جامعة الوادي

جامعة الوادي
كلية العلوم الدقيقة
تخصص نظم المعلوماتية

اكاديمي ليسانس

مذكرة مشروع نهاية العام

منصة برمجية لتسيير وإدارة الحسابات في برامج الويب

من اعداد :

تليلي زكرياء

تحت اشراف :

الدكتور يعقوب محمد امين

السنة الدراسية : 2025/2024

الملخص

يهدف مشروعي إلى تطوير منصة برمجية متكاملة لتسيير وإدارة الحسابات بطريقة آمنة وفعّالة، وذلك عبر إنشاء إطار عمل برمجي (Framework) يُسهل عملية التحكم في الحسابات، إدارة الصلاحيات، وتنظيم الوصول إلى الموارد.

تم اعتماد Angular لتصميم واجهة المستخدم الديناميكية وSpring Boot لتطوير الواجهة الخلفية والخدمات، مع دمج خادم التوثيق Keycloak القوي جدا في الامان لتأمين النظام وتوفير إدارة متقدمة للمستخدمين والأدوار.

قمت في البداية بدراسة معمّقة لمتطلبات الأنظمة المعتمدة في تسيير الحسابات وتحليل أبرز النقائص والتحديات التي تواجهها، مما ساعدني على تحديد الوظائف الأساسية للمنصة. كما استخدمت لغة UML لتوضيح وشرح طريقة عمل هذا الاطار البرمجي.

المنصة المقترحة تُمكن المؤسسات من التحكم الكامل في بيانات المستخدمين، ضبط الصلاحيات بدقة، وتكامل مرن مع أنظمة خارجية، مما يجعلها أداة قوية لبناء تطبيقات آمنة وقابلة للتوسع.

الكلمات المفتاحية : Angular , Spring Boot , Keycloak , الأمان , منصة برمجية لتسيير وإدارة الحسابات , الأدوار.

Abstract

My project aims to develop an integrated software platform for managing and administering accounts in a secure and efficient way, by creating a software framework that simplifies account control, permission management, and resource access organization.
I adopted Angular for designing the dynamic user interface and Spring Boot for developing the backend and services, while integrating the highly secure Keycloak authentication server to secure the system and provide advanced user and role management.
I began with an in-depth study of the requirements of existing account management systems and analyzed the main shortcomings and challenges they face, which helped me define the platform's core functionalities. I also used UML diagrams to illustrate and explain how this framework works.
The proposed platform enables organizations to have full control over user data, precisely configure permissions, and flexibly integrate with external systems, making it a powerful tool for building secure and scalable applications.

Keywords: Angular, Spring Boot, Keycloak, Security, Account Management Software Platform, Roles.

شكر وتقدير

اشكر الله العلي القدير الذي أنعم علي بنعمة العقل والدين، القائل في محكم تنزيله:
﴿ وَفَوْقَ كُلِّ ذِي عِلْمٍ عَلِيمٌ ﴾ (سورة يوسف، آية 76)
وقال رسول الله صلى الله عليه وسلم:

"من صنع إليكم معروفًا فكافئوه، فإن لم تجدوا ما تكافئونه به فادعوا له حتى تروا أنكم قد كافأتموه"
رواه أبو داوود.

وتقديرًا واعترافًا مني بالجميل، اتقدم بجزيل الشكر والعرفان لأولئك الأساتذة المخلصين الذين لم يألوا جهدًا في دعمي ومساندتي في مجال البحث العلمي، واخص بالذكر الأستاذ الفاضل: الدكتور يعقوب محمد أمين، لما بذله من جهد كبير في توجيهي ومساعدتي في تجميع المادة البحثية، فجزاه الله عني كل خير.

وأخيرًا، اتقدم بخالص شكري وامتناني إلى كل من قدم لي يد العون والمساعدة لإخراج هذه الرسالة في أفضل صورة ممكنة

الاهداء

من رياحين الكلام و ياسمين الجمل وشقائق الاحرف ، اقدم لكم جزيل الشكر والعرفان،
إلى التي حملتني تسعة أشهر وتحملت عناء تربيتي حتى وصلت إلى ما انا عليه اليوم،
إلى الأم الغالية.

إلى من أضاء شموع دربنا ، إلى من حمل عبء حياتي ، إلى الأب الحنون.

الى اخوتي الاحباء ،
إلى رفقاء الحياة

الطالب :
تليلي زكرياء

فهرس المحتويات

فهرس الاشكال

مقدمة عامة

أصبح الاعتماد على الإعلام الآلي أمرًا لا غنى عنه في مختلف مجالات الحياة الاجتماعية، الاقتصادية، التقنية، وحتى الإدارية، حيث يساهم بشكل كبير في تحسين الأداء وتسهيل التسيير اليومي للوظائف والخدمات. وفي هذا الإطار، تلعب الحلول البرمجية الحديثة دورًا محوريًا في تطوير اساليب **تسيير وإدارة الحسابات**.
تسعى هذه المذكرة إلى دراسة **إمكانية إنشاء منصة برمجية (إطار عمل/إطار برمجي)** مخصصة لتسيير وإدارة الحسابات، وذلك عبر توفير أدوات ذكية وآلية لمعالجة البيانات، تتبع العمليات، وتوليد التقارير بشكل دقيق وسلس. تهدف المنصة إلى تعويض الأساليب اليدوية التقليدية، التي غالبًا ما تكون عرضة للأخطاء وتستهلك وقتًا وجهدًا كبيرًا، بنظام رقمي فعال وآمن.
تتمثل أهمية هذا المشروع في تقديم حل برمجي مرن وقابل للتخصيص، يمكن اعتماده من قبل مؤسسات مختلفة لتنظيم حساباتها وفقًا لاحتياجاتها الخاصة، مع مراعاة الجوانب التقنية، الأمنية، والقانونية. كما يتيح هذا الإطار إمكانية التكامل مع منصات أخرى وخدمات الدفع الإلكتروني وإدارة المستخدمين والصلاحيات.
وبناءً عليه، نحاول من خلال هذا العمل الخروج بنموذج برمجي موحد يُمكّن المؤسسات من إدارة حساباتها بكفاءة أكبر، وتقديم تجربة استخدام احترافية تدعم متطلبات التطوير المستقبلي.

فكيف يمكننا الخروج بنموذج جديد يقوم بحل جميع مشاكل الأنظمة القديمة التقليدية ؟

سأحاول خلال هذا العمل تصميم وتطوير اطار برمجي يساعد في تسيير وإدارة الحسابات والمستخدمين بغرض تعزيز الأمان في الأنظمة وتحديد المهام الخاصة بكل مستخدم والصلاحيات الخاصة به .

تنقسم الدراسة الى ثلاثة فصول رئيسية :

الفصل الأول : الدراسة التمهيدية
نعرض فيه مجال المشروع، ونقوم بتوصيف الوضع الحالي وأساليب إدارة الحسابات التقليدية، مع تسليط الضوء على أبرز الإشكالات والصعوبات التي تواجه المؤسسات

الفصل الثاني : التصميم
نقدم فيه تصورًا دقيقًا للنظام المراد تطويره باستخدام أدوات التحليل والتصميم(UML) ، من خلال نماذج مثل Use Case Diagram، Class Diagramوغيرها، لوصف مكونات النظام وعلاقاته

الفصل الثالث : الدراسة التفصيلية
نعرض فيه الأدوات والتقنيات المعتمدة في بناء الإطار البرمجي، مع تقديم بعض الواجهات التوضيحية التي تبرز الخصائص والمميزات الأساسية للمنصة

الفصل الأول : الدراسة التمهيدية

مقدمة :

مع التقدم السريع في تقنيات المعلومات والاتصالات، أصبحت أنظمة إدارة حسابات المستخدمين جزءًا أساسيًا في أي تطبيق أو خدمة رقمية. ومع زيادة الحاجة إلى التحكم الدقيق في صلاحيات الوصول وحماية البيانات الحساسة، أصبح من الضروري تطوير أنظمة موثوقة تجمع بين سهولة الإدارة وأعلى معايير الأمان. في هذا السياق، تعتبر تطبيقات الويب الحديثة خيارًا مثاليًا بفضل مرونتها وإمكانية الوصول إليها عبر مختلف الأجهزة والمنصات.

لكن، يطرح إنشاء نظام لإدارة حسابات المستخدمين عدة تحديات، أبرزها حماية المعلومات الشخصية من جهة، وتوفير إدارة مرنة وفعّالة للمستخدمين والأدوار من جهة أخرى. كما أن التهديدات الأمنية المستمرة تستدعي حلولًا مبتكرة قادرة على التصدي للهجمات الإلكترونية وضمان خصوصية المستخدمين.

