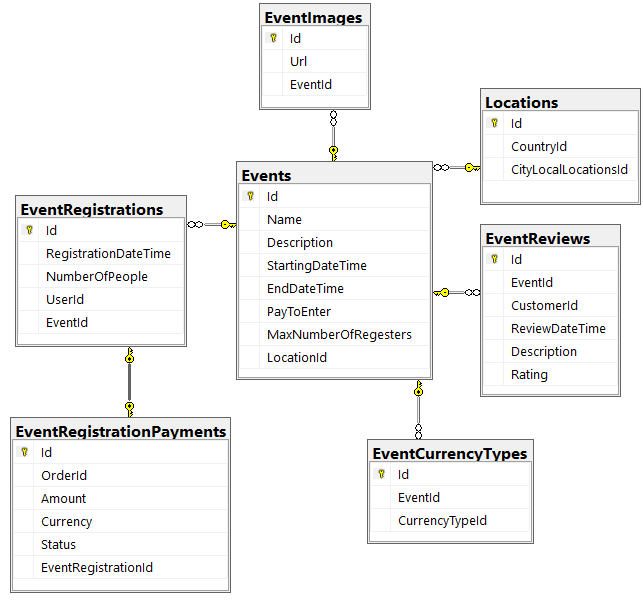
الشكل : شكل يوضح جدول الفنادق و الجداو المرتبطة به.

### **4.3.5. مخطط المطاعم:**

****

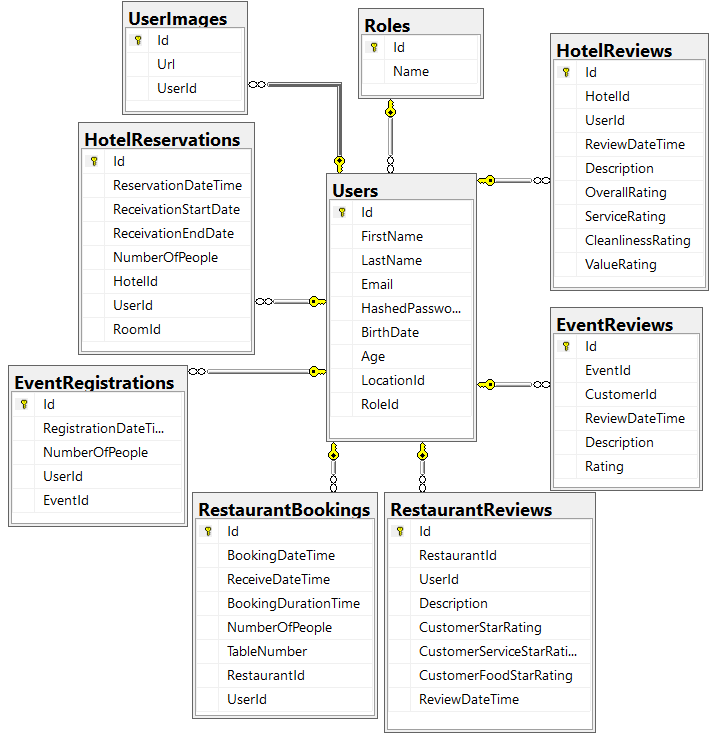
الشكل : شكل يوضح جدول المطاعم و الجداو المرتبطة به.

### **5.3.5. مخطط الأحداث:**

****

الشكل : شكل يوضح جدول الأحداث و الجداو المرتبطة به.

### **6.3.5. مخطط المستخدمين و ارتباطاته مع باقي المخططات:**

****

الشكل : شكل يوضح جدول المستخدمين و الجداو المرتبطة به.

# الفصل الخامس

**بيئة العمل و الدراسة التصميمية**

*سنورد في هذا التصاميم المعتمدة في بناء النظام، اضافة ً الى شرح موجز عن الأدوات المُستعملة في تنجيز هذا* *النظام*

