**الجمهوريـة العربيـة الس ورية**
**المعهد العالي للعلوم التطبيقية والتكنولوجيا قسم المعلوميات**
**العام الدراسي 2025/2024**

**مشروع سنة رابعة**
هندسة البرمجيات

# نظام إلكتروني لإدارة تذاكر الصيانة في المعهد العالي

Maintenance Ticket Management System In Hiast

تقديم الطالب

**أحمد أمين الكريان**

إشراف

ما.محمود **إلياس**

**ما. إياد سبيع**

2/8/2025

# كلمة شكر

كل الشكر والإمتنان لكل من **المهندس محمود إلياس** للمساعدة على إنجاز هذا العمل، وتوجيهاته الق ي مة و نصائحه المستمرة و مرافقته لنا خطوة بخطوة في هذا العمل، و **المهندس إياد سبيع** لملاحظاته المفيدة و دعمه الدائم والمستمر.

**أحمد أمين الكريان**

# الملخص

يُعد تبسيط عمليات الصيانة في المؤسسات التعليمية وتحويلها من إجراءات ورقية معقدة إلى نظام إلكتروني متكامل أمراً بالغ الأهمية؛ لضمان سرعة الاستجابة، وتتبع الطلبات، ورفع كفاءة فريق الدعم الفني. انطلاقاً من هذه الحاجة، قمنا بتصميم نظام ويب متكامل لإدارة تذاكر الصيانة في المعهد العالي يعمل بتكامل سلس مع نظام المصادقة الخاص بالمعهد ، كما يضمن آلية إشعارات ذكية تُعلم مسؤولي الصيانة فوراً بالطلبات الجديدة، وتُبلغ المستخدمين بشكل تلقائي عند أي تحديث لحالة تذكرتهم .لضمان تجربة مستخدم مخصصة،: فالموظفون يتابعون ، بُني النظام ليكون سريعاً وآمناً وسهل الاستخدام، وحماية بيانات المستخدمين عبر تشفير متقدم، وتوافقه مع جميع الأجهزة من حواسيب وهواتف ذكية. يحقق هذا الحل تحولاً شاملاً في إجراءات الصيانة؛ وتقليل الوقت الضائع، وتعزيز الشفافية بين المستخدمين وفريق الدعم، مما ينعكس إيجاباً على جودة الخدمات المقدمة في المؤسسة التعليمية.

# Abstract

Simplifying maintenance processes in educational institutions by transforming them from complex paper-based procedures into an integrated electronic system is of utmost importance to ensure prompt response, efficient tracking of requests, and enhanced productivity for the technical support team. To address this need, we have designed a comprehensive web-based maintenance ticket management system for the Higher Institute that seamlessly integrates with the Institute's authentication system and supports attaching photos or documents to clarify issues, featuring an intelligent notification mechanism that immediately alerts maintenance supervisors about new requests and automatically informs users about any updates to their tickets. The system ensures a personalized user experience where employees can only track their own tickets, built to be fast, secure, and user-friendly with advanced encryption for data protection while maintaining compatibility with all devices including computers and smartphones, bringing comprehensive transformation in maintenance procedures by reducing wasted time and enhancing transparency between users and the support team for improved service quality in the educational institution.

# Contents

# جدول المحتويات

# قائمة الأشكال

قائمة الجداول

# مقدمة عامة

في ظل التطور التكنولوجي المتسارع، أصبحت أنظمة إدارة الصيانة الإلكترونية من الأدوات الأساسية التي تعزز كفاءة المؤسسات التعليمية والعلمية. يُعد مشروع "نظام تذاكر الصيانة" خطوة مهمة نحو أتمتة عمليات الصيانة في المعهد العالي،

حيث يهدف إلى تبسيط إدارة الطلبات وتحسين التواصل بين الموظفين وفرق الصيانة. يعتمد هذا النظام على مفاهيم هندسة البرمجيات وقواعد البيانات، مما يجعله مثالًا عمليًا لتطبيق المعرفة الأكاديمية في حل مشكلات واقعية.

يتميز النظام بتوفير واجهة ويب تفاعلية تتيح للموظفين إنشاء تذاكر صيانة جديدة بسهولة، مع إمكانية تتبع حالة الطلبات بشكل فوري. كما يدعم النظام التكامل مع أنظمة المصادقة الموجودة في المعهد مثل ، مما يضمن أمانًا وكفاءة في إدارة المستخدمين. بالإضافة إلى ذلك، يوفر النظام لوحات تحكم مخصصة لكل من الموظفين ومسؤولي الصيانة، مع إشعارات تلقائية لتحديثات الحالة، مما يعزز الشفافية ويقلل من الوقت المهدر في المتابعة اليدوية.

من خلال هذا المشروع، يتم تعزيز المهارات التقنية للطلاب في مجال تطوير تطبيقات الويب، وإدارة قواعد البيانات، وتطوير واجهات المستخدم، فضلًا عن فهم متطلبات الأمان والأداء في الأنظمة الإلكترونية. يُعتبر هذا النظام نموذجًا لتوظيف التكنولوجيا في تحسين العمليات الإدارية، مما يسهم في رفع جودة الخدمات المقدمة داخل المعهد.

الفصل الأول

# التعريف بالمشروع

سنتطرق في هذا الفصل الى توضيح الهدف من المشروع مع ذكر المتطلبات الوظيفية غير الوظيفية

# 1.1-الهدف من المشروع

يهدف مشروع "تذاكر صيانة" إلى أتمتة عمليات إدارة طلبات الصيانة في المعهد العالي عبر نظام مركزي وفعال لتسجيل ومتابعة التذاكر، بهدف تبسيط العملية من الإنشاء حتى الحل مع تقليل التدخل البشري والحد من الأخطاء. يعمل النظام على تحسين التواصل بين الموظفين وفرق الصيانة عبر إشعارات فورية عند إنشاء التذاكر أو تحديثها، مما يضمن معالجة سريعة ودقيقة للمشكلات، كما يعزز الشفافية عبر لوحات تحكم تتيح تتبع حالة الطلبات بسهولة. بالإضافة إلى ذلك، يسعى المشروع لتطبيق مفاهيم هندسة البرمجيات مثل بناء تطبيقات الويب المتكاملة وإدارة قواعد البيانات والاستفادة من أنظمة المصادقة مثل LDAP، مما يجعله خطوة مهمة نحو التحول الرقمي في المعهد وبيئة عمل أكثر كفاءة وتنظيمًا.

# 1.2- المتطلبات

تقسم المتطلبات الى متطلبات وظيفية واخرى غير وظيفية سنجدها بشكل مُف صل كما يلي:

## 1.2.1-المتطلبات

### الوظيفية

.يجب على النظام أن:

1. يسمح بتسجيل الدخول باستخدام بيانات LDAP الخاصة بالمعهد.
2. يعبئ تلقائياً بيانات المستخدم الأساسية (الاسم، القسم، البريد الإلكتروني) من نظام LDAP.
3. يوفر مستويين للصلاحيات:

- موظف عادي (يمكنه إنشاء ومتابعة تذاكره فقط).
- مسؤول الصيانة (يملك صلاحيات كاملة).

**.2إدارة تذاكر الصيانة**

- يجب على النظام أن:

.4يوفر واجهة لإنشاء تذاكر صيانة جديدة تحتوي على:

- حقل لوصف المشكلة (إجباري).

- قائمة منسدلة لأنواع الأجهزة (إجباري).
- حقل لمعرّف الجهاز (اختياري).
- إمكانية رفع ملفات مرفقة (صور/وثائق).

5. يولد أرقام تذاكر فريدة تلقائياً بالصيغة: [سنة]-[رقم تسلسلي]-[نوع الجهاز]) مثال: 2024-056).PC-
6. يسمح بالبحث عن التذاكر السابقة باستخدام:

- رقم التذكرة.
- تاريخ الإنشاء.
- حالة التذكرة.

**3.معالجة التذاكر**

- يجب على النظام أن:

7.يرسل إشعاراً فورياً لمسؤولي الصيانة عند إنشاء تذكرة جديدة.
8.يسمح لمسؤولي الصيانة بتغيير حالة التذاكر (جديد، قيد المعالجة، مكتمل، مرفوض).
9.يوفر إمكانية إضافة ملاحظات وتحديثات على التذكرة من قبل مسؤولي الصيانة.
10.يرسل إشعارات للموظفين عند تحديث حالة تذاكرهم.

**4.لوحات التحكم**

- يجب على النظام أن:

11.يعرض للموظفين لوحة تحكم شخصية تحتوي فقط على تذاكرهم.
12.يوفر لمسؤولي الصيانة لوحة تحكم شاملة تحتوي على:

- جميع التذاكر.
- إحصاءات وأرقام عن حالات التذاكر.
- أدوات تصفية متقدمة.

**المتطلبات غير الوظيفية**

**1.متطلبات الأداء**

- يجب على النظام أن:

13.يستجيب لأي عملية خلال أقل من ثانيتين.

14.يدعم 100 مستخدم متزامن على الأقل.
15.يحافظ على وقت تشغيل 99.9%.

- التوفر: يجب أن يتوفر استخدام النظام على مدار 24 ساعة يوميا و على مدار 7 أيام بالإسبوع، أي بشكل دائم.

**2.متطلبات الأمان**

- يجب على النظام أن:

16.يستخدم تشفير TLS 1.2+ لجميع الاتصالات.
17.ينفذ التحقق من الصلاحيات قبل كل عملية حساسة.
18.يخزن كلمات المرور باستخدام تقنيات التجزئة الآمنة.

**3.متطلبات التوافقية**

- يجب على النظام أن:

19.يعمل بشكل كامل على أحدث إصدارين من:

  - Google Chrome.
  - Mozilla Firefox.
  - Microsoft Edge.

.يكون متجاوباً ويعمل بكفاءة على الأجهزة المحمولة.

- يجب استخدام إطار العمل .NET والمكتبة React لتنجيز المشروع، لأن مهندسي المعهد العالي يعملون على إعادةبناء الأنظمة البرمجية المستخدمة ضمن حرم المعهد العال، إذ سيقومون باستكمال المشروع بعد تقديمه وهذه هي أطر العمل المستخدمة في عملية إعادة البناء.

الفصل الثاني

# الدراسة النظرية والمرجعية

نقدم في هذا الفصل فكرة عن أهمية نظام تذاكر الصيانة ،بالإضافة الى إستعراض تطبيقات مشابهة لنظامنا.

## 2.1- أهمية نظام تذاكر الصيانة

نظام تذاكر الصيانة هو أداة حيوية لإدارة طلبات الصيانة في المؤسسات التعليمية أو الصناعية. يهدف هذا النظام إلى أتمتة عملية تقديم الطلبات، تتبعها، وحلها بكفاءة، مما يقلل من الوقت والجهد المبذول في الإجراءات اليدوية. يعمل النظام على تحسين التواصل بين الموظفين وفرق الصيانة عبر إشعارات فورية عند إنشاء التذاكر أو تحديثها كما يعتمد النظام على مفاهيم هندسة البرمجيات وقواعد البيانات، مع دمج تقنيات مثل LDAP للمصادقة و TLS لتأمين الاتصالات حيث يجمع بين ميزات الأنظمة المشابهة مثل Zendesk,osTicket هذا يجعله حلًا متكاملاً وفعالاً لإدارة طلبات.

## 2.2- التطبيقات والأنظمة المشابهة

### 1.2.2- نظام osTicket

**الميزات:**

1. إدارة فرق العمل يسمح بإنشاء فرق متعددة وتخصيص مسارات عمل (Workflows)
2. واجهة بسيطة مناسبة للمستخدمين غير التقنيين لإنشاء التذاكر ومتابعتها.
3. يدعم تعدد اللغات مناسب للمؤسسات متعددة الجنسيات.

**المساوئ:**

1. واجهة مستخدم قديمة تصميم غير حديث مقارنةً بمنافسيه .
2. لا يوجد دعم فني رسم :يعتمد على مجتمع المستخدمين لحل المشكلات.
3. يتطلب صيانة دوريةكونه مفتوح المصدر، يحتاج إلى تحديثات أمنية وبرمجية مستمرة.
4. اعتماده الكامل على الإنترنت غير مناسب للمؤسسات التي تفضل حلولًا محلية (.(On-Premise


شكل osTicket

الفصل الثالث

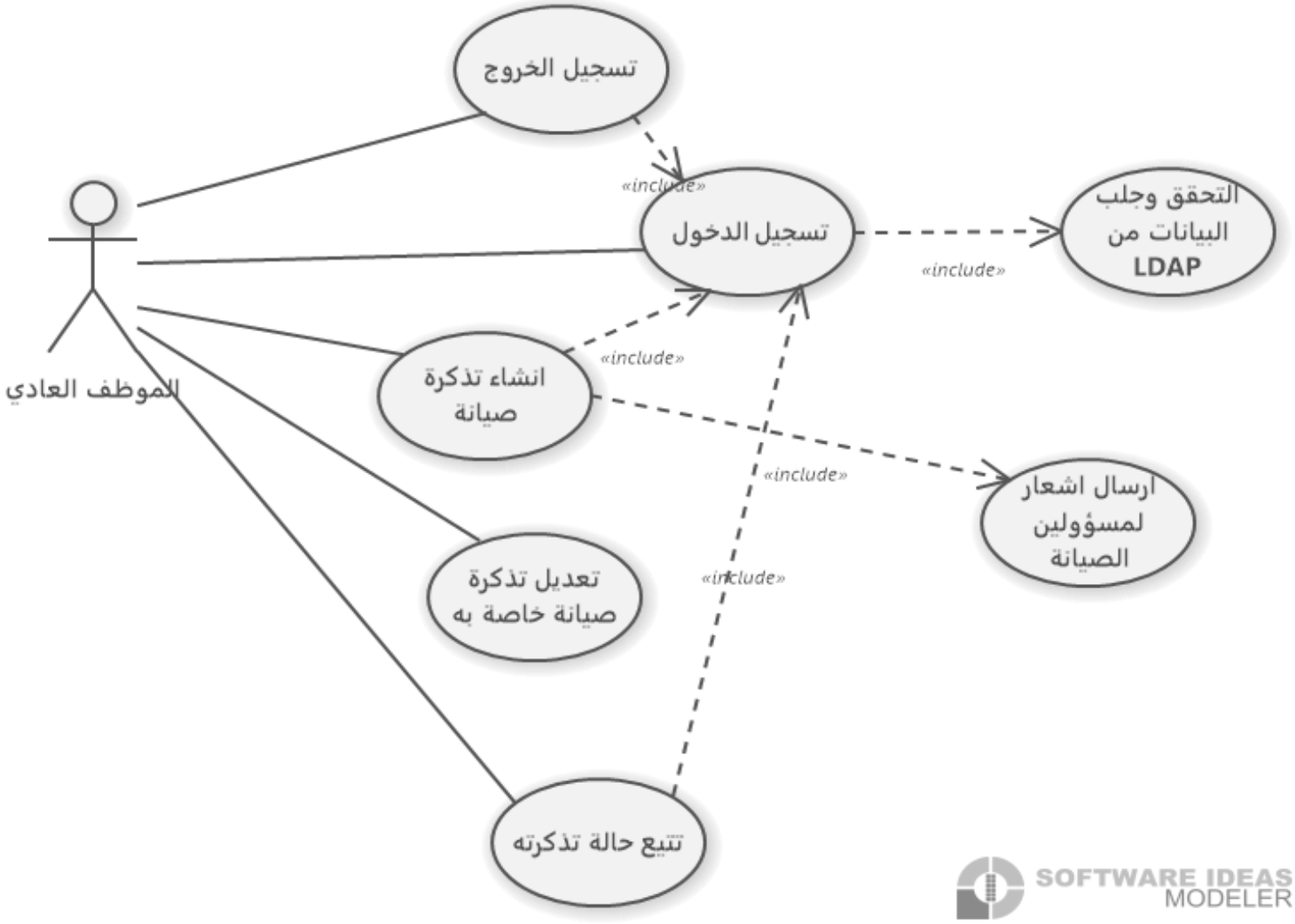
# الدراسة التحليلية

يقدم هذا الفصل تحليلا مفصلا لما ذكرناها في الفصل الأ ول من المتطلبات )الوظيفي ة وغير الوظيفية(، حيث نورد مخطط حالات الاستخدام مع الوصف النصي الخاص بكل منها.