بناءً على هذه التحديات، تهدف هذه المذكرة إلى دراسة وتحليل متطلبات تطوير تطبيق ويب لإدارة حسابات المستخدمين، مع التركيز على الجوانب الأمنية وإدارة الحسابات بشكل ذكي. فكيف يمكننا تصميم نموذج جديد يعالج مشاكل الإدارة والأمان بشكل فعّال؟

1.1 تعريف تسيير حسابات المستخدمين في الويب

تسيير المستخدمين (User Management) هو الإطار المؤسسي والتقني لإدارة الهويات الرقمية للمستخدمين، ويتضمن إنشاء الحسابات وتعديلها وحذفها، بالإضافة إلى التحكم في الصلاحيات وتوثيق الوصول. تواجه المؤسسات تحديات عدة مثل تشتت الهويات عبر أنظمة متعددة (identity sprawl) وإدارة دورة حياة المستخدم (Provisioning/Deprovisioning) والتكامل مع البنى التحتية الهجينة (onprem & cloud)، إضافةً إلى متطلبات الامتثال وتوليد التقارير وتطبيق معايير الأمان الحديثة. من الحلول المتاحة: أنظمة إدارة الهوية والوصول (IAM) الموحدة، المصادقة متعددة العوامل (MFA)، التسجيل الأحادي (SSO)، أتمتة دورات حياة الحسابات، ونماذج التحكم في الوصول كالتحكم بالدور (RBAC) أو بالسمات (ABAC)، إلى جانب أدوات سوقية مثل Okta وJumpCloud و AWS IAM Identity Center، مع الاعتماد على معايير NIST وOWASP للتوجيه.

يتضمن تسيير المستخدمين عدة مهام رئيسية: إدارة الحسابات (إنشاء وتعديل وحذف)، المصادقة، التوثيق، التحكم في الصلاحيات، وتوفير سجلات التتبع (audit logs) لمراقبة النشاطات.

تختص إدارية تكنولوجيا المعلومات بضبط الهوية والامتيازات للمستخدمين وفقًا للأدوار الوظيفية أو السياسات المؤسسية، مما يحقق الأمان والكفاءة التشغيلية.

1.2 أهداف ومهام نظام الحسابات

نظام الحسابات يؤدي عدة وظائف رئيسية لضمان أمان، فعالية، وسلاسة استخدام المنصة:

توثيق الهوية (**Authentication**):
التأكد من أن الشخص الذي يحاول الدخول هو فعلاً من يدعي أنه هو، عبر التحقق من بيانات الاعتماد (مثلاً البريد الإلكتروني وكلمة المرور).

التحكم في الوصول (**Authorization**):
تحديد ماذا يمكن للمستخدم القيام به داخل النظام، ومنع الوصول إلى الموارد غير المصرح بها.

إدارة الجلسات (**Session Management**):

إنشاء وتتبع الجلسات النشطة للمستخدمين، مع ضمان انتهائها بشكل صحيح لحماية الحسابات.

حماية البيانات:

تأمين المعلومات الشخصية والحساسة ضد السرقة أو التلاعب، سواء أثناء النقل أو التخزين.

تحقيق تجربة مستخدم مخصصة:

عبر تسيير الحسابات، يمكن تقديم محتوى وخدمات مخصصة لكل مستخدم بناءً على بياناته واهتماماته.

1.3 مكونات إدارة الحسابات

تتكون إدارة الحسابات من عدة عناصر أساسية:

أ) التسجيل (**Registration**):

إنشاء حساب جديد عبر إدخال معلومات ضرورية كالبريد الإلكتروني وكلمة المرور وأحيانًا معلومات إضافية.

ب) تسجيل الدخول (**Login**):

عملية دخول المستخدم إلى حسابه عبر توفير بيانات الاعتماد الصحيحة.

ج) إدارة الجلسات:

إنشاء جلسة بمجرد نجاح عملية تسجيل الدخول، وإدارتها إلى حين خروج المستخدم أو انتهاء صلاحية الجلسة.

د) إعادة تعيين كلمة المرور:

آلية لاستعادة الوصول إلى الحساب في حال نسيان كلمة المرور عبر خطوات تحقق إضافية.

هـ) تغيير معلومات الحساب (**Profile Update**):

تمكين المستخدم من تحديث معلوماته الشخصية مثل الاسم، البريد الإلكتروني، أو كلمة المرور مع اتخاذ إجراءات تحقق مناسبة لضمان أمان البيانات.

و) التحقق من الهوية (**Email/SMS Verification**):

تأكيد صحة بيانات الاتصال عبر إرسال رمز تحقق أو رابط تنشيط.

1.4 إدارة صلاحيات المستخدمين (User Roles and Permissions)

في بيئات الويب، ليس كل المستخدمين متساويين في صلاحياتهم. لهذا السبب يعتمد معظم المطورين على أنظمة التحكم في الوصول.

نموذج التحكم المعتمد على الدور (**RBAC**):

حيث يتم تحديد مجموعة أدوار (Roles) مثل:

زائر (**Guest**)

مستخدم عادي (**User**)

مشرف (**Admin**)

مدير ((**Manager**)

وترتبط كل دور بمجموعة صلاحيات محددة مثل القراءة، الإضافة، التعديل، الحذف.

نموذج التحكم المعتمد على الأذونات (**Permission-Based Access**):

حيث يُعطى كل مستخدم صلاحيات دقيقة بدون التقيد بدور ثابت.

هذا النظام يُسهم في تعزيز الأمان وتقليل مخاطر الوصول غير المصرح به للموارد.

1.5 واجهات برمجة التطبيقات الخاصة بالمستخدمين (User API)

مع اعتماد نموذج العمارة الموزعة (Frontend منفصل عن Backend)، أصبح من الضروري وجود مجموعة من الواجهات البرمجية (APIs) لإدارة الحسابات:

أمثلة على واجهات API:

POST /api/register لإنشاء حساب جديد

POST /api/login للدخول

GET /api/user/profile لاسترجاع معلومات الحساب

PUT /api/user/update لتحديث بيانات المستخدم

تأمين الـ **APIs**:

يجب حماية هذه الواجهات من هجمات مثل:

هجوم **CSRF (Cross-Site Request Forgery)** :

هجوم يحاول فيه المهاجم خداع المستخدم المسجّل الدخول لإرسال طلب غير مرغوب فيه إلى موقع ويب (مثل تحويل أموال أو تغيير كلمة المرور) دون علمه، مستغلًا صلاحياته (تزوير طلب عبر المواقع).

هجوم **Injection** بأنواعه :

مجموعة من الهجمات حيث يقوم المهاجم بإرسال بيانات خبيثة إلى البرنامج (مثل قواعد البيانات أو أنظمة أخرى) لجعله ينفذ أوامر غير مقصودة. من أشهر أنواعها: SQL Injection (حقن قواعد البيانات) و Command Injection (حقن أوامر النظام).

هجمات كسر المصادقة (**Broken Authentication**) :

وهو يعني أن آليات التحقق من الهوية (تسجيل الدخول) تكون ضعيفة أو مصممة بطريقة سيئة، مما يسمح للمهاجمين بالتحكم في حسابات الآخرين أو الدخول بدون تسجيل صحيح.

1.6 معايير الأمان العالمية المتعلقة بإدارة الحسابات

لتأمين واجهات المستخدم API، ينبغي الالتزام بما يلي:

-استخدام رموز تحقق (Tokens) لتوثيق الجلسات.

-فرض استخدام HTTPS لتشفير البيانات أثناء الإرسال.

-تطبيق تحقق متعدد العوامل (MFA) لتعزيز الأمان.

-الحد من عدد محاولات الدخول لتفادي هجمات القوة الغاشمة.

-تسجيل ومتابعة النشاطات باستخدام سجلات التدقيق (Audit Logs).

1.7 التحقق المتقدم للمستخدم (Advanced Authentication)

لزيادة مستوى الأمان، ينصح باستخدام تقنيات تحقق إضافية:

المصادقة الثنائية (**FA**2):

طلب خطوة تحقق إضافية بعد كلمة المرور، كإدخال رمز يصل إلى الهاتف.

OTP عبر البريد أو **SMS**:

إرسال رمز تحقق مؤقت لتأكيد الهوية.

تطبيقات المصادقة:

مثل Google Authenticator لتوليد رموز تحقق محلية.