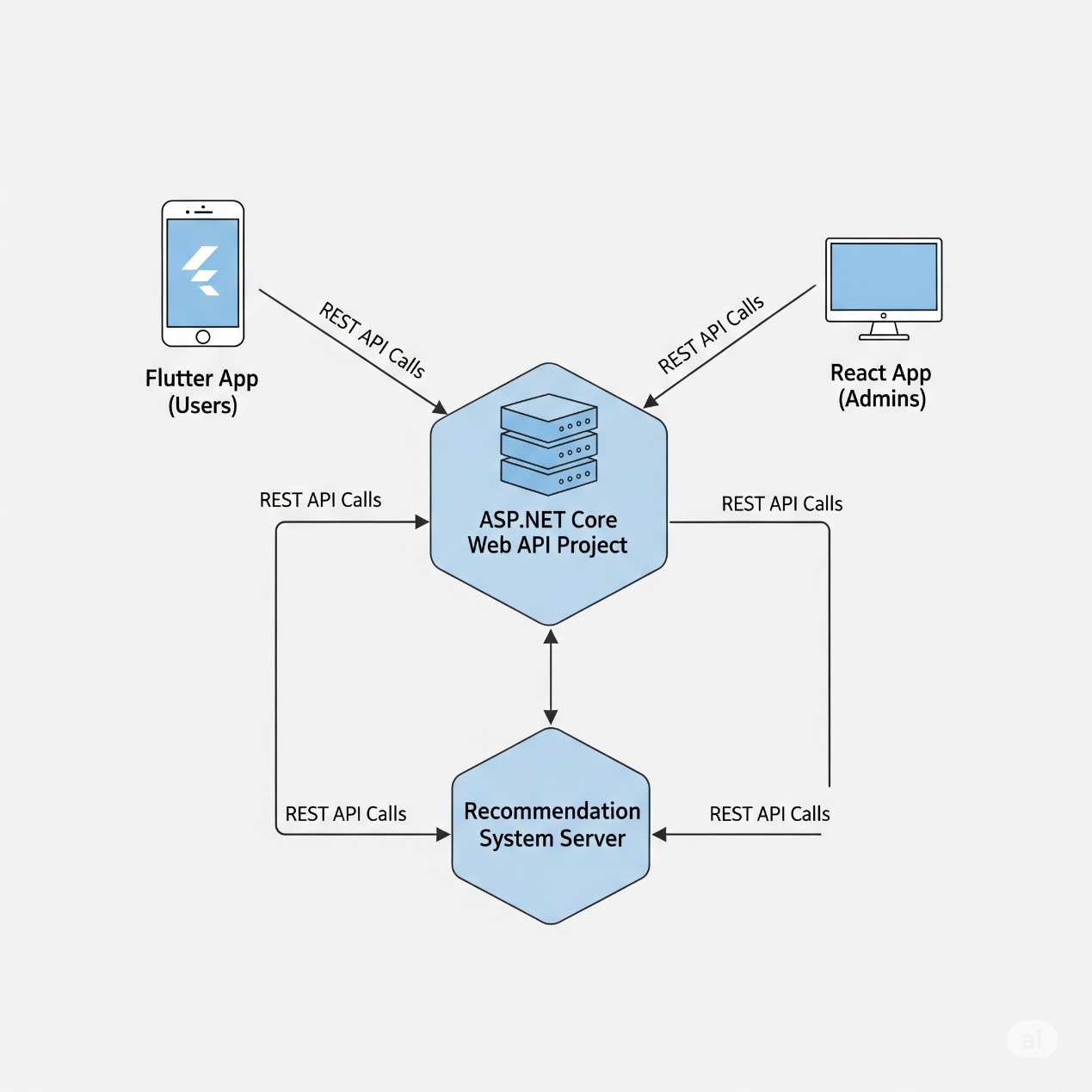
## **1.6. البنية المقترحة:**

إنّ اعتماد منهجيّة واضحة لبناء النظام منذ البداية يساعد على إيجاد رماز مصدري (codebase) قابل للاختبار والتعديل بمرور الوقت. تم الاعتماد على البنية المعمارية النظيفة (Clean Architecture) في الواجهة الخلفية، إذ تعطي إمكانية فصل قواعد العمل وحالات الاستخدام عن الخيارات التشغيلية، مما يسمح باستبدالها عند الحاجة. يضمن هذا النهج أن النظام مستقل عن أطر العمل وقواعد البيانات والواجهات الأمامية.