## 1.3- مخططات حالات الاستخدام

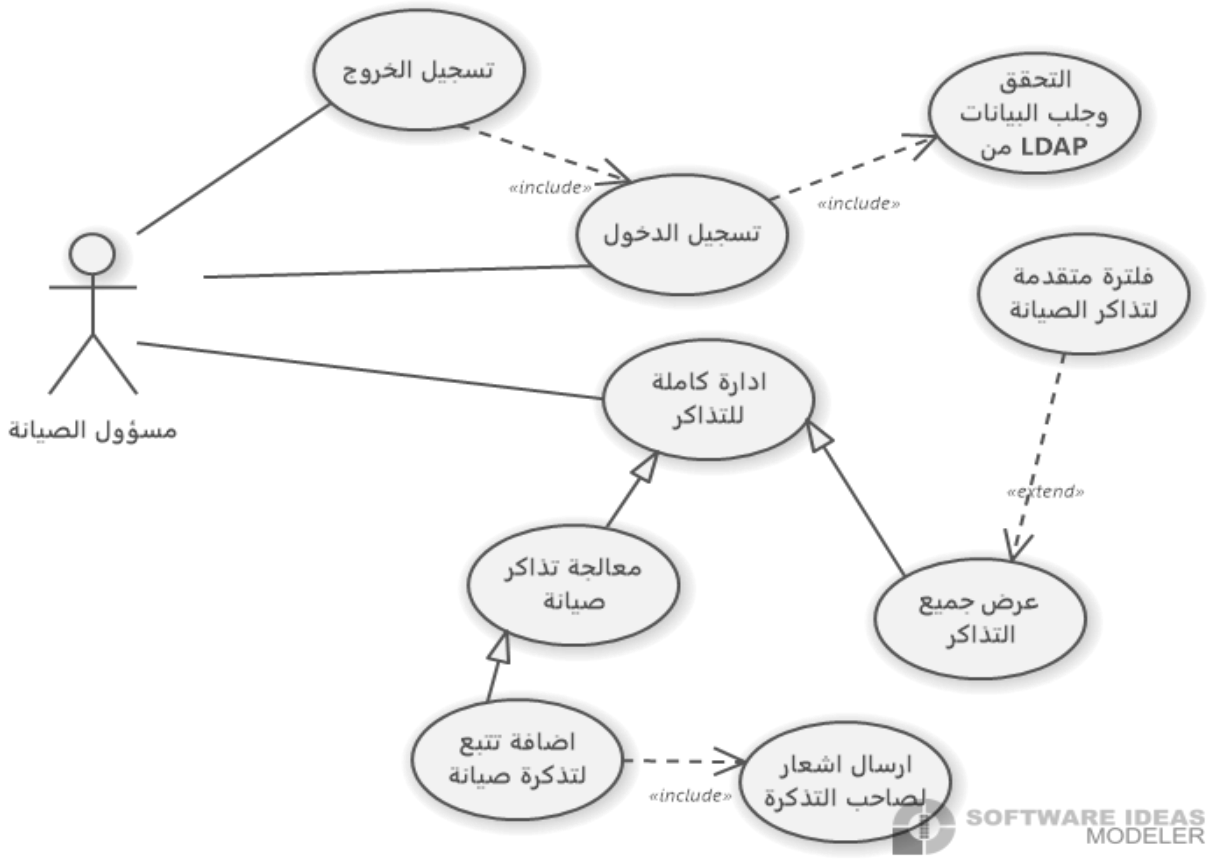
## 3.2- مخطط حالات الإستخدام

- للموظف:



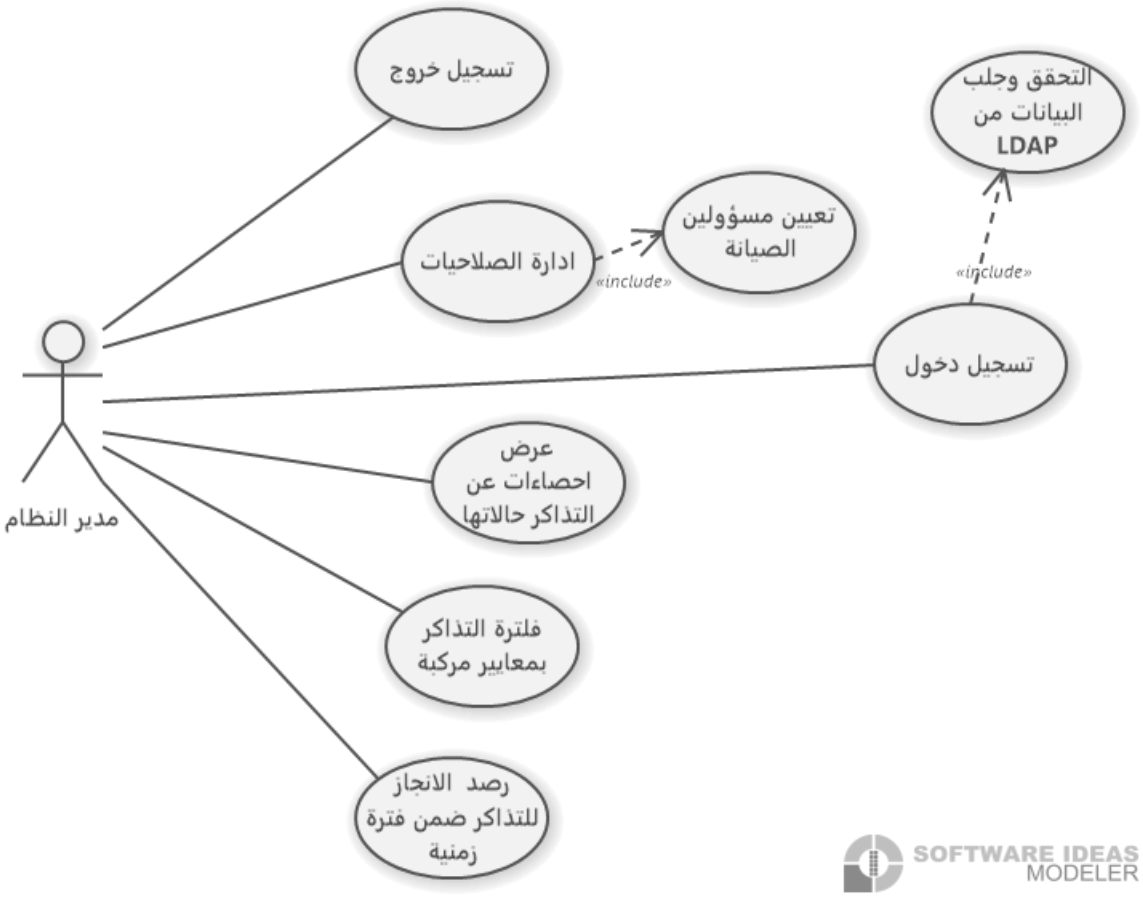

شكل 4- مخطط حالات الإستخدام بالنسبة للموظف

❖ يوضح الشكل التالي مخطط حالات المستخدام بالنسبة للابن:




شكل 5- مخطط حالات الإستخدام بالنسبة لمسؤول الصيانة.

❖ يوضح الشكل التالي مخطط حالات المستخدام بالنسبة لمدير النظام:




شكل 6- مخطط حالات الإستخدام بالنسبة لمدير النظام

## 3.3- الوصف النص ي لحالات الاستخدام ومخططات التسلسل

نستعرض الوصف النصي لجزء من حالات الإستخدام المبينة في الأشكال التوضيحية الموضحة أعلاه، وبقية الحالات موجودة في الملحق آ.

### A. الموظف:

#### 3.3.1- حالة الاستخدام: تسجيل دخول الموظف.

- النمط: أساسية.
- الفاعلون: الموظف.
- الظروف السابقة:

- يجب على الموظف أن يكون مُسجلاً في النظام LDAP.

- السيناريو الرئيسي الناجح:

| النظام | الفاعلون |
|---|---|
| | 1. تبدأ حالة الاستخدام هذه عندما يريد الموظف تسجيل الدخول إلى النظام فيقوم بإدخال بياناته(اسمه, كلمةالمرور,القسم) وإرسالها للنظام. |
| 2. يقوم النظام باستقبال هذه المعلومات للتحقق منها وإذا تمت عملي ة التحقق بنجاح يقوم بإدخال المستخدم إلى النظام ويعطيه Token خاصة فيه تحوي هذه ال Token معلومات عن ال id الخاص بال user ، حيث أ ن هذه ال Token تختلف من مُستخدم لأخر. | |

جد ول 1-السيناريو الرئيسي لحالة تسجيل دخول.

- المسارات البديلة:

| |
|---|
| في الخطوة 2: في حال كانت بيانات الموظف المدخلة غير صحيحة، يرسل النظام للأب تنبيهاً بفشل عملية تسجيل الدخول مع توضيح السبب ضمن الرسالة حيثُ أنه يوضح الخطأ وأسبابه ويطلب من ا لمستخدم إعادة المحاولة |

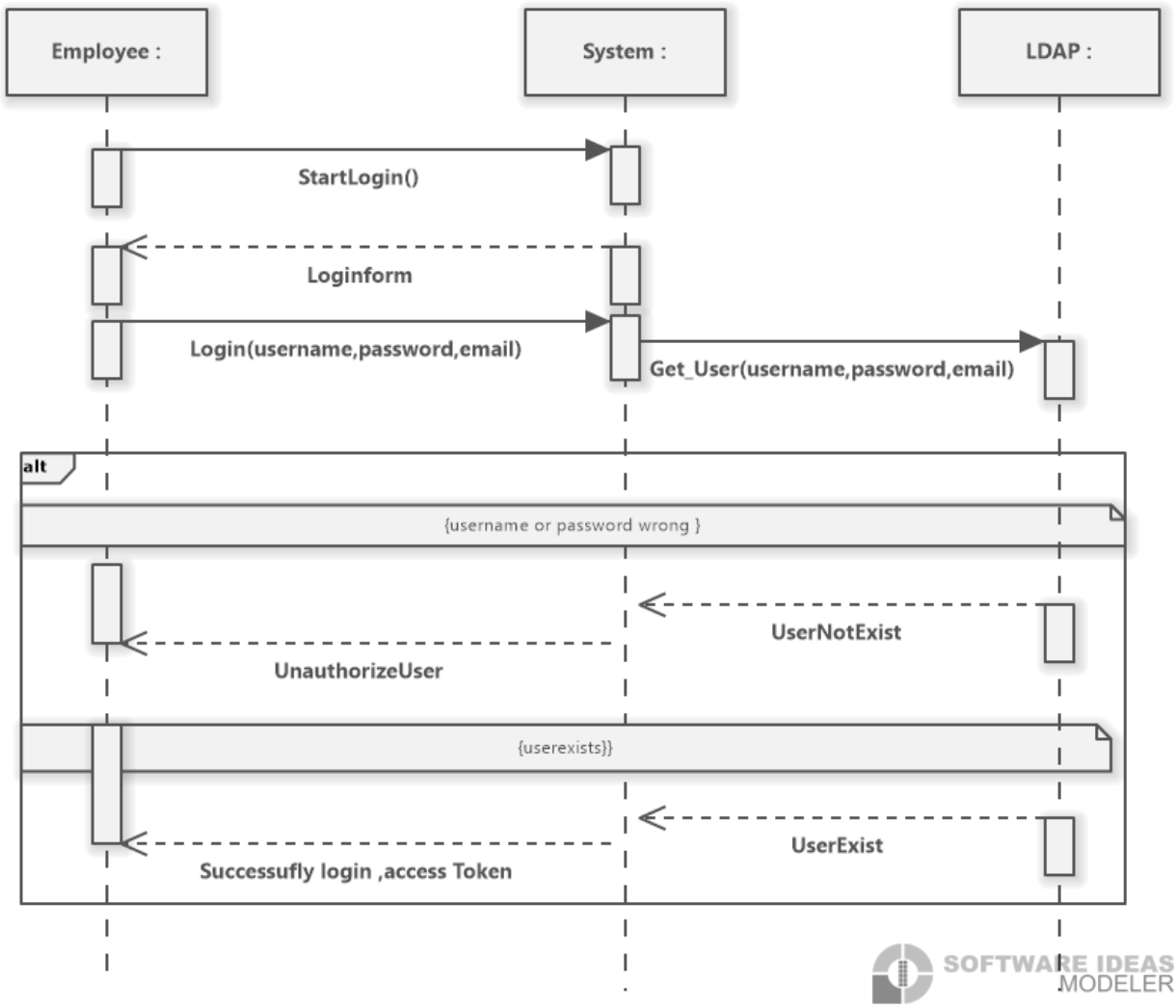
جدول 2-السيناريو البديل لحالة الاستخدام تسجيل الدخول.

- الظروف ال لاحقة:

- تم حفظ حالة الموظف كمتصل ويصبح داخل النظام وله access Token خاصة به ومن هنا يتمثل تحقيق الأمان بالنظام.

- مسارات خاطِئة:
  لا ي وجد.

- مخطط التتالي لحالة الإستخدام:




شكل 7- SSD تسجيل دخول.

### 2.3.3- حالة الاستخدام: استعراض الاحصاءات

- النمط: أساسية.
- الفاعلون:المدير .
- الظروف السابقة:

- يجب على المدير أن يكون مُسجل مسبقا في النظام.