التوثيق البيومتري:

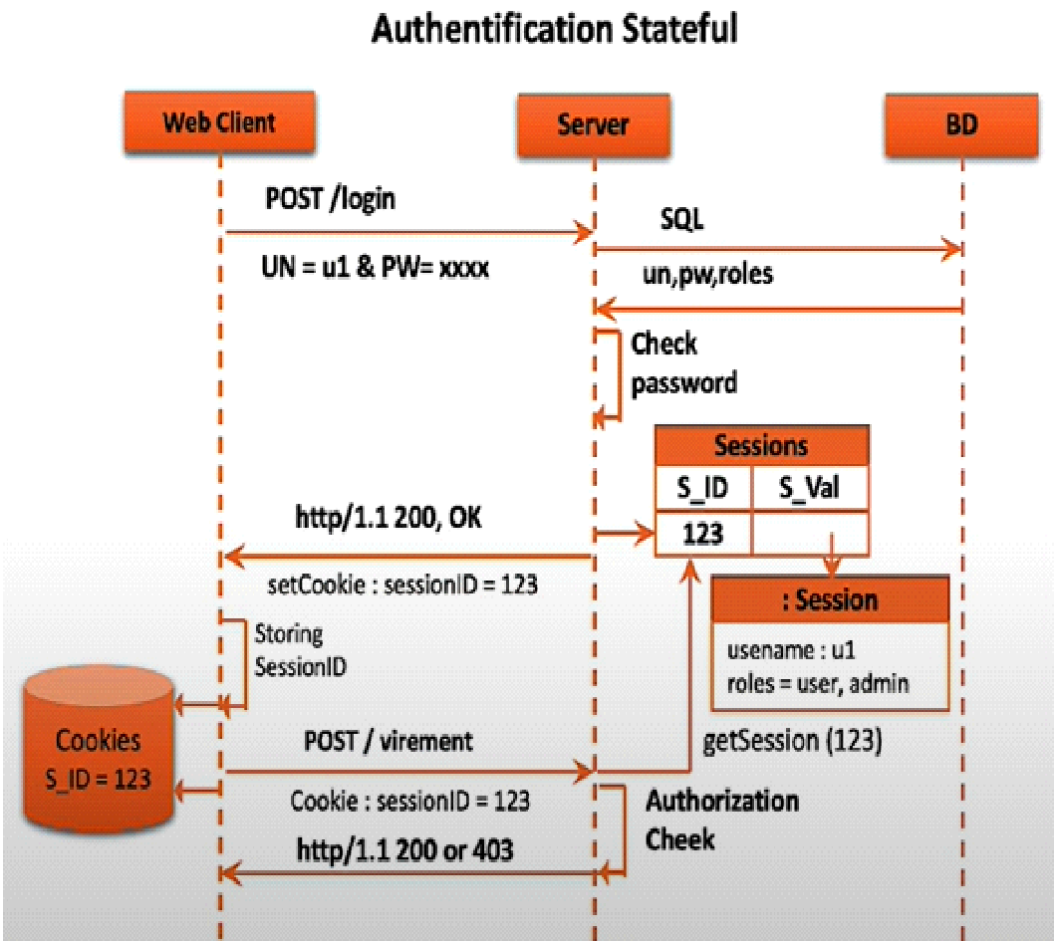
مثل بصمة الإصبع أو التعرف على الوجه، عند توفر الأجهزة الداعمة.

1.8 أنماط المصادقة في أنظمة تسيير المستخدمين

في سياق تسيير المستخدمين، تلعب المصادقة دورًا محوريًا في ضمان الوصول الآمن للموارد. وهناك نمطان رئيسيان لأنظمة المصادقة في البنى التحتية الحديثة: المصادقة Stateful والمصادقة Stateless، ويعتمد كل منهما على آلية مختلفة لتخزين وتتبع الجلسات.

- المصادقة **Stateful**:

تعتمد على تخزين معلومات الجلسة في جهة الخادم. عند نجاح تسجيل الدخول، يقوم الخادم بإنشاء معرف جلسة (Session ID) يُخزن على الخادم ويُرسل للعميل غالبًا عبر كوكي. عند كل طلب لاحق، يُرفق هذا المعرف للتحقق من الجلسة



Authentication Stateful مخطط : 1

- الخصائص :

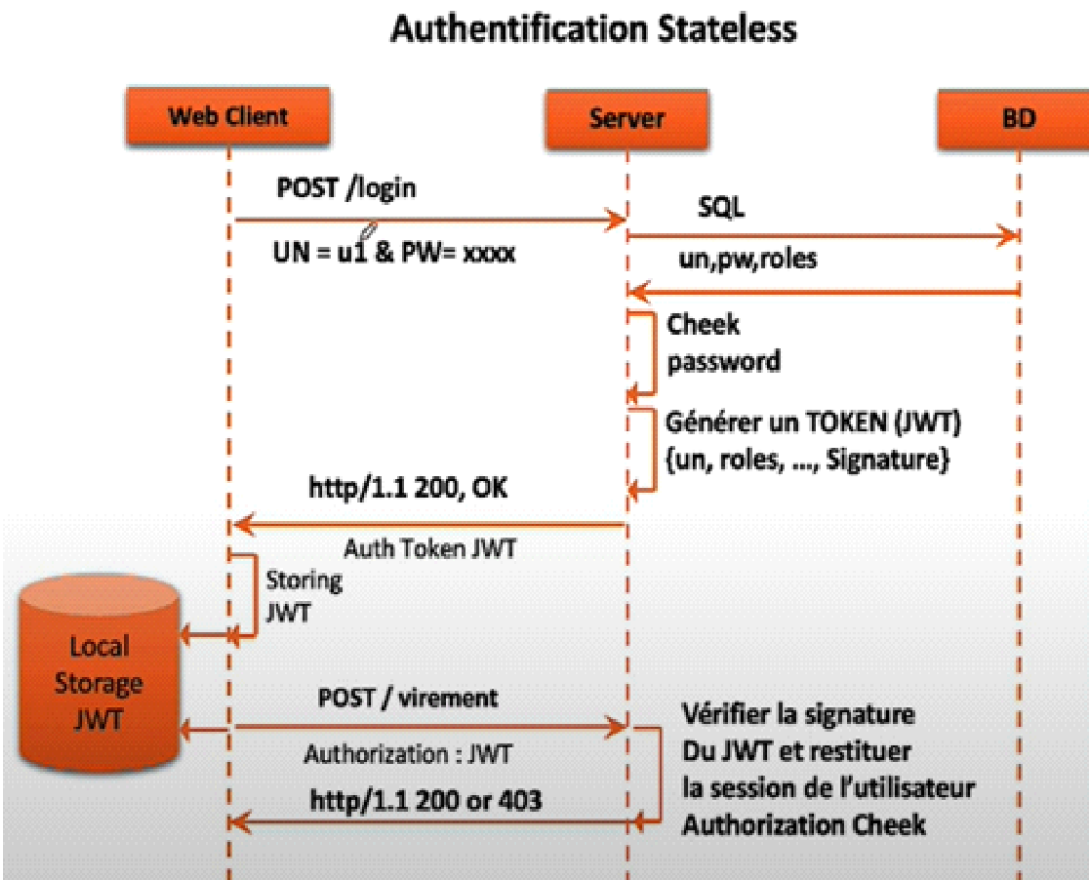
- يحتاج الخادم إلى ذاكرة أو قاعدة بيانات لتخزين الجلسات
- التحقق يتم داخليًا عبر الرجوع إلى بيانات الجلسة المخزنة.

- السلبيات :

- لا يتناسب بسهولة مع الأنظمة الموزعة أو المصغرة
- صعوبة في التوسع الأفقي إلا مع آليات مثل التخزين المشترك

- المصادقة **Stateless**:

لا يحتفظ الخادم بأي حالة للجلسة. بل عند المصادقة الناجحة، يُصدر الخادم رمزًا مميزًا (غالبًا JWT) يحتوي على بيانات المستخدم وتوقيع رقمي. يُخزن هذا الرمز في جانب العميل، ويُرفق مع كل طلب، ويكفي لفحص التوقيع والمحتوى دون الرجوع إلى الخادم.