يتكون النظام من ثلاثة مكونات رئيسية:

* 1. التطبيق الخلفي Backend: مبني باستخدام ASP.NET Core Web API، وهو مسؤول عن إدارة بيانات الفنادق والمطاعم والحجوزات والمستخدمين.
  2. محرك التوصية Recommendation Engine: وهو نظام تعلم عميق منفصل، تم بناؤه باستخدام Python و Torch/nn، وهو مسؤول عن توليد توصيات مخصصة للمستخدمين.
  3. الواجهات الأمامية Frontend: واجهتان منفصلتان، واحدة للمستخدمين Flutter وأخرى لمدير النظام React، تتفاعلان مع كل من التطبيق الخلفي ومحرك التوصية.

يوضح الشكل التالي الشكل المختار لمعمارية النظام:



الشكل : شكل يوضح أجزاء النظام.

## **2.6. تنجيز التطبيق الخلفي Backend:**

### **1.2.6. إطار العمل:**

بالنسبة للتطبيق الخلفي، فقد تم إنشاء المشروع ضمن إطار عمل. ASP.NET Core Web API حيث تم اختيار هذا الإطار لكونه يدعم الأنظمة الأساسية، ويتميز بأدائه العالي، ويتكامل بشكل جيد مع التقنيات الأخرى المستخدمة في هذا المشروع. يوفر ASP.NET Core Web API بنية قوية لبناء خدمات HTTP التي يمكن استهلاكها من قبل العديد من العملاء، مثل تطبيقات الويب والهواتف المحمولة.

### **2.2.6. هيكلية الطبقات وتدفق العمليات:**

لتنظيم هيكلية الواجهة الخلفية، تم تقسيم المشروع إلى طبقات منفصلة لتعزيز مبدأ فصل الاهتمامات Separation of Concerns وجعل النظام أكثر نمطية وقابلية للصيانة. تم اعتماد الأنماط التصميمية التالية:

* نمط المستودع Repository Pattern: تُستخدم لإنشاء طبقة تجريد بين طبقة الوصول إلى البيانات وطبقة منطق الأعمال. هذا يسمح بتغيير آلية تخزين البيانات دون التأثير على بقية أجزاء النظام.
* نمط الوسيط Mediator Pattern مع مكتبة MediatR :الهدف الأساسي من استخدامه هو تقليل التبعيات الفوضوية بين الكائنات، حيث يتم التواصل فيما بينها عن طريق كائن وسيط. تقوم مكتبة MediatR بتنجيز هذا النمط، حيث يتم فصل إرسال الطلبات Requests عن معالجاتها Handlers.
* نمط النّتيجة Result Pattern:

نمط النّتيجة هو نمط تصميم يستخدم لتغليف نتائج عمليّة، بما في ذلك حالات النّجاح أو الفشل و أي بيانات أو أخطاء مرتبطة بها. يتم استخدام هذا النّمط بشكل شائع في البنية المعماريّة النّظيفة وتطبيقات المبنية باستخدام النّمط المعماريّ CQRS لتوفير طريقة منظمة للتعامل مع نتائج العمليّات. يتيح نمط النتيجة إمكانية إرجاع كائن منظم بدلاً من طرح استثناءات Exceptions عند التّحقق من صحّة قواعد العمل [16].

يشير الكائن الذي يتم إرجاعه إلى:

* حالة النجاح أو الفشل للدلالة ما إذا تمّت العملية بنجاح أو ظهرت مشكلة أثناء تنفيذها.
* الغرض الذي أنتجته العمليّة إذا تمّت بنجاح.
* الخطأ الذي واجهته العمليّة إذا انتهت بالفشل.
* نمط فصل مسؤولية الاستعلام والأوامر: CQRS ظهر هذا النمط انطلاقاً من نمط CQS، حيث يتم فصل مسؤوليات النظام لأوامر Commandsو استعلامات Queriesعلى المستوى المعماري. يساعد هذا النمط في تحسين الأداء والمرونة وإدارة التعقيد ضمن النظام.

## **3.6. محرك التوصية Recommendation Engine:**

لتقديم توصيات مخصصة ودقيقة، تم بناء نموذج تعلم عميق يعتمد على نهج هجين يجمع بين التصفية التعاونية Collaborative Filtering والتصفية القائمة على المحتوى Content-Based Filtering.