- السيناريو الاساسي الناجح:

| الفاعلون | النظام |
| --- | --- |
| 1. تبدأ حالة الاستخدام هذه عندما يريد المدير رؤية احصاءت عن التذاكر. | |
| | 2. يقوم النظام بإظهار لائحة الخاصة بالاحصاءات عن التذاكر التي تحوي مخططات ورسومات. |

جدول 3-السيناريو الرئيسي لحالة استعراض احصاءات الخاصة بالتذاكر

المسارات البديلة:

| في الخطوة2: إذا لم يكن هتاك أي تذاكر بالنظام يرجع النظام للمدير رسالة |
|---|

| الخطوة. " Tikets Not Found " وتنتهي حالة الإستخدام عند هذه |
|---|

جدول 4-السيناريو البديل لحالة الاستخدام عرض إحصاءات التذاكر

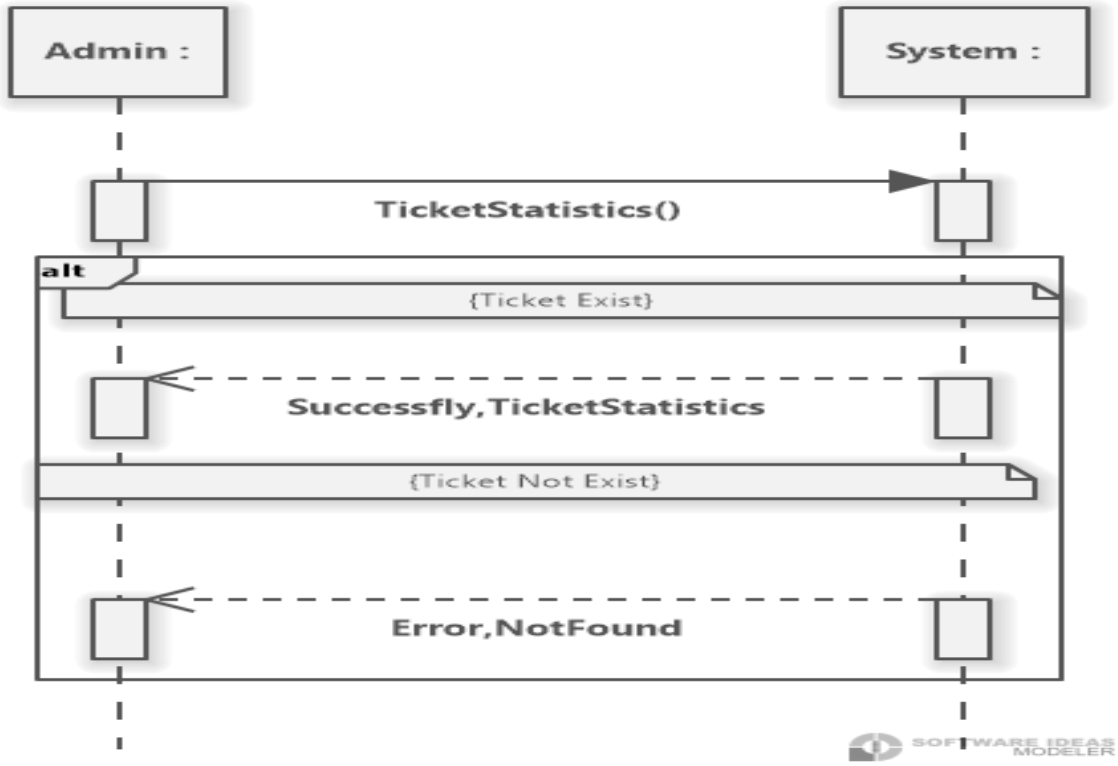
- الظروف اللاحقة:

- ظهور التفاصيل الخاصة بالاحصاءات

- مسارات خاطِئة:

لا يوجد.

- مخطط التتالي لحالة الإستخدام:




شكل 8- عرض الإحصاءات الخاصة عن التذاكر

## B. الموظف

### 3.3.3- حالة الاستخدام: إنشاء تذكرة جديد ة

- النمط: أساسية.
- الفاعلون:الموظف.
- الظروف السابقة:

- يجب أن يكون الموظف متصل بالنظام.

- السيناريو الرئيسي الناجح:

| النظام | الفاعلون |
|---|---|
| | 1. تبدأ حالة الاستخدام هذه عندما يريد الموظف انشاء تذكرة جديدة، فيقوم بإدخال بيانات التذكرة المطلوبة منه وهي ملاحظات ، نوع الجهاز، صور للجهاز-مرفقات. |
| 2. يقوم النظام في هذه الحالة بالتحقق فيما إذا كانت البيانات صحيحة بارسال رسالة توضيحية بنجاح العملية | |

جدول 5-السيناريو الرئيسي لحالة الاستخدام إنشاء حساب جديد للابن

- مسارات خاطِئة:

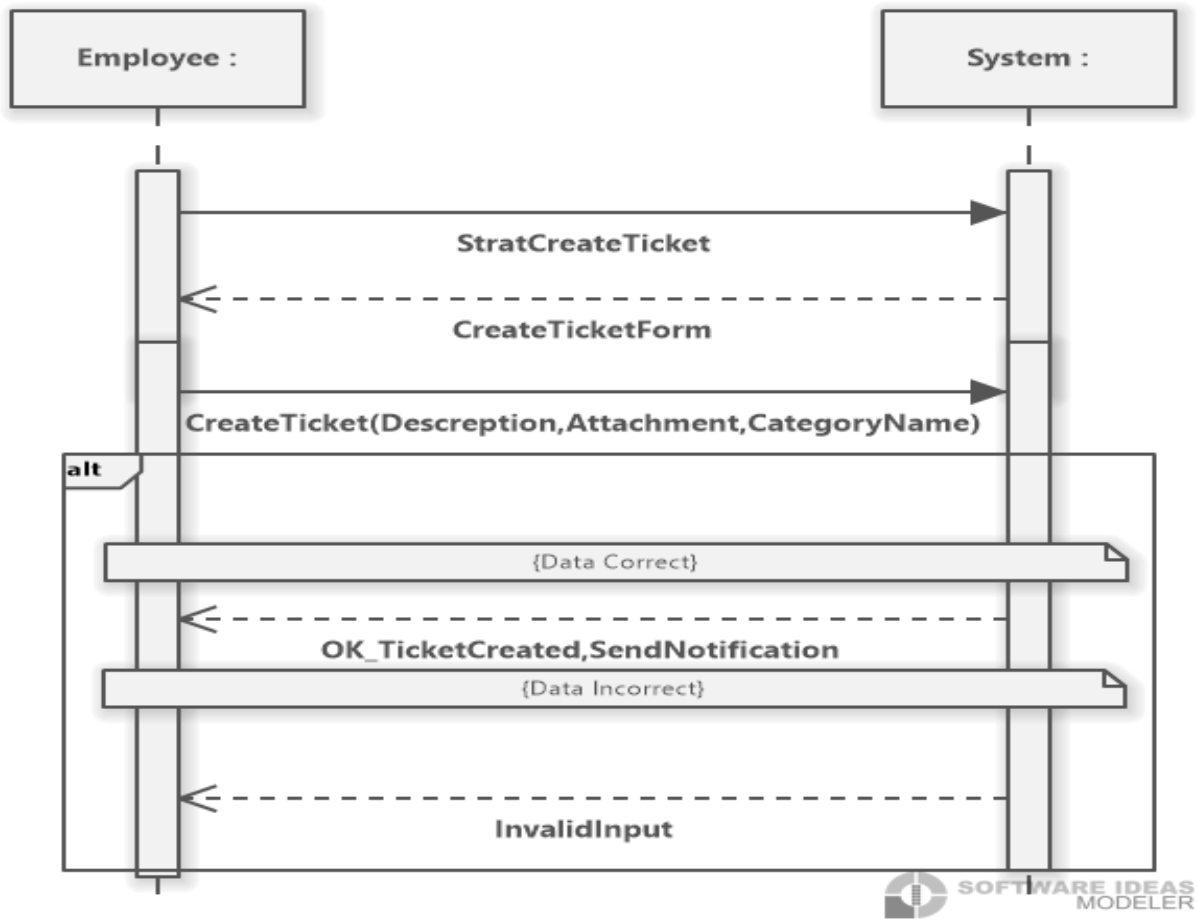
- إدخال خاطئ للمعلومات عندها يطلب النظام من الموظف إعادة الإدخال مجدداً مع إظهار رسالة الخطأ.

المسارات البديلة: لا يوجد.

 الظروف اللاحقة:

- حفظ معلومات التذكرة الجديدة في النظام.
- ارسال اشعار لموظفين الصيانة.

 مخطط التتالي لحالة الإستخدام:




شكل SSD-9 إنشاء تذكرة جديد للموظف

### 4.3.3- حالة الاستخدام: معالجة تذكرة

- النمط: أساسية.
- الفاعلون:موظف الصيانة.
- الظروف السابقة: وصول اشعار بإضافة تذكرة جديدة او يوجد تذكرة بحالة قيد المعالجة.

- يجب أن يكون موظف الصيانة مسجل لدى النظام.

- السيناريو الرئيسي الناجح:

| الفاعلون | النظام |
|---|---|
| 1. تبدأ حالة الاستخدام هذه عندما يريد موظف الصيانة بتعديل حالة تذكرة . | |
| | 2. يقوم النظام بعرض جميع التذاكر للموظف |
| 3. يختار الموظف التذكرة التي يريد تعديل حالتها . | |
| | 4. يقوم النظام بعرض واجهة لاضافة تتبع تذكرة |
| 5. يقوم الموظف بإدخال البيانات . | |
| | 6. يقوم النظام بالتحقق من صحة البيانات المدخلة ويرسل رسالة بنجاح العملية ويرسل اشعار للموظف الموافق للتذكرة. |

جدول 7-السيناريو الرئيسي لحالة الاستخدام معالجة تذكرة

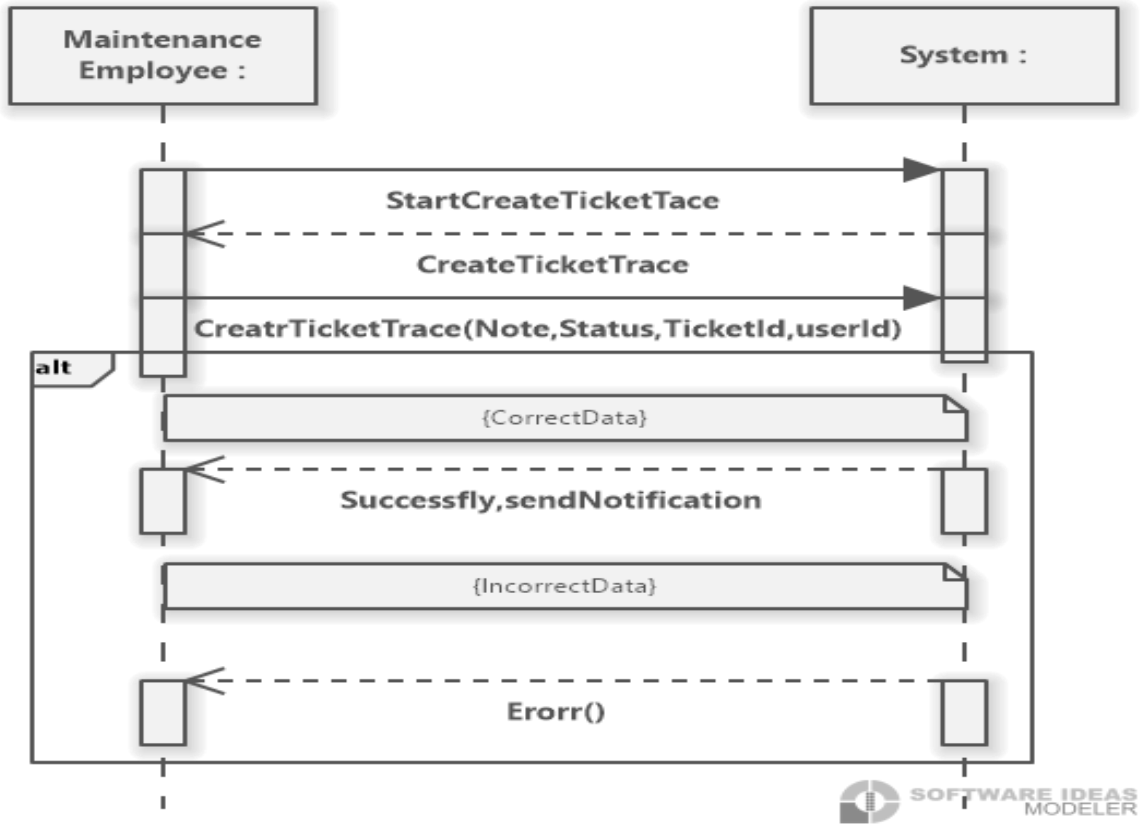
- مسارات خاطِئة:

- إدخال خاطئ للمعلومات عندها يطلب النظام من الموظف إعادة الإدخال مجدداً مع إظهار رسالة الخطأ.

- المسارات البديلة: لا يوجد.

الظروف اللاحقة: تم انشاء تعديل حالة تذكرة بنجاح وارسال اشعار للموظف.

- مخطط التتالي لحالة الإستخدام:




شكل 10- **SSD** تعديل حالة تذكرة

### 5.3.3- حالة الاستخدام: إستعراض وفلترة التذاكر .

- النمط: أساسية.
- الفاعلون: المدير .
- الظروف السابقة:

- يجب أن يكون المدير مسجل دخول في النظام.

- السيناريو الرئيسي الناجح:

| الفاعلون | النظام |
| --- | --- |
| 1. تبدأ حالة الاستخدام هذه عندما يريد المديرإستعراض التذاكر وفق معايير معينة | |
| | 2. يعرض النظام للمدير واجهة لفلترة التذاكر |
| 3. يقوم المدير بإدخال البيانات الفلترة | |
| | 4. يقوم النظام بعرض التذاكر الموافقة لمعايير الفلترة بعد التأكد من صحة المعايير |

جدول 9-السيناريو الرئيسي لحالة الاستخدام إستعراض وفلترة التذاكر .

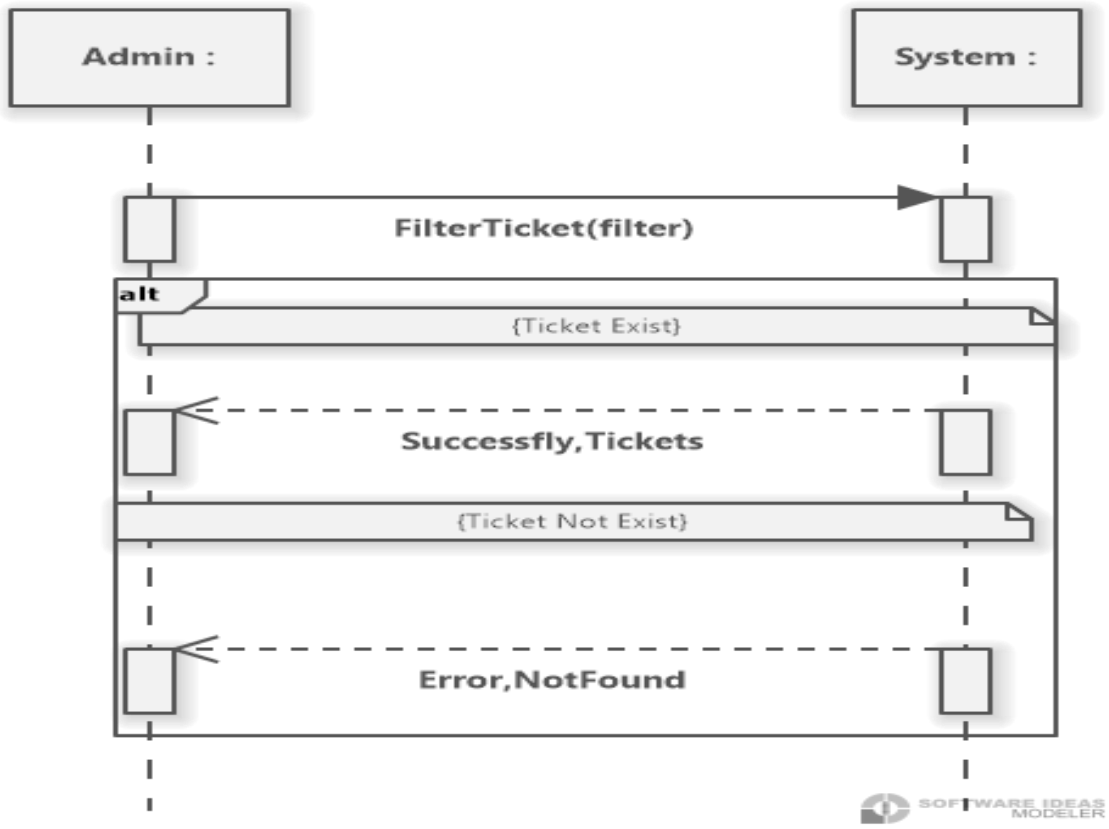
- مسارات خاطِئة أو بديلة:

- إدخال خاطئ للمعلومات عندها يطلب النظام من المدير إعادة الإدخال مجدداً مع إظهار رسالة الخطأ.