Authentication Stateless مخطط : 2

- الخصائص :
- مناسب للأنظمة الموزعة
- لا حاجة لتخزين مركزي للجلسات
- سريع وسهل التوسع
- السلبيات :
- لا يمكن إلغاء الجلسة بمجرد إصدار الرمز، إلا باستخدام قوائم الحظر (Token Blacklists)
- يتطلب إجراءات دقيقة لحماية البيانات المُشفرة داخله

1.9 تجربة المستخدم في واجهات التوثيق (UX of Authentication)

تصميم واجهة التوثيق له تأثير كبير على نجاح النظام:

مشاكل شائعة عند الإهمال:

تعقيد نماذج التسجيل يؤدي إلى عزوف المستخدمين.

رسائل خطأ غامضة تسبب إحباط المستخدم.

نصائح لتحسين تجربة المستخدم:

توفير نماذج تسجيل دخول مبسطة وواضحة.

عرض رسائل خطأ وإشعارات مفهومة.

توفير مؤشرات نجاح واضحة عند إكمال العملية.

1.10 نظام keycloak :

1.10.1 تعريف:

هو نظام مفتوح المصدر لإدارة الهوية والوصول(IAM - Identity and Access Management) ، يوفر ميزات تسجيل الدخول الموحد(SSO) ، المصادقة(Authentication) ، والتفويض (Authorization) للتطبيقات والخدمات. تم تطويره من قبل شركة Red Hat، ويُستخدم لحماية التطبيقات بسهولة وفعالية.

2.10.1 أهميته:

- تسجيل دخول موحد (SSO) بين عدة تطبيقات.
- إدارة مركزية للمستخدمين، الصلاحيات، والمجموعات.
- دعم بروتوكولات قياسية مثل:
  OAuth 2.0 - OpenID Connect - SAML 2.0
- التكامل مع LDAP وموفري الهوية الخارجيين مثل Facebook, Google وغيرهم.
- دعم المصادقة متعددة العوامل (2FA) وسياسات الأمان.

3.10.1 مميزاته:

- مفتوح المصدر ومجاني.
- مرن وقابل للتخصيص.
- يمكن تشغيله محليا او في السحابة.
- يدعم بروتوكولات متعددة.

4.10.1 اهم استخداماته :

- بناء نظام تسجيل دخول موحد بين التطبيقات.
- حماية واجهات برمجة التطبيقات (APIs) باستخدام.OAuth2
- تكامل مع مزودي الهوية الخارجيين أو قواعد بيانات المستخدمين.(LDAP)
- نظام إدارة مستخدمين داخلي للشركات أو المشاريع الكبيرة.

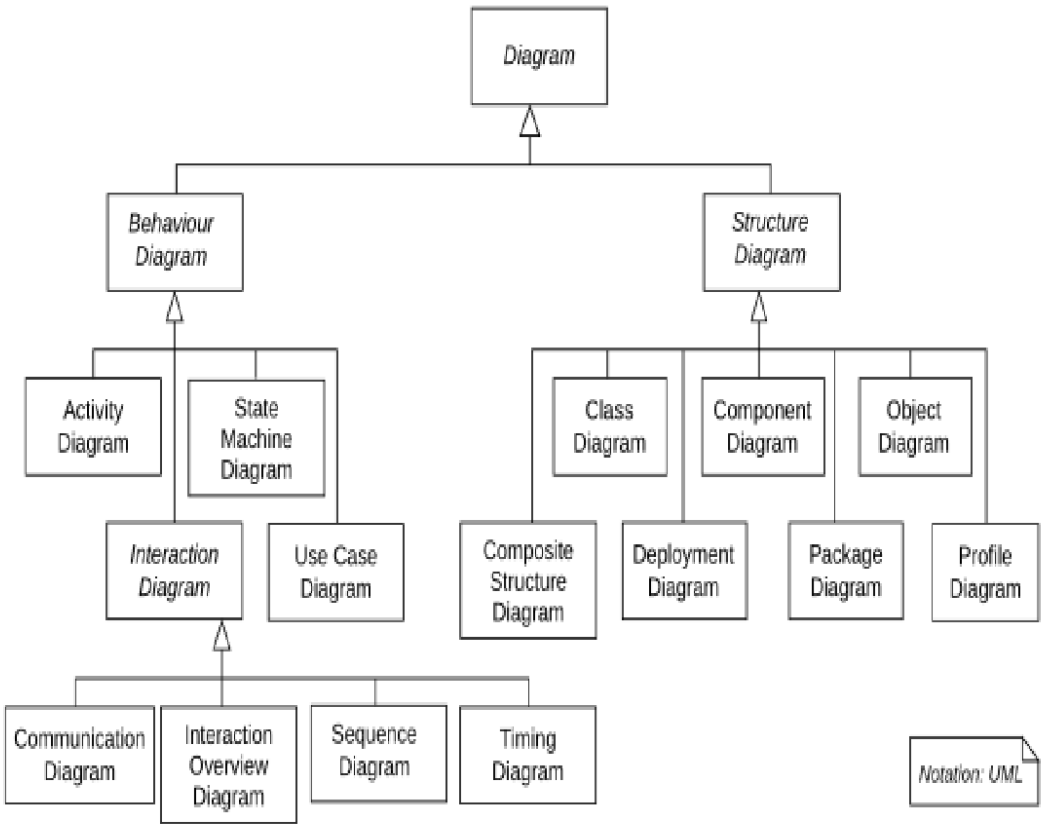
1.11 لغة النمذجة الموحدة (UML)

1.11.1 تعريف UML :

لغة النمذجة الموحدة UML (Unified Modeling Language) هي لغة رسومية تُستخدم في نمذجة الأنظمة البرمجية.

توفّر UML أسلوبًا منهجيًا موحدًا لوصف المكونات الأساسية للنظام، مما يسهل فهمه وتحليله من قبل مختلف الأطراف المعنية، مثل المبرمجين والمصممين.

تُعتبر UML لغة قياسية ومعتمدة لترميز وتصميم العمليات البرمجية، حيث تتيح وسيلة رمزية مبسطة لتمثيل مختلف جوانب النظام، وتساعد بشكل فعال في متابعة مراحل تطوير البرمجيات.(1)



3 : مخطط لغة النمذجة الموحدة UML

تتميز هذه اللغة باعتمادها على مخططات وأشكال هندسية تمثيلية تُمكّن من تقديم رؤية شاملة للنظام قيد التطوير. هذا بدوره يسهم في توضيح مكونات النظام وتسهيل عملية عرضه، مما يجعل صيانته وتحديثه لاحقًا أكثر بساطة وفعالية.وتنقسم مخططات UML الى قسمين اساسيين هما : المخططات الهيكلية و المخططات السلوكية.

1.12 اقسام مخططات : UML

- المخططات الهيكلية :

تركز المخططات الهيكلية على تمثيل البنية الثابتة للنظام أو البرنامج.

تُظهر هذه المخططات مستويات مختلفة من التجريد والتنفيذ، وتُستخدم لتوضيح البنية الداخلية للنظام، كتصميم قواعد البيانات أو مكونات التطبيقات.

كما توضح كيفية تنظيم الأجزاء المختلفة من النظام، مثل الوحدات أو الكائنات، والعلاقات التي تربط بينها، بما في ذلك التسلسل الهرمي والتفاعل.

تساعد هذه المخططات على ضمان توافق مكونات النظام مع بعضها البعض، وتوفر إرشادات واضحة أثناء مراحل التطوير والتنفيذ. (2) وهي :

Class Diagram - (مخطط الفئة)

Object Diagram - (مخطط الكائن)

Component Diagram - (مخطط المكونات)

Deployment Diagram - (مخطط النشر)

Package Diagram - (مخطط الحزم)

Composite Structure Diagram - (مخطط البنية المركبة)

Profile Diagram - (مخطط الأنماط)

-المخططات السلوكية :

تركّز هذه المخططات على الجوانب الديناميكية للنظام.

فهي تُستخدم لتمثيل سلوك النظام أثناء التشغيل، مع التركيز على العمليات التي يجب أن تحدث، والتفاعلات بين المستخدمين والنظام، أو بين مكونات النظام نفسه.

تعكس المخططات السلوكية ما "يحدث" داخل النظام، مما يجعلها مفيدة في فهم المتطلبات الوظيفية وآلية التفاعل بين العناصر المختلفة في بيئة العمل. (2) وهي:

Use Case Diagram - (مخطط حالات الاستخدام)

Sequence Diagram - (مخطط التسلسل)

Activity Diagram - (مخطط النشاط)

State Machine Diagram - (مخطط الحالة)

Communication Diagram - (مخطط الاتصال)

Interaction Overview Diagram - (مخطط نظرة عامة على التفاعل)

Timing Diagram - (مخطط التوقيت)

1.13 منهجية التصميم 2 TUP (Two-Track Unified Process)

1.13.1 تعريف منهجية 2 TUP

تُعد 2TUP منهجية حديثة لتطوير البرمجيات، مستوحاة من المنهجية الشهيرة RUP (Rational Unified Process)، لكنها تضيف إليها بعدًا عمليًا جديدًا من خلال الفصل بين المسار الفني والمسار الوظيفي في عملية التطوير.