### **1.3.6. بنية النموذج**

تم بناء النموذج باستخدام مكتبتي Tourch و nn و هو يتكون من المكونات التالية:

* مدخلات النموذج: يستقبل النموذج مُعرّفات المستخدمين User IDsومُعرّفات العناصر (الفنادق/المطاعم).
* طبقات التضمين Embedding Layers : يتم تحويل كل من مُعرّف المستخدم والعنصر إلى متجه رقمي كثيف Embedding Vector هذه الطبقة تسمح للنموذج بتعلم تمثيلات كامن latent features لكل من المستخدمين والعناصر.
* دمج المتجهات: يتم دمج متجهات التضمين الخاصة بالمستخدم والعنصر باستخدام طبقة ضرب نقطي Dot Product ، و التي تحسب مدى التشابه بين تفضيلات المستخدم وخصائص العنصر.
* طبقة الخرج: طبقة خرج نهائية تقوم بتوليد التقييم المتوقع Predicted Rating.

### **2.3.6. التدريب والتقييم:**

* دالة الخسارة: تم استخدام دالة متوسط الخطأ التربيعي Mean Squared Errorلقياس الفرق بين التقييمات المتوقعة والفعلية.
* المُحسِّن Optimizer :تم استخدام مُحسِّن Adam لتحديث أوزان الشبكة بكفاءة أثناء التدريب.
* التقييم: تم تقييم أداء النموذج باستخدام مقياس NDCG على مجموعة بيانات الاختبار.

### **3.3.6. واجهة برمجة التطبيقات API:**

تم عرض النموذج المدرب كخدمة ويب باستخدام واجهة برمجة تطبيقات APIمبنية بلغة Python. هذه الواجهة مسؤولة عن استقبال طلبات التوصية من التطبيق الخلفي الرئيسي، ومعالجتها باستخدام النموذج المحفوظ، وإعادة قائمة بالعناصر الموصى بها.

## **4.6. تنجيز الواجهات الأمامية Frontend:**

### **1.4.6. منهجية العمل:**

تم بناء واجهتين منفصلتين للتفاعل مع النظام:

#### **1.1.4.6. واجهة المستخدم (الزبون):**

تم استخدام إطار العمل Flutter لبناء تطبيق جوال يعمل على منصتي Android و iOS. تم اعتماد بنية معمارية واضحة تفصل بين الاهتمامات المختلفة للتطبيق، وهي نمط MVVM (Model-View-ViewModel). حيث تتكون هيكلية التطبيق من المجلدات التالية:

1. models: تحتوي على نماذج البيانات التي تمثل الكيانات في النظام (مثل الفنادق والمطاعم).
2. api / services: طبقة مسؤولة عن الاتصال بالواجهة الخلفية Backend.
3. ui/viewmodels: تدير حالة التطبيق State Management وتحتوي على منطق الأعمال الخاص بالواجهة. تم استخدام حزمة Provider لتوفير الحالة والاعتماديات بشكل فعال عبر التطبيق.
4. ui/views: تحتوي على واجهات المستخدم المختلفة التي يتفاعل معها المستخدم.

#### **2.1.4.6. واجهة مدير النظام:**

تم استخدام مكتبة React. و هي مكتبة JavaScript مفتوحة المصدر تستخدم لبناء واجهات مستخدم ديناميكية وتفاعلية. تتبع React البنية القائمة على المكونات Components، مما يساعد على بناء مكونات واجهة مستخدم قابلة لإعادة الاستخدام.

**المراجع:**

[1] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*

[2] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press

[3] Kingma, D. P., & Ba, J. (2014). *Adam: A Method for Stochastic Optimization*. arXiv preprint arXiv:1412.6980

[4] Järvelin, K., & Kekäläinen, J. (2002). Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*

[5] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014)

[6] Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix Factorization Techniques for Recommender Systems. *Computer*

[7] Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press

[8] Ricci, F., Rokach, L., & Shapira, B. (Eds.). (2015). *Recommender Systems Handbook*. Springer

[9] Martin, R. C. (2017). Clean architecture

[10] Design Patterns: Elements of Reusable Object-Oriented Software

[11] Korkmaz, N., & Nilsson, M. (2014). Practitioners’ view on command query responsibility segregation. M Sc Thesis

[12] https://business.yelp.com/data/resources/open-dataset

[13] DB-MRec: A Deep learning-based Recommender System for Yelp Dataset

[14] A-Seof: An Attention-based Semantics-aware Online Feature-making for Restaurant Recommendation