الظروف اللاحقة:

- ظهور لائحة بالتذاكر الموافقة لعايير الفلترة .

- مخطط التتالي لحالة الإستخدام:




شكل 11- **SSD** حالة الاستخدام إستعراض وفلترة التذاكر

### 6.3.3- حالة الاستخدام: عرض كل التذاكر .

- النمط: أساسية.
- الفاعلون: موظف الصيانة .
- الظروف السابقة:

- يجب أن يكون الموظف مسجل دخول في النظام.

-

- السيناريو الرئيسي الناجح:

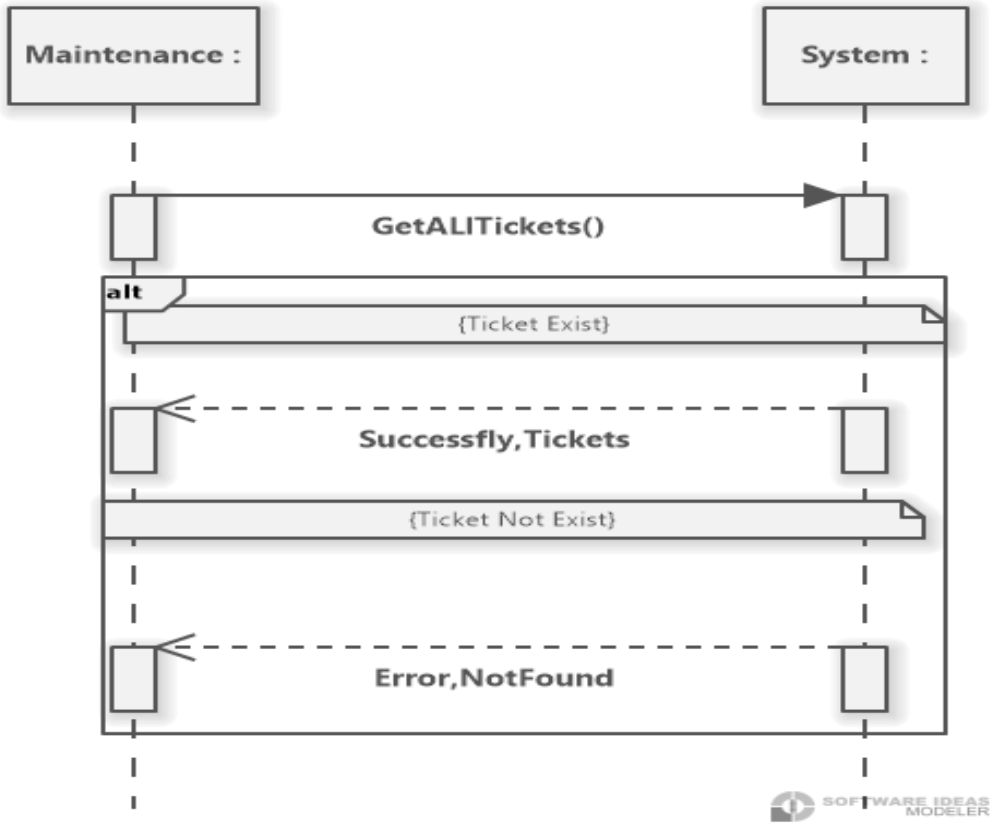
| الفاعلون | النظام |
|---|---|
| 1. تبدأ حالة الاستخدام هذه عندما يريد الموظف استعراض التذاكر الموجودة . | |

| | |
|---|---|
| 2. يعرض النظام للموظف قائمة بالتذاكر الموجودة . | |

جدول 10-السيناريو الرئيسي لحالة الاستخدام عرض التذاكر

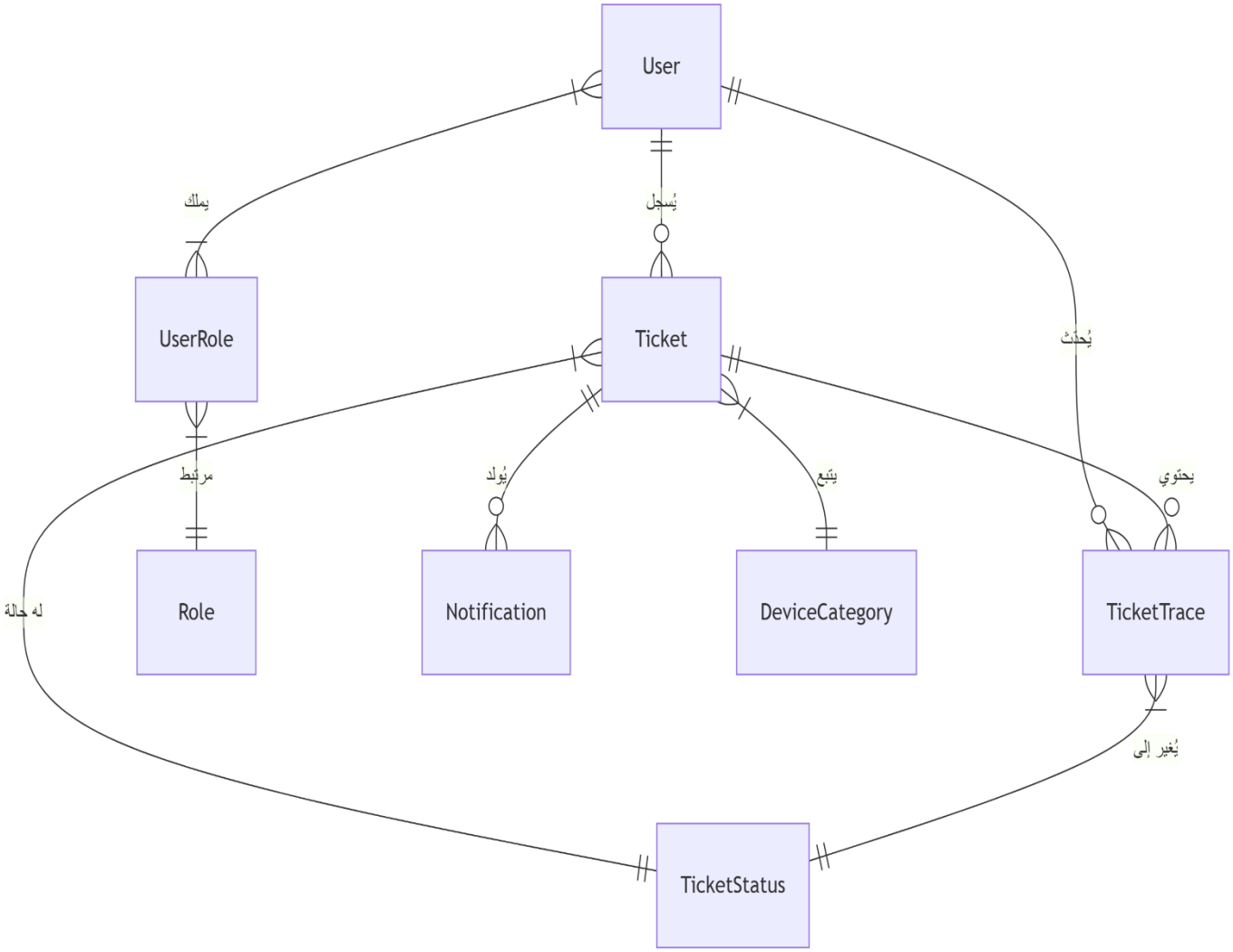
- مسارات بديلة او خاطئة : عدم وجود أي تذاكر يقوم النظام بارسال رسالة توضيحية.
- الظروف اللاحقة: عرض كل التذاكر الموجودة.

- مخطط التتالي لحالة الإستخدام:




شكل 11- SSD حالة الاستخدام إستعراض وفلترة التذاكر

## -. **مخطط الصفوف** (class diagram)




الشكل 10 مخطط يمثل الصفوف

# الفصل الرابع

# بيئة العمل و الدراسة التصميمية

يقدم هذا الفصل شرحا موجزا عن المنهجيات والأساليب المستخدمة في تنجيز هذ النظام .

## 1.4- البنية المعمارية Architecture

البنية المعمارية هي هيكل وتنظيم عال المستوى للنظم البرمجية ت شتمل على قرارات التصميم الأساسية التي تشكل مكوناتالنظام وعلاقاتها وكيفية تفاعله ا مع بعضها البعض، حيث يساعد بناء هندسة معمارية جيدة على تصميم نظام مرن قابل للصيانة والتكيف مع التغييرات بمرور الوقت [5]

## 2.4- .- البنية المعمارية النظيفة Clean Architecture

### 1.2.4- مفهوم البنية المعمارية النظيفة

هي فلسفة لتصميم البرمجيات ونمط تصميمي معماري، يهدف إلى إنشاء نظام برمجي قابل للصيانة ومستقل عن تفاصيل التنفيذ .يعزز هذا النهج استقلالية المكونات المختلفة للنظام، مثل واجهات المستخدم ومنطق الأعمال وتخزين البيانات وغيرها، ممايسمح لهذه المكوّنات بالتطور بشكل مستقل دون التأثير على النظام بأكمله، كما توفر هذه المنهجية قاعدة رماز **(codebase)**قابلة للاختبار، مما يسهل إدارة وتوسيع النظام البرنامجي [**5**].

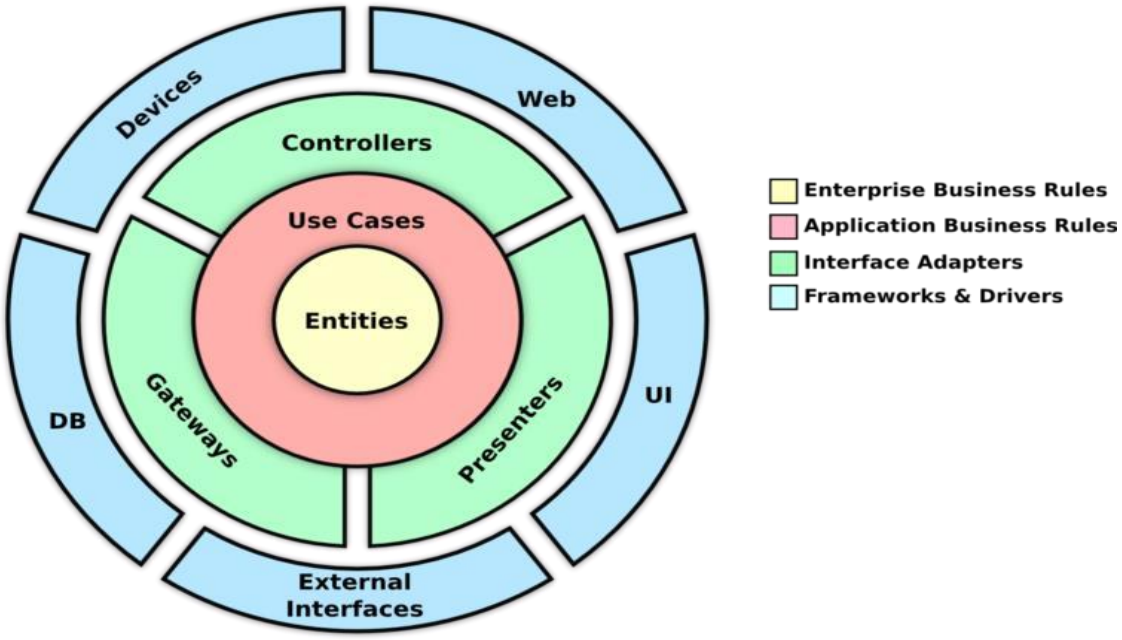
ينتج عن استخدام البنية النظيفة نظام يحمل الخواص التالي ة [**5**]:

- **مستقل عن أطر العمل (Independence of Frameworks)**: لا يرتبط منطق العمل الأساسي والمكونات

الأساسية للنظام بأي إطار خارجي أو مكتبة أو تقنية معينة، الأمر الذ ي يسمح باستبدال أو تحديث هذه التبعيات الخارجية دون التأثير على الوظائف الأساس يّة للنظام .

**استقلالية واجهات المستخدم (Independence of User Interface):** تكون واجهة المستخدم منفصلة عن بقية النظام، مما يسمح بإجراء تغييرات على واجهة المستخدم دون التأثير على منطق الأعمال الأساسي، بالإضافةإلى إمكانية وجود واجهات مستخدم متعددة )الويب، الجوال، ..إلخ( تستخدم نفس منطق الأعمال الأساسي.

- **استقلالية تخزين البيانات (Independence of Data Storage)**: إمكاني ة تغيير آلية تخزين المعطيات دون التأثيرعلى بقية أجزاء النظام .
- **نظام قابل للاختبا ر (Testable):** إمكاني ة اختبار قواعد العمل بدون واجهة المستخدم أو قاعدة البيانات أو خادم الويب أو أي عنصر خارجي آخر.




الشكل مكونات البنية النظيفة

## 2.2.4-- قاعدة التبعية

تعد قاعدة التبعية مبدأ أساسي ا يوجّ ه كيفية تنظيم المكونات والوحدات داخل النظام البرمجي وتفاعلها مع بعضها البعض في البنيةالنظيفة. تنص قاعدة التبعية على أن التبعيات يجب أن تشير دائ ما إلى الداخل نحو جوهر النظام، مما يعني أن المكونات ذاتالمستوى الأعلى والأكثر تجري دا وثباتا يجب أ لّا تعتمد على المكونات ذات المستوى الأدنى، بدلا من ذلك، يجب أن تتدفق التبعيات من الطبقات الخارجية نحو الطبقات الداخلية للبنية [5].

## 4.2.3- مكونات البنية النظيفة

تتكون البنية النظيفة بشكل رئيسي من الطبقات التالية [5]:

- **Entities**: تمثّل الكيانات عناصر الأعمال الأساسية وقواعد التطبيق، حيث تقوم بتغليف البيانات الأساسية والسلوكالذي يحدد الغرض من التطبيق. تكون هذه الكيانات مستقلة عن العوامل والأطر الخارجية، وتركز فقط على تمثي المجال الأعمال.

- **Use Cases**: تحدد حالات الاستخدام منطق عمل التطبيق وتنظم التفاعلات بين الكيانات، حيث تلخّ ص العملياتالمحددة ومهام سير العمل التي يدعمها التطبيق، وتكون حالات الاستخدام مستقلة عن الواجهات الخارجية والتقنيات.