تهدف هذه المنهجية إلى ضمان التكامل بين الجوانب التقنية والتطبيقية للمشروع، مع مراعاة متطلبات المستخدمين من جهة، والجوانب المعمارية والتكنولوجية من جهة أخرى.

2.13.1 مبدأ عمل المنهجية

ترتكز منهجية TUP2 على تطوير تكراري وتزايدي (itératif et incrémental) من خلال مسارين متوازيين:

أ. المسار الوظيفي **(Functional Path)**

يهتم بجمع وتحليل المتطلبات , كما يركز على المستخدم النهائي واحتياجاته و يستخدم أدوات مثل:حالات الاستخدام (Use Cases) , نماذج سلوك النظام و واجهات الاستخدام UI/UX.والهدف هو تحديد ما يجب أن يفعله النظام.

ب. المسار التقني**(Technical Path)**

يهتم بالتصميم الداخلي والمعماري للنظام. و يتعامل مع الجوانب التالية : اختيار التكنولوجيا , الأمن المعلوماتي , الأداء و البنية التحتية (معمارية النظام).والهدف هو تحديد كيف سيتم تنفيذ ما طُلب في المسار الوظيفي.

3.13.1 المراحل الأساسية لـ 2 TUP

تشبه إلى حد كبير مراحل RUP، لكنها تُطبق على كلا المسارين بشكل متزامن:

1. البداية (**Inception**): تحديد نطاق المشروع، الأهداف الأساسية، والمتطلبات الأولية. ويتم فيها استخدام : Use Case Diagram , Activity Diagram , Class Diagram و Package Diagram.

2. التفصيل (**Elaboration**): تحليل معمق للمتطلبات، تحديد المخاطر، وضع التصميم المعماري الأولي. ويتم فيها استخدام : Class Diagram , Sequence Diagram , State Machine Diagram , Activity Diagram و Component Diagram.

3. البناء (**Construction**): تطوير البرمجية تدريجيًا حسب الأولويات.ويتم فيها استخدام : Class Diagram , Component Diagram , Sequence Diagram , Communication Diagram و Deployment Diagram.

4. الانتقال (**Transition**): اختبار، توثيق، نشر وتسليم النظام للمستخدم النهائي.ويتم فيها استخدام : Deployment Diagram , Activity Diagram , Use Case Diagram و State Machine Diagram.

تُعتبر منهجية 2TUP مقاربة فعّالة لتطوير البرمجيات من خلال اعتمادها على مسارين متوازيين ومتكاملين، مما يضمن تحقيق التوازن بين ما يحتاجه المستخدم وما يمكن تحقيقه تقنيًا. وهي مناسبة جدًا للمشاريع التي تتطلب تكرارات متعددة وتفاعل مستمر بين الأطراف المعنية.

1.14 المشاكل الشائعة في أنظمة تسيير المستخدمين

يواجه نظام تسيير وإدارة الحسابات والمستخدمين عدة تحديات تؤثر على فعالية الأداء الأمني والتنظيمي داخل المؤسسات الرقمية، مما يتطلب حلولًا تقنية حديثة تواكب متطلبات الحوكمة الرقمية وضمان أمان المعلومات:

النقد:

خلال دراسة بيئة العمل، تبيّن وجود عدد من النقاط السلبية التي تعرقل حسن سير تسيير الحسابات، ومن أبرز هذه المشاكل:

- غياب نظام مركزي لإدارة الصلاحيات والأدوار(Role-Based Access Control) .
- صعوبة تتبع أنشطة المستخدمين والتدقيق في السجلات(Audit Logs) .
- استخدام كلمات مرور ضعيفة أو عدم وجود نظام للتحقق الثنائي(2FA) .
- تأخر معالجة طلبات إنشاء أو تعديل أو حذف حسابات المستخدمين .تكرار الحسابات أو تضارب المعلومات بين أنظمة متعددة دون تكامل بينها.

الحلول:

لمعالجة هذه الإشكالات، تم اقتراح مجموعة من الحلول التقنية والتنظيمية التي من شأنها تحسين تسيير وإدارة الحسابات، ومنها:

- اعتماد نظام مركزي لإدارة الهوية والوصول (IAM) مثل Keycloak أوOkta .
- تفعيل آليات تسجيل وتتبع الأنشطة (Logging & Auditing) لتحليل سلوك المستخدمين وضمان الشفافية.
- فرض سياسة كلمات مرور قوية وتفعيل التحقق الثنائي لتعزيز الأمان .أتمتة عمليات إدارة الحسابات (مثل الإنشاء، التعديل، التعطيل) لتقليل الأخطاء اليدوية وتسريع الاستجابة.
- ربط الأنظمة الداخلية من خلال واجهات برمجية موحدة (APIs) لضمان تكامل البيانات وتفادي التكرار.

1.15 أهمية هذا الفصل في السياق العام

هذا الفصل يُشكل القاعدة النظرية الضرورية لفهم وبناء أنظمة تسيير حسابات المستخدمين بشكل صحيح وآمن.

من خلال الإلمام بالعمليات والمكونات الأساسية، والتحديات التي قد تواجه النظام، ومعايير الأمان المطلوبة، يتمكن المطور لاحقًا من إنشاء إطار عمل متكامل قادر على التكيف مع متطلبات مختلف أنواع تطبيقات الويب.

كما أن هذه المعرفة تساهم في اتخاذ قرارات أفضل فيما يخص التقنيات والأدوات المستخدمة مستقبلاً خلال تطوير المشروع.

خاتمة

بناءً على ما سبق، يأتي هذا المشروع لتقديم نظام تسيير شامل للمستخدمين، يهدف إلى حل المشاكل التقليدية وتقديم تجربة تفاعلية آمنة وفعالة. يركز النظام على:

- بنية مرنة تسمح بإدارة عدد غير محدود من المستخدمين.
- تقسيم المستخدمين حسب الأدوار مع تحكم كامل في الصلاحيات.
- واجهات استخدام بسيطة وسريعة التفاعل.
- نظام أمان متقدم يعتمد التشفير والتحقق المتعدد.
- لوحة تحكم مخصصة للإدارة لمراقبة النشاطات وتحليل الأداء.

هذا النظام يمكن تكييفه مع أي مؤسسة: تعليمية، إدارية، تجارية أو طبية، حيث يُمكن تطويره ليخدم احتياجاتها الخاصة بطريقة مهيكلة وفعالة.

الفصل الثاني : التصميم

1.2 مقدمة

في الفصل السابق، قمنا بدراسة الوضع الحالي وتحليل المشاكل الموجودة، كما اقترحنا الحلول المناسبة لها.

أما في هذا الفصل، فإن الهدف الأساسي هو تحقيق تقدم ملموس في مشروعنا من خلال انشاء تصميم.

ولضمان ذلك، من الضروري اتباع منهجية تصميم واضحة. ومن بين الأساليب المتاحة، اخترنا التصميم الموجه نحو الكائنات (Object-Oriented Design) نظرًا لقدرته العالية على تمثيل الواقع بشكل منطقي وثابت، مما يعزز جودة النمذجة ويقربها من العالم الحقيقي.وذلك من خلال لغة النمذجة الموحدة .UML

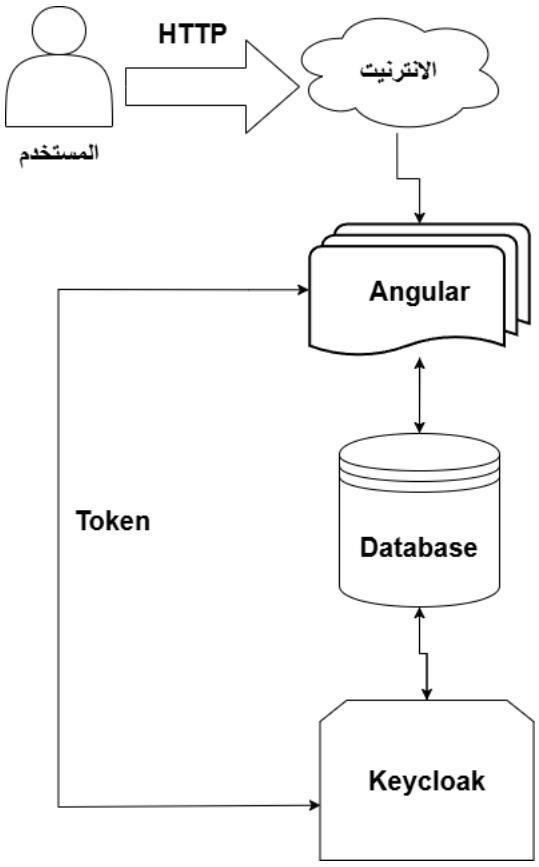
في هذا الفصل، سنقوم بعرض الجوانب التالية:

- المخططات المستخدمة في المشروع

- التصميم

- قاموس المعطيات

2.2 مخطط النظام الجديد :



4 : مخطط هندسة النظام الجديد

عرض بنية النظام الجديد (بنية متعددة الطبقات مع خادم توثيق مستقل):

يعتمد النظام الجديد على بنية موسعة تُبنى على النمط الثلاثي الطبقات (Tier-3)، مع إضافة طبقة رابعة مخصصة لخادم التوثيق. البنية العامة موزعة على النحو التالي:

-المستخدم : جهاز المستخدم الذي يطلب الوصول إلى الموارد، مزود بواجهة استخدام رسومية (غالبًا متصفح ويب) تُعنى بعرض المحتوى.

-خادم التطبيق (**Application Server**): المسؤول عن تنفيذ منطق الأعمال ومعالجة الطلبات، ويعتمد على خوادم خارجية للحصول على البيانات والتحقق من الهوية.

-خادم قاعدة البيانات (**Database Server**): يوفر البيانات المطلوبة لخادم التطبيق ويضمن حفظها وإدارتها بشكل آمن.

- خادم التوثيق (**Authentication Server**): مسؤول عن التحقق من هوية المستخدمين وتوليد الرموز المميزة (Tokens)، ويُستخدم لضمان الأمان وإدارة الجلسات، مثل Keycloak أو OAuth2 server.

بإضافة خادم التوثيق، تصبح البنية أكثر أمنًا ومرونة، حيث يتم فصل عمليات المصادقة عن منطق التطبيق الأساسي.

توفر هذه البنية الموسعة : مرونة أعلى في التطوير والصيانة مع أمان معزز من خلال عزل مسؤولية التوثيق والتفويض عن باقي النظام , إمكانية دعم أنظمة خارجية من خلال خدمات موحدة للمصادقة (SSO) و أداء أفضل نتيجة توزيع المهام بين الخوادم.

3.2 التصميم :

1.3.2 الدراسة الأولية للسياق:

تشكل الدراسة الأولية أولى مراحل عملية 2TUP، وتهدف إلى تكوين تصور أولي عن الاحتياجات التشغيلية، مستندة إلى وصف بسيط باستخدام نصوص أو مخططات، مما يساعد في التحضير لجمع المتطلبات بشكل رسمي في المراحل اللاحقة.

تحديد الفاعلين:

الفاعل هو كيان خارجي (مستخدم، جهاز، نظام آخر) يتفاعل مباشرة مع النظام.

في نظامنا ، حددنا الفاعلين الأوليين كما يلي:

-المستخدم : في حالتنا هو من يقوم بانشاء حساب او صاحبه و المستخدم هنا هو فاعل بصفة عامة لمجموعة فاعلين مختلفين لبرامج معينة

-المسؤول: وهو الذي يقوم بتسيير الحسابات

تحديد الرسائل:

الرسالة هي عملية إرسال معلومات من كائن إلى كائن آخر بهدف تنفيذ نشاط معين.

في نظامنا:

- النظام يستقبل الرسائل التالية:

- معلومات حساب

- ادوار المستعملون المعينة

- النظام يرسل الرسائل التالية:

- قائمة الحسابات المرشحة للحذف

- قائمة الحسابات غير المؤكدة

- بيانات حساب

- قائمة الادوار المسندة

بيئة النظام :

بناءً على التحليل السابق، نحدد وظائف الفاعلين على النحو التالي:

- المسؤول:

- تسجيل الدخول

- تسيير الحسابات

- تغيير المعلومات

- حذف الحسابات

- المستخدم :

-انشاء حساب

- تسجيل الدخول

- ارجاع كلمة السر

- تغيير المعلومات

المستعملين النهائيين	وصف المتطلبات الوظيفية
المسؤول	-تسجيل الدخول -تسيير الحسابات -تغيير المعلومات - حذف الحسابات

المستخدم	- تسجيل الدخول - انشاء حساب - ارجاع كلمة السر - تغيير المعلومات

2.3.2 تحديد الاحتياجات الوظيفية:

يتمثل الهدف في هذه المرحلة في تنظيم المتطلبات الوظيفية باستخدام مخطط حالات الاستخدام، التي تعتمد على خطوتين رئيسيتين: تحديد ووصف حالات الاستخدام و استخراج الفئات المرشحة للنمذجة.

1.2.3.2 تحديد حالات الاستخدام :

الهدف هو كما يلي: يجب أن يصف مجمل حالات الاستخدام بشكل شامل جميع المتطلبات الوظيفية للنظام. وعليه، فإن كل حالة استخدام تمثل وظيفة من وظائف العمل داخل النظام، وذلك من وجهة نظر أحد الفاعلين فيه.

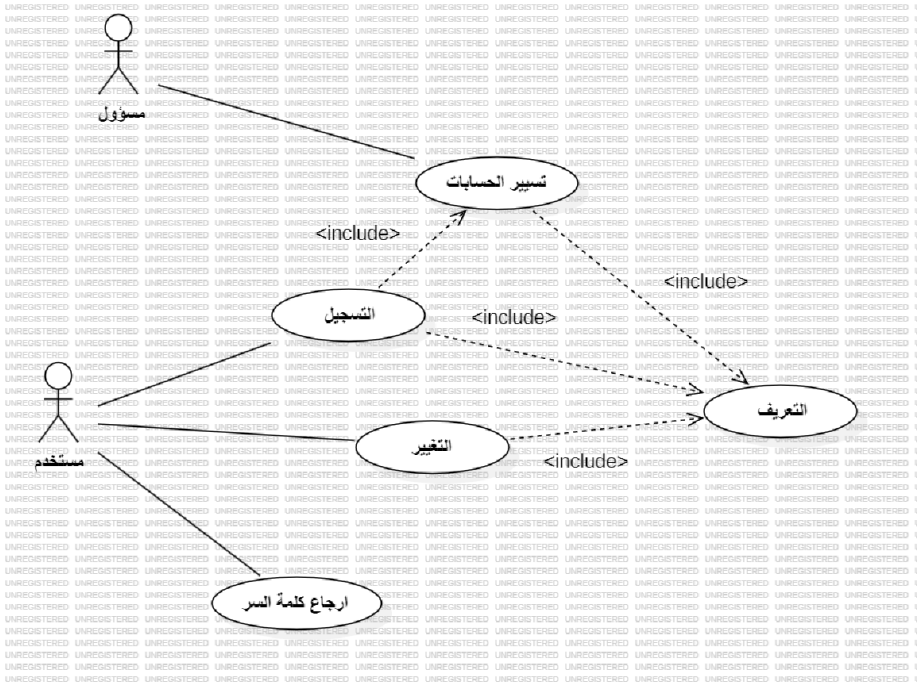
ولكل فاعل يتم التعرف عليه، ينبغي القيام بما يلي:

- البحث عن مختلف النوايا التي تدفعه لاستخدام النظام

- تحديد الخدمات الوظيفية التي يتوقعها من النظام

الجدول التالي يلخّص جميع حالات الاستخدام التي سيتم تفصيلها لاحقًا، مع إبراز العناصر التالية: الفاعل الرئيسي , الفاعل الثانوي , مخطط حالة الاستخدام , التسلسلات (السيناريوهات) ومخطط النشاط.