[15] BERT-Based Hybrid Collaborative Filtering for Recommender Systems

[16] Patterns of Enterprise Application Architecture

1. MediatR:
2. MediatR هي مكتبة .NET يتُنجّز نمط الوسيط مما يتيح فصل المكونات في التطبيق أثناء المراسلة ضمن العمليات. تُساعد هذه المكتبة على تبسط الاتصال بين الكائنات من خلال تقديم وسيط يتعامل مع الطلبات والإشعارات. تحتوي المكتبة على نوعين من الرّسائل التي ترسلها:

* رسائل الطلب/الاستجابة، يتم إرسالها إلى معالج Handler واحد. هناك نوعان من الطلبات تلك التي ترجع قيمة، وتلك التي لا تفعل ذلك:
  + IRequest<TResponse> هي الطّلبات التي ترجع قيمة.
  + IRequest هي الطّلبات التي لا ترجع قيمة.
* كل طلب يوجد له معالج واحد حسب نوعه:
  + IRequestHandler<TRequest, TResponse> معالج في حالة كان الطّلب ييرجع قيمة.
  + IRequestHandler<TRequest> معالج في حال كان الطّلب لا يرجع قيمة.
* رسائل الإشعارات، يتم إرسالها إلى معالجات متعددة. هنا يوجد مرسل واحد وعدة معالجات كل معالج يستقبل الطّلب و يعالجه حسب الغاية المرادة منه كما يلي:
  + INotification هو المرسل الذي يرسل الطلبات.
  + INotificationHandler هو المعالج الذي يستقبل الطلبات و يمكن أن يتكرر. كل معالج ينفّذ المنطق الخاص به.

1. يسمح للمستخدم باستعراض أهم الأحداث و الأماكن السياحية في منطقة معينة يختارها المستخدم.
2. يسمح للمستخدم باستعراض المطاعم و الفنادق في منطقة معينة يختارها المستخدم.
3. يقوم بعمل دليل سياحي لمنطقة ما يختارها المستخدم.
4. نمط الوسيط Mediator Pattern :

نمط الوسيط هو عبارة عن نمط تصميمي سلوكي الهدف الأساسي من استخدامه هو تقليل التبعيات الفوضويّة بين الكائنات حيث يتم التواصل فيما بينها عن طريق غرض من نمط وسيط [Evans, E. (2004). Domain-driven design: tackling complexity in the heart of software. Addison-Wesley Professional.].

كمثال على أهمية التواصل مع وسيط لتقليل التبعيات بين الكائنات، نذكر هبوط الطائرات ضمن مدرج المطار، حيث أن ّلطيار عند اقترابه من مدرج الهبوط يتواصل فقط مع برج المراقبة -الذي يعّبر عن الوسيط في مثالنا- الذي تتلخص مهمته بتنظيم مواعيد الهبوط للطائرات المختلفة فقط، من خلال هذه المنهجية لن يحتاج الطيار لأن يقوم بالتواصل مع أي طيار آخر، مما يجعله مركزا على مهمته الأساسيّة وهي الهبوط بالطّائرة.

لابد من الإشارة إلى أن إحدى مساوئ استخدام نمط الوسيط هي أن الغرض الذي يلعب دور الوسيط من الممكن أن يتحول إلى غرض مرتبط بالعديد من الأغراض الأخرى، ممّا يجعل صيانته أمرا صعبا أحيانا.

1. نمط المستودع Repository Pattern:
2. هو نمط تصميمي يستخدم لإدارة تخزين واسترجاع كائنات المجال، يعتمد على تجريد التعامل مع آلية تخزين البيانات مما يحمي بقية أجزاء النظام من تعقيدات التخزين والاسترجاع، ويقلل الاعتماديات ضمن النظام. تترتب على عائق المستودعات مسؤوليّة تمثيل البيانات ضمن المخزن سواءكان المخزن قاعدة معطيات أو نظام ملفات أو غيره، ثم إعادة استرجاعها وتحويلها لكائنات المجال المناسبة.
3. البنية المعمارية Model-View-ViewModel (MVVM):