- **Interface Adapters**: محولات الواجهة تسد الفجوة بين حالات الاستخدام والعالم الخارجي، حيث تعمل على تحقيق التواصل وتحويل البيانات بين نواة التطبيق وأطر العمل الخارجية، مما يضمن بقاء النواة معزولة عن تفاصيل التنفيذ .

- **Frameworks and Drivers**: الأطر والمحركات هي الطبقة الخارجية للبنية، ال تي توفر الأدوات والبنية التحتيةالمطلوبة لتشغيل التطبيق، مثل أطر عمل الويب ومكتبات واجهة المستخدم وقواعد البيانات وغيرها. تتفاعل هذه الأدوات مع التطبيق من خلال محولات الواجهة.

## 3.4- Domain Driven Designالتصميم المقاد بالمجال

التصميم المقاد بالمجال هو منهجية لتطوير البرمجيات ونهج معماري يركز على خلق فهم مشترك بين المطورين وخبراء المجال. يهدف إلى تصميم وهيكلة أنظمة برمجي ة بطريقة تتماشى مع تعقيدات ومتطلبات مجال العمل الأساسي [6].

يقدم التصميم المقاد بالمجال عدة مفاهيم أهمها [6] :

- **Entities**: يعبّر الكيان عن كائن (object) يمكن التعرف عليه بشكل فريد من خلال هويته، لا سماته (attributes) فقط. يمثل الكيان شيئ ا له وجود مُم يَّز ويحافظ على هو يّته طوال دورة حياته، حتى عند تغير سماته. يؤكد مفهوم الكيانعلى أن الهوية هي الأهم، وتظل ثابتة على الرغم من التعديلات التي قد تحصل على سماته، الأمر الذ ي يفيد في ضماناستقرار النظام من خلال عدم الخلط بين الكيانات المخت لفة الذي قد يؤدي إلى تلف البيانات وحصول تناقضاتضمن النظام.

- **Value objects**: يعرف كائن القيمة على أنه كائن ليس له هوية فريدة حيث يتم تحديد هويته من خلال سماته،ومنه نقول عن كائني قيمة أنهما متساويان إذا كان لهما نفس السمات. يتميز كائن القيمة بأنه غير قابل للتغيير ممايعني أن حالت ه تبقى ثابتة لا تتغير بمجرد إنشاءه.

  تتلخّص أهمية استخدام كائنات القيمة فيما يلي :

  - يساهم في تغليف المفاهيم الموجودة ضمن مجال معين والتي لا تحتاج لهوية خاصّة ليتم تعريفها، نذكر مثلا :
    تمثيل التاريخ أو مبلغ مال.

- تساعد على ضمان الد قّة في محاكاة الواقع مما يفيد في تعزيز التواصل بين خبراء المجال والمطورين .
- تساعد على تحسين الأداء ضمن النظام، نظرا لأنّ كائن القيمة غير قابل للتغيير مما يسمح بتخزينه مؤقتا وإعادة استخدامه دون القلق من الآثار الجانبية غير المقصودة.

- **Aggregate**: التجميع هو عبارة عن مجموعة من كائنات المجال (domain objects) المرتبطة ببعضها البعض، وال تييتم التعامل معها كوحدة واحدة يجري من خلالها تخزين البيانات واسترجاعها. يتألف التجميع بشكل رئيسي من جذرتجميعي (aggregate root) يمثل الكيان الرئيسي ضمن التجميع، ويحتوي على مجموعة من الكيانات (entities) وكائنات القيمة) objects value(. يمكن الوصول إلى محتويات التجميع وتعديلها من خلال الجذر التجميعي فقط،الأمر الذ ي يضمن الاتساق ضمن النظام والحفاظ على استقرار حالته الداخلية.

تتلخّص أهمية استخدام التّجميعات فيما يلي :

- تساعد على إيجاد حدود واضحة ضمن نموذج المجال الكلي، مما يسهل التفكير في التفاعلات والتبعيات داخل النظام.
- تسمح بالتركيز على وحدة متماسكة صغيرة ضمن المجال الكلي، مما يضمن سهول ة إدارة العمليات ضمن النظام.

- **Services**: يمكن تعريف الخدمات على أنّه ا المكونات التي تغ لّف سلوك النظام الذي لا ينتمي لكيان محدد. توفرالخدمات طريقة لنمذجة التفاعلات الحاصلة بين عدّة كائنات، وعند استخدامها يمكن إبقاء كائنات المجال مركزة علىمسؤوليتها الأساسية، مما يزيد من قابل يّة القراءة والصيانة للرماز البرمجي.

- **Repositories**: هو نمط تصميمي يستخدم لإدارة تخزين واسترجاع كائنات المجال، يعتمد على تجريد التعامل معآلية تخزين البيانات مما يحمي بقية أجزاء النظام من تعقيدات التخزين والاسترجاع، ويقلل الاعتماديات ضمن النظام.

تترتب على عائق المستودعات مسؤوليّ ة تمثيل البيانات ضمن المخزن سواء كان المخزن قاعدة معطيات أو نظام ملفاتأو غيره، ثم إعادة استرجاعها وتحويلها لكائنات المجال المناسبة.

- **Sub-domain**: يشير النطاق الفرعي إلى جزء مميز ومحدد من نطاق الأعمال الكلّي، حيث يتم عزل المجالات الفرعيةوت عَامَل ضمن سياق المجا ل الكلّي على أنّه ا مجال منفصل .

تتلخّص أهم يّة استخدام المجالات الفرعية فيما يلي :

- تقدم حدودا واضحة تساعد على فصل جوانب العمل المختلفة.

- تسمح لفرق المطوّرين بالتخصص ضمن مجالات عمل محددة، مما يؤدي إلى فهم أكبر للمجال المطروح دونالغوص في جميع قواعد المجال.

## 4.3- Logging التسجيل

يعد التسجيل أحد الممارسات المهمّة في تطوير البرمجيات، حيث يتم من خلاله التقاط وتسجيل المعلومات المتعلقة بسلوكالنظام. تتلخّص أهمية استخدام التسجيل فيما يلي:

- يساعد على اكتشاف الأخطاء وإصلاحها.
- يساعد على مراقبة سير عمل النظام وتسجيل العمليات المنفّذة من قبل المستخدمين بقاء النظام في حالة مستقرة ، واكتشاف التلاعب من قبل المستخدمين.
- يساعد على تحسين أداء النظام، حيث نستطيع من خلال التسجيل معرفة معوقات الأداء والعمل على معالجتها.
- يساعد على وضع سجل تاريخي لعمل النظام لإعطاء متخذي القرار معلومات مفيدة لوضع خطط مستقبلية.

## 4.4 - آليات معالجة تعدد المستخدمين

إن ضمان تكامل وصحّة البيانات هو حاجة ضمن أي نظام برمجي خصوصا في حال تعدد المستخدمين، فمثلا إذا قام عدة مستخدمين بقراءة وتعديل البيانات المشتركة في نفس الوقت، سينتج تضارب في البيانات المخزنة، يشكّل حله تحدّيا بالنسبة للمطورين.

نستعرض هنا منهجيتين لحل هذه المشكلة:

- **القفل المتفائل (Optimistic lock)**: نفترض في هذه المنهجية أن احتمال حدوث التضاربات صغير، حيث يتم السماح للمستخدمين بقراءة البيانات المشتركة والتعديل عليها، مع إقران رقم إصدار أو طابع زمني بكل سجل أو عنصر بيانات. عندما يريد مستخدم ما تحديث سجل معين، فإنه يقرأ أولا الإصدار الحال أو الطابع الزمني. بعد إجراء التغييرات، وعندما يحاول المستخدم حفظ التعديلات، يتحقق النظام مما إذا كان الإصدار أو الطابع الزمني يتطابق مع الإصدار المسجل مسبقا . إذا كانت متطابقة، يتم تطبيق التغييرات؛ أمّا في الحالة المعاكسة، فهذا يشير إلى حدوث تحديث آخر في هذه الأثناء ،عندها يقوم النظام بمعالجة هذه الحالة [7].

- **القفل المتشائم (Pessimistic lock):** يتخذ القفل المتشائم نه جا أكثر حذر ا من القفل المتفائل، حيث يفترض أنهمن المحتمل حدوث تعارضات عند محاولة عدة مستخدمين الوصول للبيانات

المشتركة، فيستخدم استراتيجية لمنع الوصول المتزامن إلى البيانات من قبل أكثر من مستخدم وحيد. وبالتال، عندما يريد مستخدم الوصول إلى عنصر معين لتعديله، يطلب قفلا على هذا العنصر. في حالة تم إعطاء القفل للمستخدم، تصبح البيانات غير متاحةللمستخدمين الآخرين حتى يتم تحرير القفل، الأمر الذ ي يمنع حدوث عدة تعديلات في نفس الوقت ويضم ن الحفاظعلى تكامل البيانات وصحتها [7].

### 4.4.1.5- نمط ال Repository

وهو طبقة وسيطة بين البرنامج وآلية التخزين، وبالتالي كود التطبيق يتعامل مع هذه الطبقة بدلاً من أن يعتمد على الية التخزين مباشرة والتي يمكن أن تتغير، كما في الصورة التالية سوف تجد أن طبقة التخزين Repository الآن وسيطة بين الكود وبين قاعدة البيانات


شكل 32- طبقة التخزين

# الفصل الخامس

# الأدوات المستخدمة

يقدم هذا الفصل شرحا عن الأدوات المستخدمة لتنجيز المشروع .

## 1.5- إطار عمل Net Core.

هو إطار عمل مفتوح المصدر ومتعدد المنصات يعمل على عدة أنظمة تشغيل، تم تطويره بواسطة Microsoft لإنشاء تطبيقات حديثة وعالية الأداء وقابلة للتطوير. يوفر تحسينات كبيرة من حيث السرعة واستخدام الذاكرة مقارنة ب .NET Framework.التقليدي .كما يدعم ممارسات التطوير الحديثة حيث أنه يتكامل بشكل جيد مع أدوات مثل Docker وKubernetes.

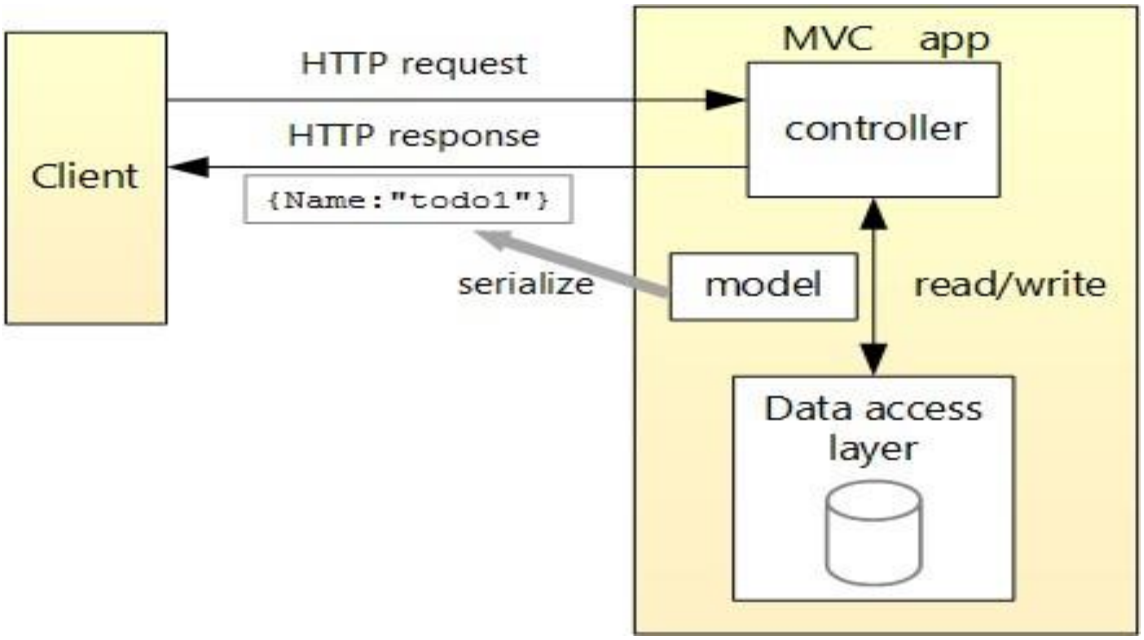
## 2.5- ASP.NET Web API

ASP.NET Web API هو عبارة عن إطار عمل توفره Microsoft لبناء خدمات HTTP يمكن أن يستهلكها العديد من العملاء، مثل متصفحات الويب وتطبيقات الهاتف المحمول وخدمات الويب الأخرى. يسمح إطار العمل المذكور للمطورين بإنشاء RESTful API باستخدام إطار العمل (Net Core.).

يقدم إطار العمل ASP.NET Web API العديد من الميزات أهمها [1]:

- ) ..Post, Put, Delete, Get,etc( . المختلفة HTTP يدعم أفعال

• يدعم وحدات التحكم Controllers التي تتعامل مع طلبات HTTP.
• يدعم التوجيه Routing حيث يقوم بربط الطلب الوارد إلى وحدة التحكم المناسبة.
• يدعم إرسال واستقبال البيانات بواسطة العديد من الطرق أهمها Json و XML .
• يدعم حقن التبعية Dependency injection مما يسهل إدارة واختبار التبعيات في النظام.
• يدعم التسلسل Serialization حيث يقوم بتحويل البيانات ضمن الجواب إلى Json أو XML حسب حقل Accept الموجود ضمن ترويسة الطلب




الشكل 13 النمط التصميمي ضمن إطار العمل **ASP.NET Web API**

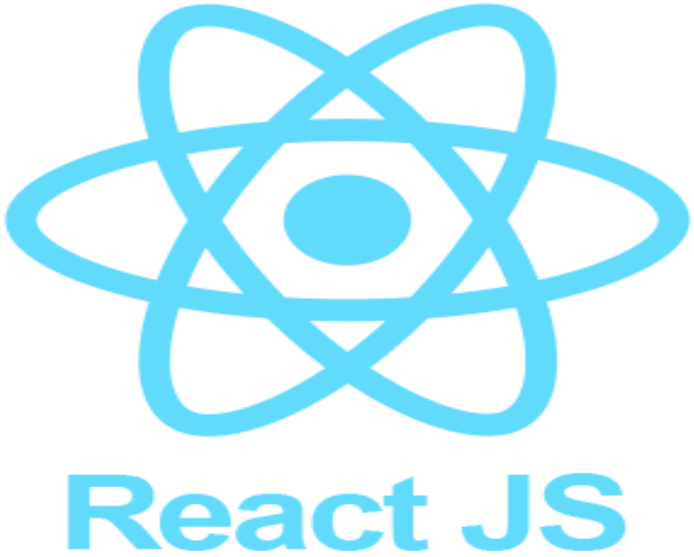
## React -3.5

React هي مكتبة JavaScript مفتوحة المصدر ،تستخدم لبناء واجهات المستخدم، خا صّة لتطبيقات الويب. تم تطويره ابواسطة شركة Facebook، وتستخدم على نطاق واسع لإنشاء واجهات مستخدم ديناميك ية وتفاعل يّة. تتبع React البنيةالقائمة على المكونات، مما يساعد المطورين على بناء مكونات واجهة المستخدم التي يمكن إعادة استخدامها وتكوينها م عا لإنشاء واجهات مستخدم معقّدة [2].