الفاعل	حالة الاستخدام
المسؤول	- تسجيل الدخول - تسيير الحسابات
المستخدم	- تسجيل الدخول -انشاء حساب - التغيير -ارجاع كلمة المرور



5 : مخطط الحالة للنظام

ملاحظة : في المخطط "التعريف" تدل على عملية انشاء حساب المعمولة مسبقا أي ان كل عملية تتطلب وجود حساب لكي تتم تنفيذها

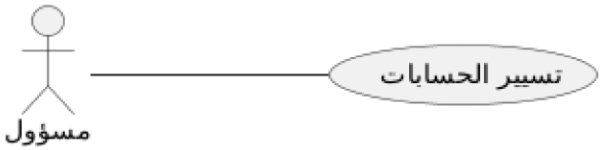
I - حالة الاستخدام " تسيير الحسابات "

الوصف الاولي :

المطلوب : تسيير الحسابات

الاجراءات : التسجيل , اعطاء الدور , حذف مستخدم , تغيير معلومات , تأكيد حساب , المراقبة

مخطط حالة الاستخدام :



6 : مخطط الحالة لمهام المسؤول

سجل حالة الاستخدام لتسيير الحسابات :

ملخص التعريف :

- العنوان : مهام المسؤول
- الهدف : التسيير والمراقبة
- الملخص : يقوم المسؤول بتسجيل الدخول ثم يقوم بادارة المهام والتسيير
- المعني : المسؤول

وصف السلسلة :

- الشروط الابتدائية : يجب على المستخدم ان يقوم بعملية تسجيل الدخول (المصادقة)
- السيناريو الطبيعي : يبدأ الاستخدام بعد كل عملية انشاء حساب
- التسلسل : تسيير الحسابات تندرج فيها عدة مهام وهي :

- تأكيد حساب

-إعطاء الدور

- تغيير المعلومات

- حذف حساب (الغاء التنشيط)

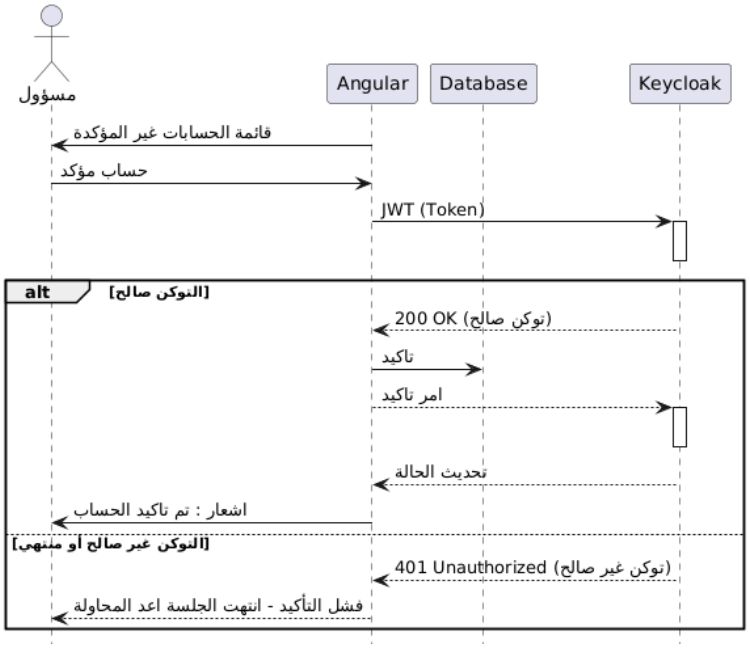
- تأكيد حساب :

الوصف الاولي :

المطلوب : تأكيد حساب

الاجراءات : المصادقة , اختيار الحسابات غير المؤكدة , التأكيد

مخطط التسلسل :



7 : مخطط التسلسل لتأكيد حساب

ملخص التعريف :

- العنوان : تأكيد حساب
- الهدف : التأكيد على الحسابات
- الملخص : يقوم المسؤول بالولوج الى قائمة الحسابات غير المؤكدة ثم يقوم بتأكيدها
- المعني : المسؤول

وصف السلسلة :

- الشروط الابتدائية : يجب على المستخدم ان يقوم بعملية تسجيل الدخول (المصادقة)
- السيناريو الطبيعي : يبدأ الاستخدام بعد كل عملية انشاء حساب
- التسلسل : تأكيد الحسابات

-الولوج الى قائمة الحسابات غير المؤكدة

- اختيار الحسابات المراد تأكيدها

- التأكيد

- انتظار تحديد الحالة الخاصة بالحسابات

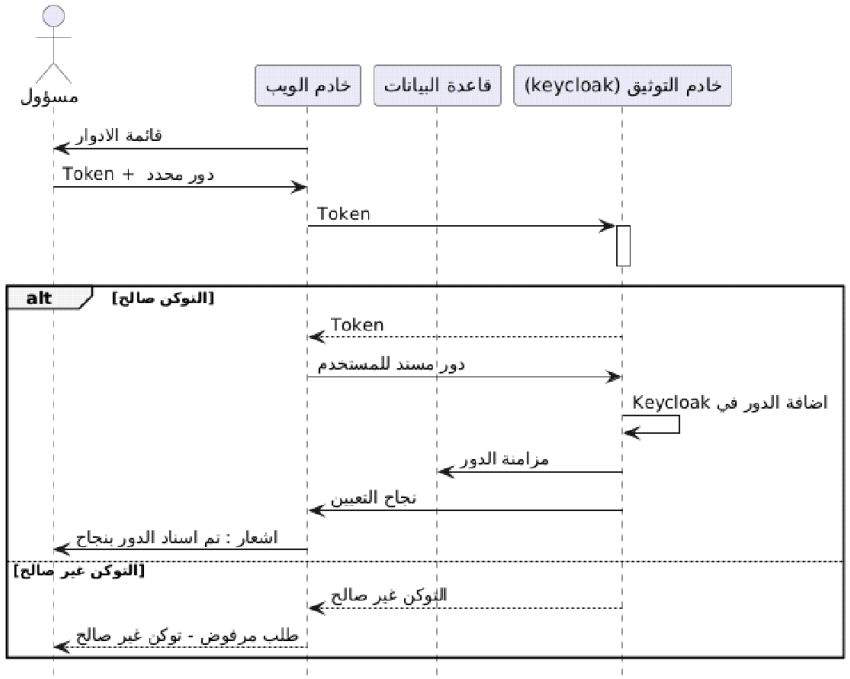
- اعطاء الدور :

الوصف الاولي :

المطلوب : تحديد الدور

الاجراءات : المصادقة , تحديد المستخدمين المعنيين , اعطاء دور لكل مستخدم

مخطط التسلسل :



8 : مخطط التسلسل لتحديد الدور

ملخص التعريف :

- العنوان : تحديد الدور
- الهدف : توزيع الأدوار على الحسابات
- الملخص : يقوم المسؤول بالولوج الى قائمة الحسابات ثم يقوم باختيار الحساب المعني وتحديد دور له
- المعني : المسؤول

وصف السلسلة :

- الشروط الابتدائية : يجب على المستخدم ان يقوم بعملية تسجيل الدخول (المصادقة)
- السيناريو الطبيعي : يبدأ الاستخدام بعد كل عملية انشاء حساب
- السنياريو الاحترازي : في حال كان لدى المستخدم دور مسبقا و أراد المسؤول تغييره
- التسلسل : تحديد الدور

-الولوج الى قائمة الحسابات

- اختيار الحسابات المطلوبة

-اختيار الدور لهاته الحسابات

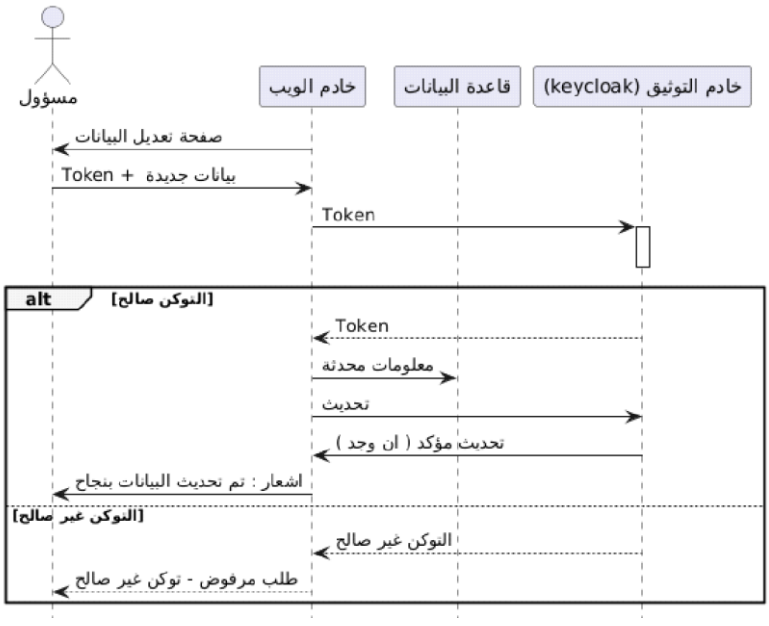
- تغيير المعلومات :