البنية المعمارية MVVM هي نمط تصميم يفصل واجهة المستخدم (UI) الخاصة بالتطبيق عن منطق الأعمال والبيانات الخاصة به. قبل ظهور أنماط مثل MVVM، كان المطورون غالبًا ما يكتبون منطق العمل مباشرةً داخل معالجات أحداث واجهة المستخدم، مثل وضع استدعاء لقاعدة البيانات داخل دالة النقر على زر. أدى هذا إلى مشكلة تُعرف بالاقتران الوثيق (Tight Coupling)، حيث يتشابك كل من واجهة المستخدم والمنطق لدرجة أن التطبيق يصبح [https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm]:

* **صعب الاختبار**، حيث لا يمكننا التحقق من منطق العمل دون تشغيل واجهة المستخدم الفعلية، وهي عملية بطيئة ومعقدة .
* **صعب الصيانة**، لأن تغييرًا بسيطًا في واجهة المستخدم قد يؤدي بسهولة إلى كسر منطق العمل، والعكس صحيح. غالبًا ما يؤدي هذا إلى ما يسمى بـكود غير منظم spaghetti code.
* **يمثل تحديًا للتعاون**، فلم يتمكن مصممو واجهة المستخدم من العمل على التصميم المرئي دون التسبب في تعارض محتمل مع المطورين الذين يكتبون المنطق.

تعمل بنية MVVM على حل هذه المشكلات عن طريق إنشاء حدود واضحة بين هذه الأجزاء المختلفة من التطبيق. فهي تتكون من ثلاث طبقات مميزة تعمل معًا لإنشاء تطبيق منظم جيدًا.

* + النموذج Model:

يمثل النموذج بيانات التطبيق ومنطق العمل الأساسي. و هو المسؤول عن إدارة البيانات وإجراء الحسابات وتطبيق قواعد العمل. على سبيل المثال، في تطبيق للتجارة الإلكترونية، يحدد النموذج ماهية المنتج، ويتعامل مع إضافته إلى سلة التسوق، ويحسب السعر الإجمالي. الميزة الرئيسية للنموذج هي أنه مستقل تمامًا عن واجهة المستخدم، ويمكن استخدامه في تطبيق ويب أو جوال أو سطح مكتب دون أي تغييرات.

* + المنظر View:

يمثل المنظر (View) واجهة المستخدم أي كل ما يراه المستخدم ويتفاعل معه، مثل الأزرار ومربعات النص والتخطيطات. وظيفته الوحيدة هي عرض البيانات التي يتلقاها وتمرير إجراءات المستخدم (مثل النقرات) إلى ViewModel. المبدأ الأساسي في MVVM هو أن المنظر يجب أن يكون بسيطًا ومخصصًا للعرض فقط، بمعنى أنه لا يحتوي على أي منطق عمل. في الأطر الحديثة مثل Flutter أو .NET MAUI، غالبًا ما يتم تعريف المنظر بشكل تصريحي باستخدام الكود.

* + عرض النموذج ViewModel:

يعمل نموذج العرض (ViewModel) كوسيط بين النموذج والمنظر. له مسؤوليتان رئيسيتان:

* + - تجهيز البيانات للعرض: يأخذ البيانات الخام من النموذج ويهيئها لتكون جاهزة للعرض في المنظر. على سبيل المثال، قد يحول كائن تاريخ ووقت إلى نص سهل القراءة مثل "26 يوليو 2025".
    - معالجة إجراءات المستخدم: يقدم الأوامر أو الدوال التي يمكن للمنظر استدعاءها عندما يقوم المستخدم بإجراء ما. على سبيل المثال، عند النقر على زر "حفظ" في المنظر، فإنه يستدعي أمر الحفظ الموجود في ViewModel، الذي يقوم بدوره بتنفيذ المنطق اللازم، غالبًا بالتفاعل مع النموذج.

يكمن سر قوة ViewModel في أنه لا يعرف أي شيء عن عناصر واجهة المستخدم المحددة في المنظر، مثل "الزر" أو "مربع النص". بدلًا من ذلك، يتواصل مع المنظر عبر آلية تسمى ربط البيانات (Data Binding)، حيث ترتبط خصائص ViewModel مباشرة بعناصر المنظر. عند تغيير أي خاصية في ViewModel، يتم تحديث واجهة المستخدم تلقائيًا لتعكس هذا التغيير.