فوائد استخدام React:

- إحدى الميزات البارزة ل React هي استخدامها ل ( Document Object (Virtual DOM Model)(نموذج كائنالمستند الافتراضي(، حيث تعد Virtual DOM تمثيلا خفيفا ل DOM الفعلي، وتستخدمه React لتحديث الأجزاء الضروريّة فقط من واجهة المستخدم بكفاءة عند حدوث التغييرات. هذا يؤدي إلى أداء أفضل وتجربة مستخدم أكثرسلاسة.
- React قائمة على المكونات حيث تساعد المطورين على تقسيم واجهة المستخدم إلى مك وّنات قابلة لإعادةالاستخدام. هذا النهج المعياري يجعل التطوير والصيانة أسهل، حيث يمكن إعادة استخدام المكونات عبر أجزاء مختلفة من التطبيق .
- إن تدفق البيانات أحادي الاتجاه ضمن React، مما يبسط تتبع وإدارة تغييرات البيانات .

- تدعم التكامل مع التقنيات والمكتبات الأخرى، مثل مكتبات إدارة الحالة (State Management) مثل Redux ومكتبات مكونات واجهة المستخدمMaterial-UI و Ant Design.




شكل 17- **React JS**

## 7.5-Options pattern نمط الخيارات

يس تَخدم للوصول لإعدادات التطبيق (appsettings) ضمن بيئة العمل (.NET)، حيث يساعد على استخراج المتحولات الثابتة المكتوبة يدويا ضمن الصفوف (hard coded)، مما يساعد على تعديلها دون الحاجة لتعديل الرماز البرمجي [9] .

## 8.5- قاعدة معطيات SQL Server

هو نظام إدارة قواعد معطيات علائقية (relational database management system)، تم تطويره وتسويقة من قبل شركةMicrosoft. يعمل على نظامي التشغيل Windows و Linux، ويقدم أداء ممتازا وسعة تخزين عالية بالإضافة لآليا ت استعادةبيانات ممتازة [11] .

## Entity Framework -9.5

هو إطار عمل مقدم من قبل شركة Microsoft، يُم كّ ن مطوري NET. من التعامل مع البنى العلائقية المخ زّنة ضمن قاعدة المعطيات باستخدام كائنات خاصّ ة بالمجال (Domain-specific-objects) دون الحاجة لكتابة تعليمات SQL.

## Json Web Token (JWT) -10.5

الرمز (JWT) هو غرض JSON يستخدم لنقل المعلومات بين طرفين بشكل آمن عبر الويب، يتم استخدام JWT على نطاق واسع في التطبيقات الحديثة كآلية مصادقة عديمة الحالة. بعكس cookies رموز JWT يمكن استخدامها للمصادقة في تطبيقاتالويب، تطبيقات الجوال وتطبيقات سطح المكتب (Desktop application).

وهي اختصار JSON Web Token، تُعتبر كحزمة تستخدم للسماح للمستخدم بالإشارة إلى هويته لمزيد من التبادلات الموثوقة بينه وبين المخدم بعد إتمام عملي ة المصادقة. يتم استخدام JWT لنقل المعلومات المتعلقة بهوي ة وخصائص العميل. يتم توقي ع هذه المعلومات من قبل الخادم حُتى يتمكن من اكتشاف ما إذا كان قد تم العَبث بها بعد إرسالها إلى العميل وهذا سيمنع المهاجمون من تغي ير الهوي ة أو أي خصائص )على سبيل المثال، تغي ير كلمة الصلاحي ة من مُستخدم بسيط إلى مسؤول أو تغي ير تسجي ل دخول المستخدم(. يتم إنشاء الرمز الممي ز أثناء المصادقة )يتم توف يره في حالة المصادقة الناجحة فقط( ويتم التح قق منه بواسطة الخادم قبل أي مُعالجة )قبل أي عملي ة إرسال إلى ال API (. يتم استخدامُه بواسطة أحد التطبيقات للسماح للعميل بتقد يم رمز مُمي ز يُمث ل "بطاقة هوي ة" المستخدم إلى الخادم والسماح للخادم بالت حقق من صحة وسلامة الرمز الممي ز بطريقة أمنة.

# Grafana K6 -11.5

هي أداة اختبار للحمل (Load testing tool) مفتوحة المصدر تساعد في اختبار أداء البرمجيات خصوصا في حال تعدد المستخدمين. k6 مجاني ومتمحور حول المطورين وقابل للتوسيع. باستخدام k6، يمكن اختبار موثوقية وأداء الأنظمة البرمج يّةوالتعرف على تراجعات الأداء والمشكلات في وقت سابق [12]

## signalR-12.5

هي مكتبة مفتوحة المصدر من Microsoft تُستخدم لإنشاء تطبيقات ويب تفاعلية في الوقت الحقيقي .(Real-time) تعتمد على تقنيات WebSockets مع وجود حلول بديلة عند عدم توفرها اتصالات ثنائية الاتجاه :(Bi-directional)يسمح للخادم والعميل بإرسال البيانات في نفس الوقت.

دعم متصفحات قديمة يعمل حتى مع المتصفحات التي لا تدعم.WebSockets

تكامل سهل مع ASP.NET Core و.JavaScript

إدارة الاتصالات الذكية يختار أفضل تقنية اتصال تلقائيًاWebSockets) ، -Server Sent Events، .(Long Polling

## Generic Repository Pattern-13.5

نمط تصميمDesign Pattern) يُستخدم في طبقة الوصول إلى البيانات Data) Access Layer).

الهدف الرئيسي توفير واجهة عامة (Generic Interface) لعمليات CRUD الأساسية (Create, Read, Update, Delete) دون الحاجة لكتابة كود مكرر لكل كيان(Entity)

## MailKit -14.5

هي مكتبة مفتوحة المصدر متعددة المنصات لمعالجة البريد الإلكتروني في تطبيقات .NET. تُعتبر البديل الحديث لـ System.Net.Mailالقديم، وتوفر دعمًا فائقًا لأحدث بروتوكولات البريد مثل IMAP، POP3، و SMTPمع ميزات أمان متقدمة.

## Custom Action Result Pattern

JSON هذا النمط يُسمى Custom Action Result Pattern، وهو جزء من أنماط تصميم ASP.NET Core لإنشاء استجابات (Responses) مخصصة. دعنا نشرحه بالتفصيل: هذا النمط يُستخدم لإنشاء **استجابات API مخصصة** تتحكم بشكل كامل في شكل البيانات المرسلة للعميل، خاصة عند الحاجة لـ:

- توحيد شكل جميع الردود
- معالجة خاصة لبياناتJSON
- تحسين الأداء

الفصل السادس

# القسم العملي

يشرح هذا الفصل كيفية تنجيز النظام من الألف
إلى الياء

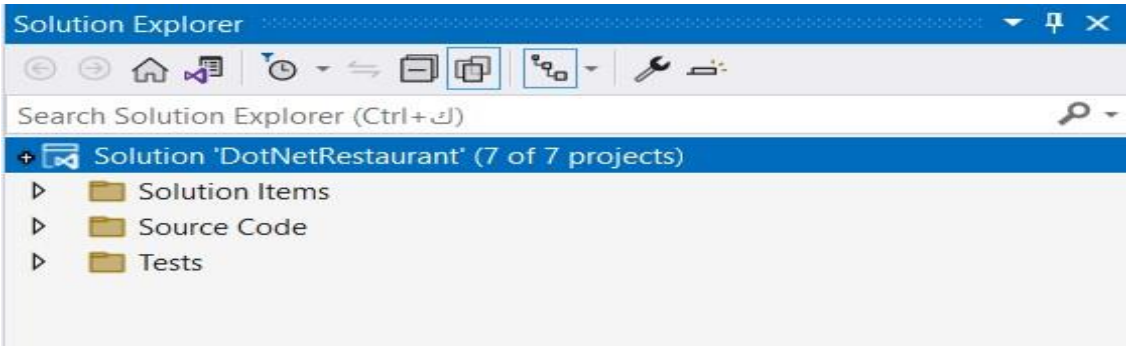
## 6.1- منهجية تنجيز النظام

### 1.1.6 - البنية المقترح ة

قابل (codebase)إن اعتماد منهجية واضحة لبناء النظام منذ البداية يساعد على إيجاد رماز مصدري للاختبار والتعديل بمرورالوقت. سيتم الاعتماد على البنية المعمارية النظيفة، إذ تعطي إمكانية فصل قواعد العمل وحالات الاستخدام عن الخياراتالتشغيلية، مما يعطي إمكانية استبدالها عند الحاجة. حيث أن هدف المشروع الأساسي هو إنشاء موقع ويب لتذاكر صيانة، لكن معمرور الوقت قد تظهر الحاجة لتغيير الويب لتطبيق جوال أو تطبيق مكتبي أو أي تقنية جديدة. كما سيتم الاعتماد على التصميم المقاد بالمجال لاحترام قواعد العمل بشكل صحيح وضمان تحقيقها للمتطلبات المقدّمة.

# 2.6- تنجيز التطبيق الخلفي Back-end

## 12.6. - هيكلية المجلدات العامة



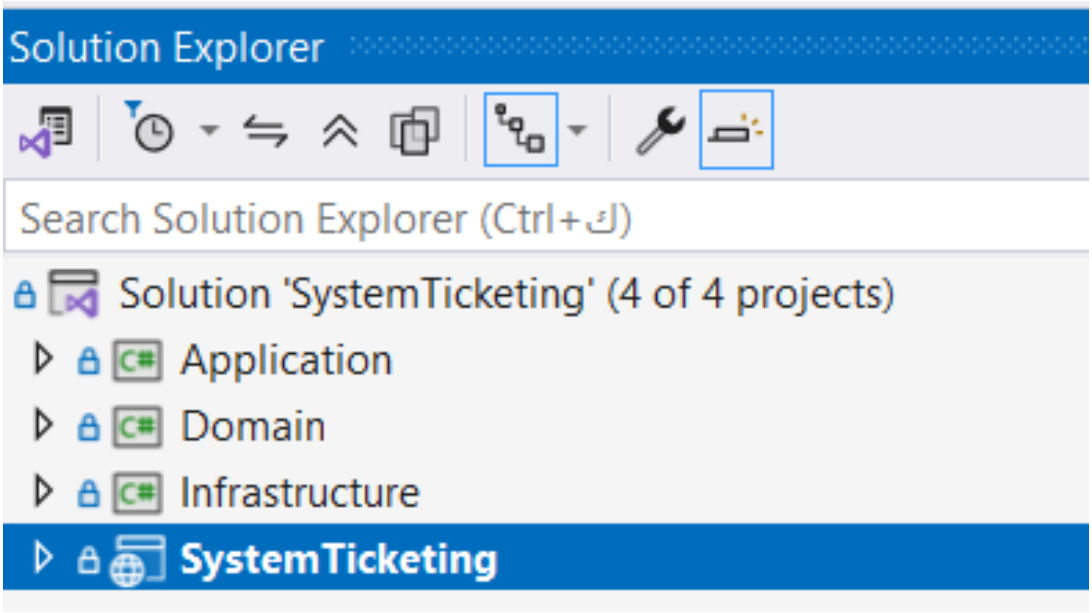

يوضح الشكل (91) هيكل ية المجلدات ضمن التطبيق الخلفي المتمثلة بما يلي :

- **عناصر الحل )Solution items(**: يحتوي عل ى البيانات الوصفية (metadata) الخاصّ ة بالمشروع، حيث يعتبر أحدأهم العناصر لكونه يفرض قيودا على المطورين العاملين على المشروع ويجنبهم ارتكاب الأخطاء.

**مثال**: يمكن إنشاء ملف ضمن مجلد عناصر الحلّ ينصّ عل ى أنّ جميع الواجهات (Interfaces) يجب أن تبدأ بحرف
(I) وإلا فإن المشروع لن يعمل، الأمر الذ ي يضمن وجود بيئة مشتركة بين جميع المطورين العاملين على المشروع.

- **الرماز المصدري (Source Code)**: يحوي التنجيز الفعلي للمشروع والذي سنتنطرق له لاحقا .
- **الاختبارات (Tests)**: تحوي الاختبارات التي تم إجراءها على النظام والتي سنتنطرق له ا لاحقا .

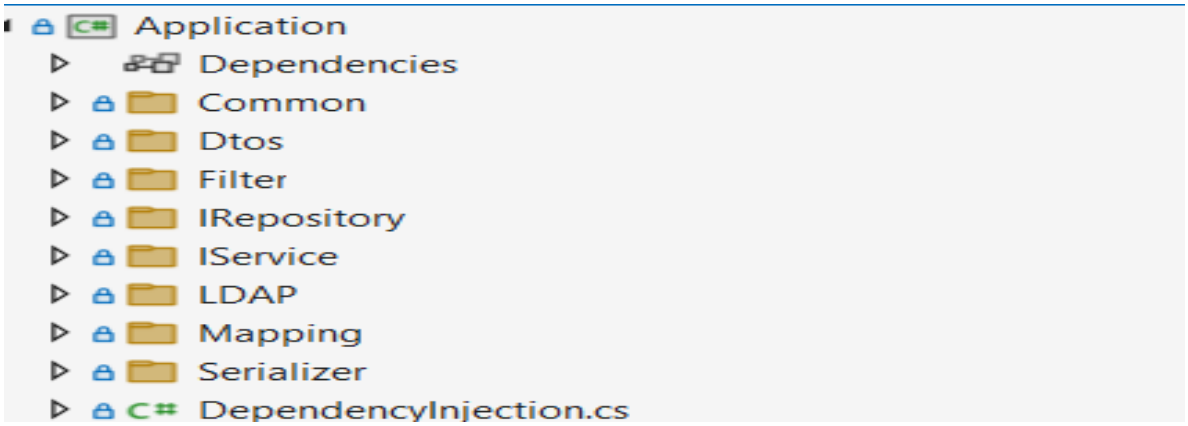
### 2.2.6 - هيكلية الرماز المصدري




يوضح الشكل (02) طبقات عمل التطبيق، حيث تم احترام منهجية البنية المعمارية النظيفة

### 5.2.6 - طبقة التطبيق Application Layer

طبقة التطبيق -يقابلها في البنية المعمارية النظيفة طبقة حالات الاستخدام (use case layer)- لا تملك أي تبعية (dependency) لأي طبقة أخرى غير طبقة المجال، ويتم فيها تنفيذ منطق عمل التطبيق وتنظيم التفاعلات بين المجالات الفرعيةالخمسة. لكل مجال مجموعة من حالات الاستخدام المنفصلة التي يمكن توضيحها في الشكل )29( .




مكونات طبقة التطبيق29الشكل

### 6.2.6 - طبقة البنية التحتيّة Infrastructure Layer

تصنف طبقة البنية التحت يّة ضمن الأطر والمحركات في الطبقة الخارج يّة للبنية المعمارية النظيفة، تملك هذه الطبقة تبعية (dependency) لطبقة التطبيق، حيث يتم فيه ا تنجيز المهام التشغيلية للنظام مثل تخزين البيانات ضمن قاعدة معطيات أو ضمن نظام ملفات أو غيرها من آليات التخزين.

يوضح الشكل (13) هرمية مجلدات طبقة البنية التحت ية.


الشكل 31 مكونات طبقة البنية التحت ي ة

توضح الفقرات التالية ما تم تنجيزه ضمن طبقة البنية التحت يّة.

### 6.12.6.- إدارة الأدوار

ضمن النظام المقترح يوجد ثلاثة أدوار أساسية:

- موظف عادي : إضافة تعديل تذكرة ومتابعة لها أي عرض تتبع تذكرته.
- مسؤول صيانة : إضافة تتبع وعرض كل التذاكر وإمكانية تصفية متقدمة للتذاكر .
- مدير النظام : تعيين مسؤولين الصيانة و عرض إحصاءات وتصفية التذكر وجمع التقارير.

### 2.6.2.6- المصادقة والتفويض (Authentication and Authorization)

تم تنجيز المصادقة من خلال استخدام رموز (**JWT**) التي تم شرح آلية عملها في قسم الأدوات المستخدمة، حيث أن لكل مستخدم دور أو عدّ ة أدوار ولكل دور سماح يّة (permission) واحدة أو أكثر.
يتم التفويض بعد ذلك من خلال التأكد من أن المستخدم يمتلك السماح يّة (permission) لطلب المعلومات من النظام ،حيث أن جميع الطلبات تمر عبر middleware تقوم بالتأكّ د من امتلاك المستخدم للسماح يّة المطلوبة.

هنالك أسلوبان للتفويض عند استخدام JWT:

- **الأسلوب الأول**: تحميل السماحيات ضمن رمز JWT ،وعند وصول طلب معين للنظام من قبل مستخدم مع ينيتأكّ د النظام من وجود السماح يّة المطلوبة ضمن رمز JWT.
- **الأسلوب الثاني**: تحميل رمز تعريف المستخدم فق ط ضمن رمز JWT، وعند وصول طلب معين للنظام من قبلمستخد م ما يتأكد النظام من وجود السماحية المطلوبة من خلال البحث عن سماح يّات الزبون ضمن قاعدة المعطيات. لكلا الأسلوبين السّابقين عدّة مشاكل، نذكر منها:

- **مشاكل استخدام الأسلوب الأول:**

إن وجود عدد كبير من السماحيات للمستخدمين سيؤدي إلى زيادة حجم رمز JWT، وبما أن هذا الرمز يُرسَ ل معكل طلب إلى النظام سيشكل ذلك عبئا على الشبكة. من ناحية أخرى، إذا تم إعطاء دور جديد لأحد المستخدمين وكانت مدة حياة رمز JWT كبيرة، سيؤد ي ذلك إلى عدم تعرف النظام على سماحيات المستخدم الجديدة لأن المعلومات ضمن الرمز لن يتم تحديثها حتى انتهاء مدة حياة الرمز.

- **مشاكل استخدام الأسلوب الثاني:**

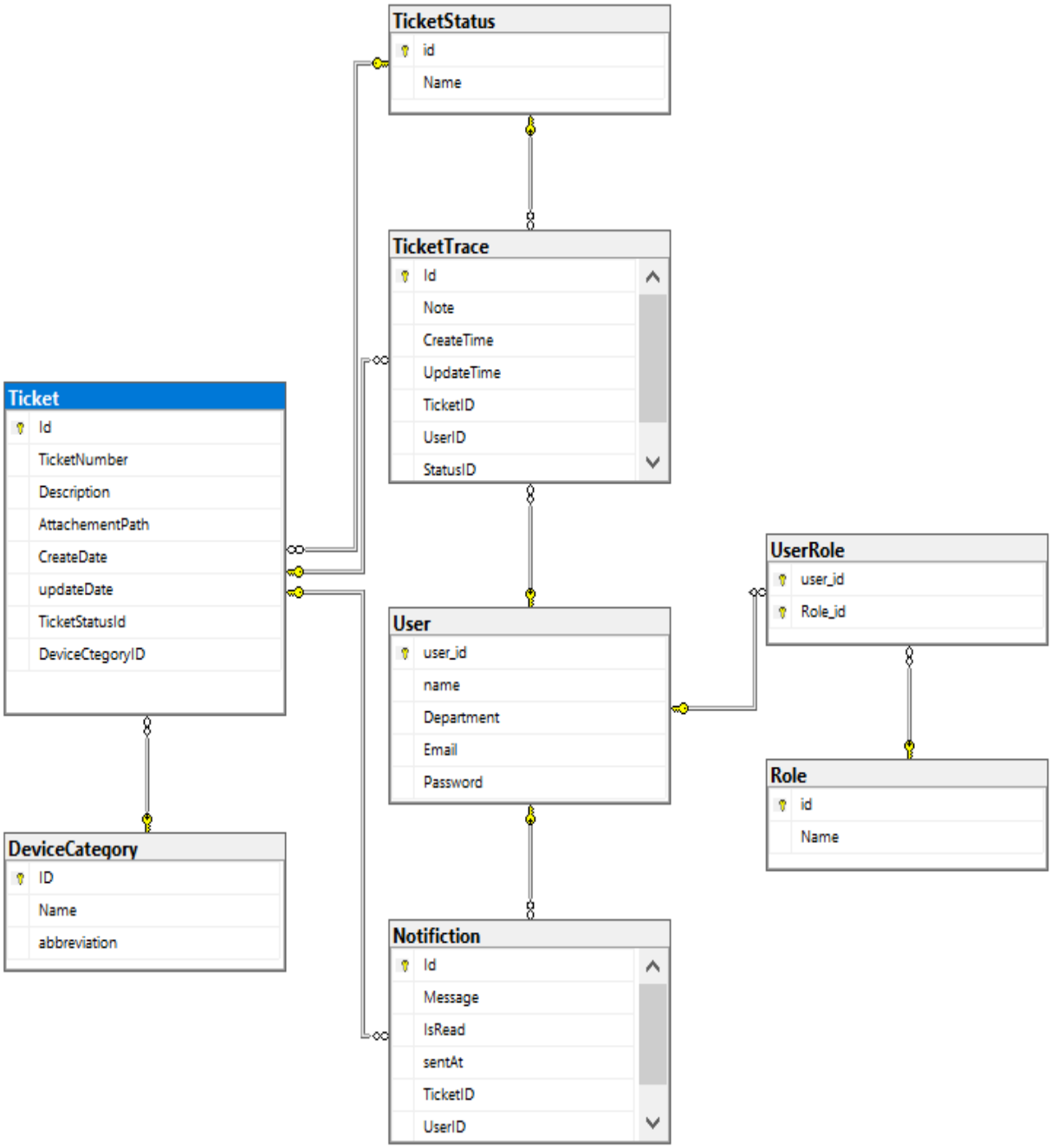
زيادة العبئ على قاعدة المعطيات وتأخير الرد على طلبات المستخدمين، حيث أن كل طلب يتضمّ ن التأكّ د منسماح يّات المستخدم يتطلب إرسال استعلام إلى قاعدة المعطيات .

تم اختيار الأسلوب الأول كأسلوب للتفويض لأن الأدوار ضمن نظام المطعم لا تتغير، وبالتال المشكلة الثانية لم تعد موجودة،ولحل المشكلة الأولى تم العمل على تقليل عدد السماح يّات للمستخدمين.

### .6.32- تخزين المعطيات

جر ى استخدام إطار العمل (Entity Framework) لتحويل الكيانات (Entities) مباشرة لجداو ل علائقية )Relational Tables( ضمن قاعدة معطيات من نوع (SQL-Server)، بالإضافةإلى تنجيز عقود المخازن (Repositories) الموجودة ضمن طبقة التطبيق.

**تم تحقيق التزامن لضمان إ تّساق البيانات المخ زّنة ضمن قاعدة المعطيات من خلا ل استخدام منهجية القفل المتفائل ،وذلك لأنعدد مستخدمي التطبيق قليل نسبي ا لذا سيكون احتمال حدوث التضارب صغيرا ، وحتّى لمراقبة حدوث مثل هذه التضاربات بعد رفع (logging)نتأكد من هذا الافتراض يمكننا استخدام التسجيل ضمن جداول قاعدة المعطيات، حيث أن (row version)التطبيق في مرحلة المنتج د م عمود من نم ط تحتوي على عداد يتم زيادته لكل عملية إدراج أو تحديث يتم إجراؤها (SQL-Server)قاعدة المعطيات . لتنجيز (production) داخل قاعدة البيانات. (row version)على جدول يحتوي على عمود من نمط القفل المتفائل أُستخ**




الشكل 32 مخطط العلاقات ضمن قاعدة المعطيات

### 6.52.- الإشعارات

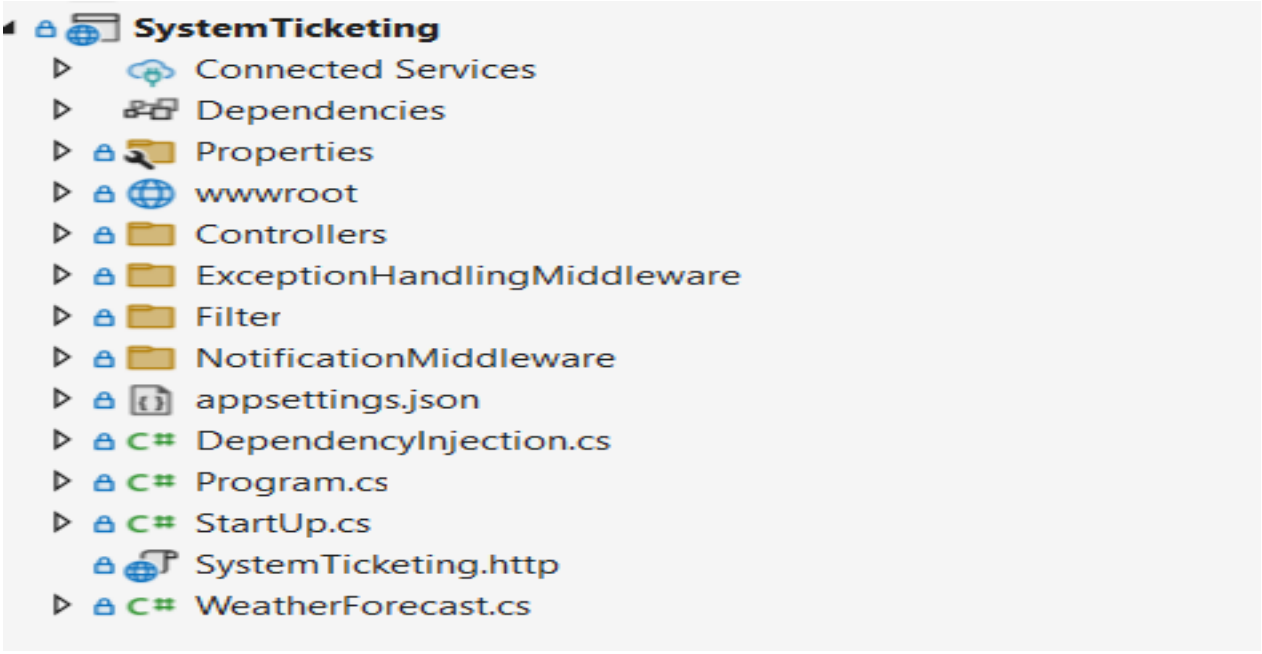
استخ دمت مكتبة SignalR لتسهيل إيجاد (websocket) بين المستخدمين من جهة والمخدم من جهة أخرى، من أجل إرسال إشعارات لكل من الموظفين ومسؤلين الصيانة.

## Domain Layer


**الشكل 32 يمثل طبقة المجال**

هي القلب النابض للتطبيق في الهندسة النظيفة، حيث تحتوي على القواعد الأساسية والمفاهيم المركزية لمجال عملك (مثل العملاء والمنتجات). تتميز بأنها مستقلة تمامًا عن أي تقنيات خارجية (مثل قواعد البيانات أو واجهات المستخدم)، مما يجعلها سهلة الاختبار والتطوير. تضم الكيانات الرئيسية **(**Entities**)**، القواعد التجارية (Business Rules)، والأحداث المهمة (Domain Events) مع التركيز على السلوكيات بدلًا من التفاصيل التقنية.

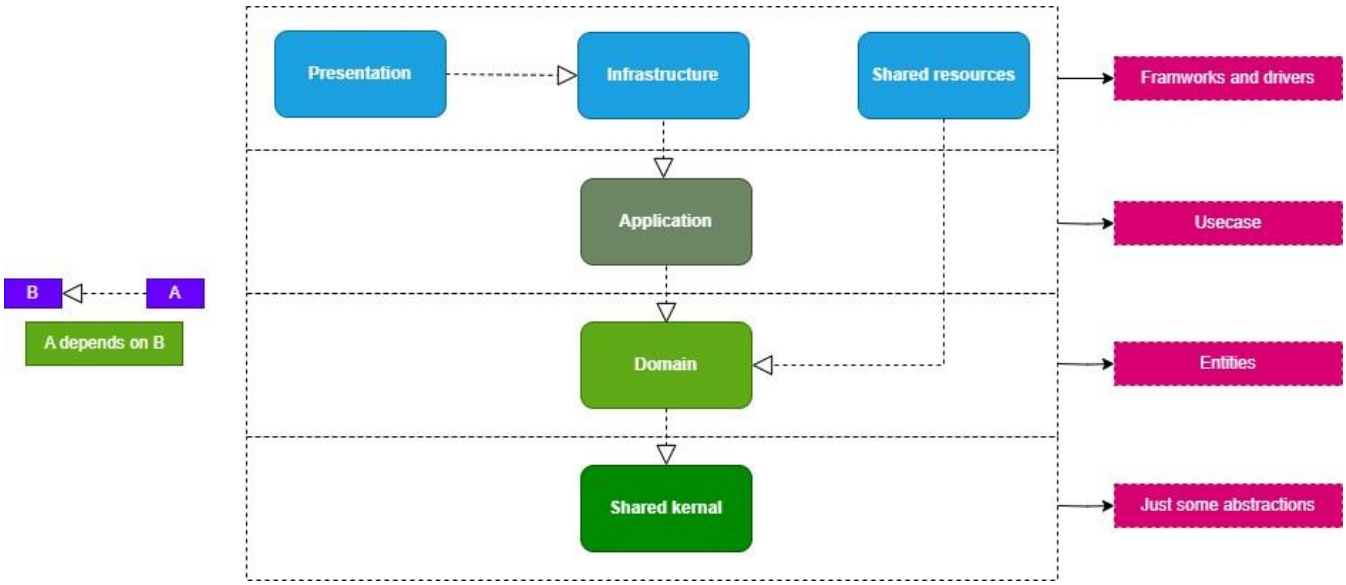
**Presentation Layer**




**الشكل 33 يمثل طبقة العرض**

تصنف هذه الطبقة ضمن الأطر والمحركات في الطبقة الخارجية للبنية المعمارية النظيفة، تملك هذه الطبقة تبعية (dependency) لجميع طبقات الحل، حيث تعدّ المشغل الأساسي للتطبيق، تقوم باستقبال طلبات المستخدم وقف برتوكول HTTP، وتتأكّ د من صلاحياته ثم ترسل الطلب لطبقة التطبيق (Application)، وتعيد الجواب القادم من هذه الطبقة.

## 9.2. مخطط الاعتماديات Dependencies



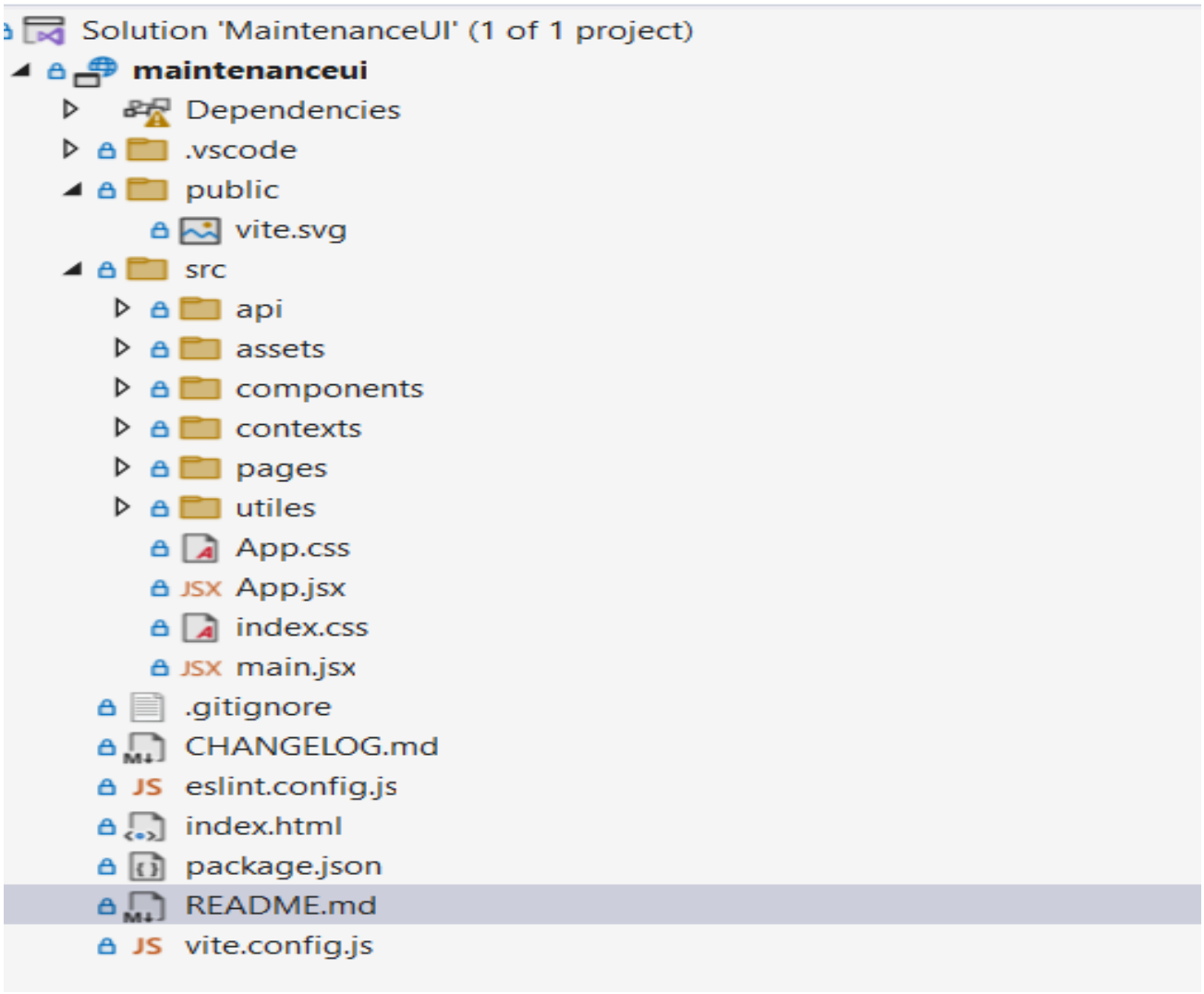

الشكل 35 اعتماديات طبقات النظا م

# 3- تنجيز الواجهات الأمامية Front-end

## 1.3.6 - منهجية العمل

استخ دمت المكتبة React لتنجيز الواجهات الأمامية باستخدام منهجية تطبيق الصفحة الواحدة( single pageapplication) مع تقسيم التطبيق لمجموعة من المكونات (components). يبين الشكل مجلدات العمل.

من ناحية أخرى، استخدمت لغة )typescript( التي تساعد على اكتشاف الأخطاء خلال مرحلة الترجمة (compile time).

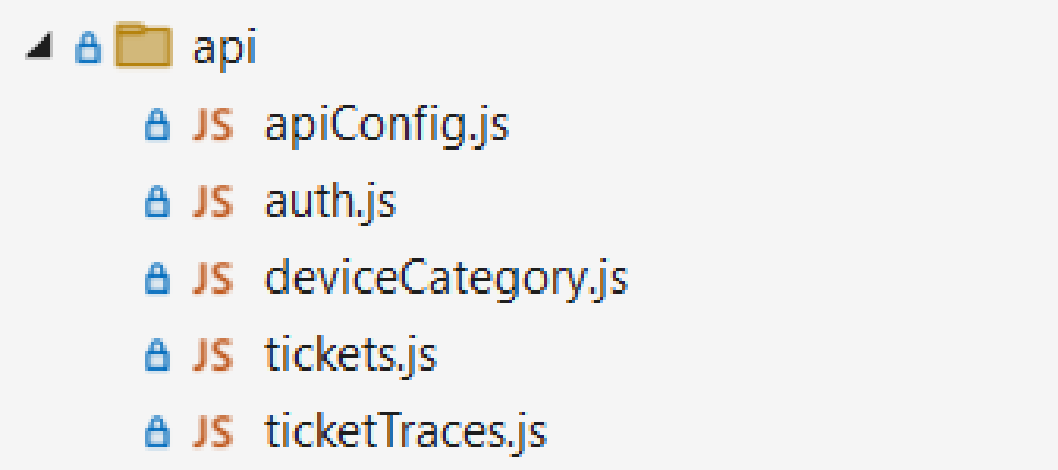



الشكل 36 مكونات طبقة الواجهات

## 2.3. - معالجة الاتصال مع التطبيق الخلف ي

### 12.3.6.- تحقيق الاتصال

جرى استخدام مكتبة axios لتحقيق الاتصال مع التطبيق الخلفي و إرسال طلبات HTTP، لكن ماذا لو أردنا تغيير هذه المكتبة هل نحن بحاجة لتعديل جميع المكونات ضمن التطبيق؟
بالطبع لا، حيث تم تحقيق الفصل بين المكونات ومكتبة axios من خلال إنشاء الصف (ClientContext) المسؤول عن التعامل مع طلبات HTTP، والذي يقوم باستقبال الطلبات من المكونات وإر سالها باستخدام مكتبة axios .




الشكل 37 الصف المسؤول عن الاتصال مع التطبيق الخلفي

### 22.3.6.- معالجة الأخطاء

لمعالجة الأخطاء القادمة من قبل التطبيق الخلفي ضمن الصف (ClientContext) قمنا بإنشاء معترض للطلبات
interceptor يقوم بمعالجة الأخطاء قبل وصولها للواجهة، مما يضمن احترام مبدأ عدم التكرار

.(don’t repeat yourself)

```
{
  "name": "staticsataui",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "lint": "eslint .",
    "preview": "vite preview"
  },
  "dependencies": {
    "@fortawesome/free-solid-svg-icons": "^7.0.0",
    "@fortawesome/react-fontawesome": "^0.2.3",
    "antd": "^5.26.7",
    "autoprefixer": "^10.4.21",
    "axios": "^1.11.0",
    "chart.js": "^4.5.0",
    "date-fns": "^4.1.0",
    "dayjs": "^1.11.13",
    "framer-motion": "^12.23.12",
    "postcss": "^8.5.6",
    "react": "^18.3.1",
    "react-chartjs-2": "^5.3.0",
    "react-datepicker": "^8.4.0",
    "react-dom": "^18.3.1",
    "react-icons": "^5.5.0",
    "react-router-dom": "^6.30.1",
    "tailwindcss": "^4.1.11"
  },
  "devDependencies": {
    "@eslint/js": "^9.30.1",
    "@types/react": "^19.1.8",
    "@types/react-dom": "^19.1.6",
    "@vitejs/plugin-react": "^4.7.0",
    "eslint": "^9.32.0",
    "eslint-plugin-react": "^7.37.5",
    "eslint-plugin-react-hooks": "^5.2.0",
    "eslint-plugin-react-refresh": "^0.4.20",
    "globals": "^16.3.0",
    "vite": "^7.0.6"
  }
}
```


**شكل 34- الحزم المستخدمة في موقع الويب**

**يحوي الجدول التالي شرحاً مقتضباً عن معظم هذه الحزم السابقة:**

| | |
|---|---|
| styled-components | مكتبة تسمح لك بكتابة تعليمات برمجية css مباشرة داخل مكونات) React Component(، تساعد على إنشاء تنسيقات و أنماط قابلة لإعادة الإستخدام ومغلفة encapsulated للمكونات Component |
| axios | تسنخدم في إنشاء طلبات http للتخاطب مع المخ دم |
| chart.js | تستخدم في إنشاء مخططات تفاعلية و قابلة للتخصيص customizable في تطبيقات الويب |
| PusherJs | مكتبة لتشغيل و تتبع الأحداث Events وتؤمن التواصل في الزمن الحقيقي -real time و ذلك WebSockets. استنادا الى |
| react-speech-kit | المكتبة التي استفدنا منها في تطبيق - speechto-text تحويل الكلام الى نص، وأيضا توفر ميزة -textto- speech تحويل النص الى كلام |
| react-beautiful-dnd | تستخدم في تطبيق السحب والإفلات Drag and Drop، وتم الإستفادة منه في تطبيقنا في سؤال إعادة الترتيب. |
| react-router-dom | توفر مجموعة من المكونات التي تمكننا من تنفيذ التوجيه Routing والتنقل بين الصفحات ضمن تطبيقنا. |
| react-icons | يحوي بعض الأيقونات المدمجة ضمن ال React |

**جدول 14- شرح عن الحزم المستخدمة في موقع الويب**

## 3.2.3.6 - المصادقة مع التطبيق الخلفي

### 12.1.4- معايير تصميمية متبعة ضمن النظام

- النظام مطابق للمتطلبات الوظيفية وغير الوظيفية .
- النظام قابل للتوسع بدرجة كبيرة و يتمتع بدرجة كبيرة من العمومية ،بحيث يمكن إضافة خصائص لصفوف أخرى بأقل تعديلات ممكنة.
- النظام غني بالوظائف والتقنيات الحديثة اللازمة لجذب الموظفين لاستخدامه .

## 2.4- أدوات مساعدة أخرى

### Visual Studio -.12.4

يُعرف بأن بيئة تطوير متميزة مجاني مفتوح المصدر تابع لشركة مايكروسوفت ويت وفر لجميع أنظمة تشغيل الحاسوب ومن مزاياه تم اختياره بسبب وجود كميات كبيرة ومُفيدة من الإضافات التي يُمكن تنزيلها، وبالتالي من الممكن أن تسهل عملية التطوير البرمجي.

يدعم Visual Studio الكثير من اللغات مثل HTML،CSS ، Type Script،JavaScript ولغات أخرى عديدة.

تم إستعمال Visual Studio لكتابة كافة الأكواد البرمجية في الواجهتين الأمامية للويب والخلفية )Backend( نظراً و مرونته و توفيره العديد من المزايا التي تجعل عمل المبرمج أكثر سهولة،

بالإضافة إلى تقديمه خدمات إكتشاف الأخطاء Debugging للمساعدة في معرفة مصدر الخطأ وإصلاحه .[10]

## Postman -.52.4

Postman هي منصة تعاون شائعة لتطوير واختبار وتوثيق APIs )واجهات برمجة التطبيقات(. يوفر واجهة سهلة الاستخدام لتقديم طلبات HTTP وتنظيم API endpoints وإدارة تدفقات العمل المتعلقة بتطوير واختبار API.

باستخدام Postman ، يمكن إنشاء طلبات وإرسالها إلى واجهات برمجة التطبيقات باستخدام طرق HTTP متنوعة مثل GET و POST و PUT و DELETE وما إلى ذلك. فهي تتيح لك إضافة رؤوس الطلبات )headers( و موسطات الطلب)parameters( وجسم الطلب)body( وتفاصيل المصادقة إلى طلبات واجهة برمجة التطبيقات.

يتضمن Postman أيضًا ميزات أخرى مثل الاختبار الآلي ، والذي يم كن من كتابة نصوص اختبار للتحقق من صحة الردود من واجهات برمجة التطبيقات ، والقدرة على إنشاء وإدارة وثائق APIللمساعدة على فهم واجهات برمجة التطبيقات. [13]

# الواجهات و الإختبار

سنورد في هذا الفصل الهوية البصرية للمشروع، وواجهاته و شرح مبسط للواجهات
إضافة الى إختبارات أجريت على النظام

الفصل السابع

# الخاتمة

بعد الانتهاء من تطوير نظام إدارة تذاكر الصيانة لمعهدنا العالي، نجحنا في تحقيق الأهداف الأساسية للمشروع عبر بناء تطبيق ويب متكامل يدمج تقنيات هندسة البرمجيات الحديثة مع احتياجات الصيانة العملية.

يمثل هذا النظام نقلة نوعية في إدارة أصول المعهد، حيث يحوّل عمليات الصيانة من إجراءات روتينية إلى نظام ذكي قائم على البيانات. نوصي بشدة بنشره على خوادم المعهد، والبناء على إطاره القوي لتحقيق المزيد من الابتكارات.

أثناء رحلتنا في تصميم وبناء النظام واجهتنا العديد من المشاكل والتحديات أهمها كان استخدام مجموعة أدوات جديدة بشكل كامل مما يؤدى الى ضياع وقت في استدراك وتعلم هذه الأدوات وطرق التعامل معها.

لم ادخر جهدا إلا ووضعته في سبيل إتمام هذا العمل عسى أن يتم الإستفادة منه أو تحديثه أو تطويره ليصبح منتج إقتصادي يمكن الاستفادة منه فيما بعد.

.

# المراجع


[1] https://dotnet.microsoft.com

[2] "React - A JavaScript library for building user interfaces". Documentation. Retrieved from https://reactjs.org/docs ↗

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

[4] "What is a REST API?" https://www.redhat.com/en/topics/api/what-is-a-rest-api.

[5] "Cloudinary - The media experience platform for developers". Documentation. Retrieved from https://cloudinary.com/documentation ↗

[6] "Visual Studio Code - Code Editing. Redefined." Documentation. Retrieved from https://code.visualstudio.com/docs ↗

[7] Postman - API Development Environment." Documentation. Retrieved from https://learning.postman.com/docs ↗

[8] https://learn.microsoft.com/en-us/sql/sql-server/?view=sql-server-ver17

[9] https://learn.microsoft.com/en-us/aspnet/core/fundamentals/configuration/?view=aspnetcore-9.0

[10] Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). Design patterns: elements of reusable object-oriented software. Pearson Deutschland GmbH.

[11] Menascé, D. A., & Nakanishi, T. (1982). Optimistic versus pessimistic concurrency control mechanisms in database management systems. Information systems, 7(1), 13-27.