الوصف الاولي :

المطلوب : تغيير المعلومات

الاجراءات : المصادقة , اختيار المستخدم , اجراء تحديث على بياناته ومعلوماته

مخطط التسلسل :



9 : مخطط التسلسل تغيير المعلومات

ملخص التعريف :

- العنوان :تغيير المعلومات
- الهدف : تحديث بيانات و معلومات المستخدمين
- الملخص : يقوم المسؤول بطلب صفحة تعديل البيانات ويختار الحساب المعني و يقوم بالتعديل عليه ثم يقوم بحفظ التغييرات
- المعني : المسؤول

وصف السلسلة :

- الشروط الابتدائية : يجب على المستخدم ان يقوم بعملية تسجيل الدخول (المصادقة)
- السيناريو الطبيعي : يبدأ الاستخدام في حالة طلب المستخدم تغيير معلوماته
- التسلسل : تغيير معلومات

- اختيار الحساب المعني
- الدخول لصفحة تغيير البيانات
- اختيار البيانات المراد تغييرها او تعديلها
- القيام بالتغييرات مع الحفظ

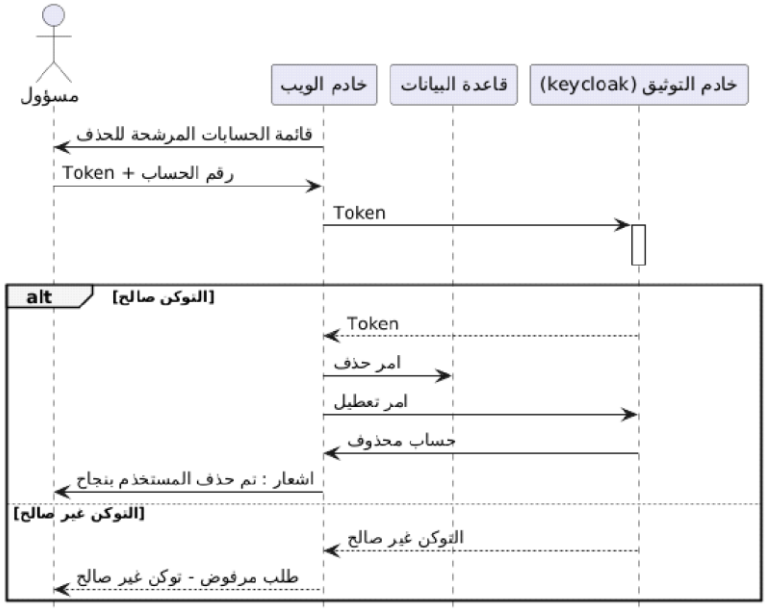
- حذف حساب :

الوصف الاولي :

المطلوب : حذف حساب (الغاء التنشيط)

الاجراءات : المصادقة , اختيار الحساب , القيام بالحذف

مخطط التسلسل :



10 : مخطط التسلسل لحذف حساب

ملخص التعريف :

- العنوان : حذف حساب
- الهدف : الغاء تنشيط الحسابات
- الملخص : يقوم المسؤول باختيار الحساب ثم يقوم بحذفه
- المعني : المسؤول

وصف السلسلة :

- الشروط الابتدائية : يجب على المستخدم ان يقوم بعملية تسجيل الدخول (المصادقة)
- السيناريو الطبيعي : يبدأ الاستخدام في حالة تم انتهاك الشروط
- السيناريو الاحترازي : يبدأ الاستخدام في حال تجاوز المستخدم مدة 365 يوم من دون القيام بأي عملية تسجيل دخول
- التسلسل : حذف حساب

- عرض الحسابات المرشحة للحذف

-تحديد الحسابات

- القيام بالحذف (الحذف يكون حذف منطقي وليس حذف فيزيائي أي أنه عبارة عن الغاء التنشيط للحساب)

II - حالة الاستخدام " التسجيل " :

الوصف الاولي :

المطلوب : التسجيل

الاجراءات : طلب انشاء حساب او طلب تسجيل الدخول , ادراج المعلومات المطلوبة , الاتمام

مخطط حالة الاستخدام :

11 : مخطط الحالة للتسجيل

ملخص التعريف :

العنوان: عملية التسجيل الهدف: تمكين المستخدم الجديد من إنشاء حساب والدخول إلى النظام الملخص: يقوم المستخدم بملء نموذج التسجيل بمعلوماته (مثل الاسم، البريد الإلكتروني، وكلمة المرور)، ثم يتم التحقق من صحة البيانات، وفي حال كانت صحيحة يُنشأ له حساب جديد في النظام. المعني: المستخدم

وصف السلسلة :

• الشروط الابتدائية : يجب ألا يكون لدى المستخدم حساب سابق في النظام. - يجب أن يكون لديه إمكانية الوصول إلى بريد إلكتروني فعّال. • - يجب على المستخدم لاحقًا تنفيذ عملية المصادقة (تسجيل الدخول) • السيناريو الطبيعي : يبدأ هذا الاستخدام عندما يختار المستخدم خيار "التسجيل" • التسلسل : التسجيل - يقوم بملء نموذج التسجيل وإرساله. -يتم التحقق من صحة البيانات وإرسال رسالة تأكيد. -بعد التأكيد، يُنشأ الحساب ويصبح المستخدم مؤهلاً لتسجيل الدخول.

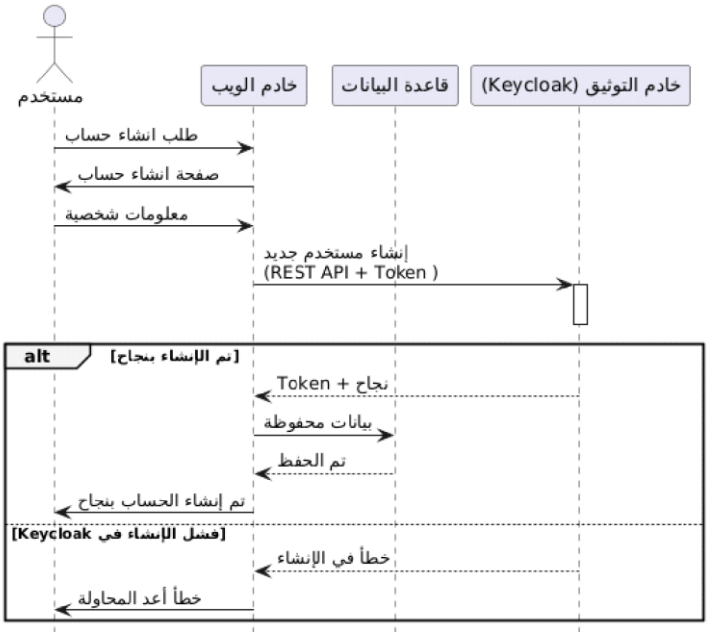
- انشاء حساب :

الوصف الاولي :

المطلوب : انشاء حساب

الاجراءات : طلب انشاء حساب , ادخال المعلومات اللازمة , الضغط على " انشاء "

مخطط التسلسل:



12 : مخطط التسلسل لانشاء حساب

ملخص التعريف :

العنوان: إنشاء حساب
الهدف: السماح للمستخدم بإنشاء حساب جديد للوصول إلى وظائف النظام
الملخص: يقوم المستخدم بملء نموذج التسجيل بالمعلومات الشخصية (مثل الاسم، البريد الإلكتروني، كلمة المرور...). بعد التحقق من صحة البيانات، يتم إرسال رسالة تأكيد إلى البريد الإلكتروني، وعند التأكيد يتم إنشاء الحساب.
المعني: المستخدم

وصف السلسلة :

الشروط الابتدائية : لا يجب أن يكون للمستخدم حساب سابق في النظام.
يجب توفر بريد إلكتروني صالح للوصول إليه.
يجب أن يوافق المستخدم على شروط الاستخدام.

السيناريو الطبيعي : يبدأ السيناريو عندما يختار المستخدم خيار "إنشاء حساب" من واجهة النظام.

التسلسل : يفتح المستخدم صفحة "إنشاء حساب".
يملأ نموذج التسجيل (اسم، بريد إلكتروني، كلمة مرور...).
يضغط على زر "تسجيل".
يتحقق النظام من صحة البيانات وصلاحية البريد.
يُنشأ الحساب نهائيًا ويُصبح نشطًا.

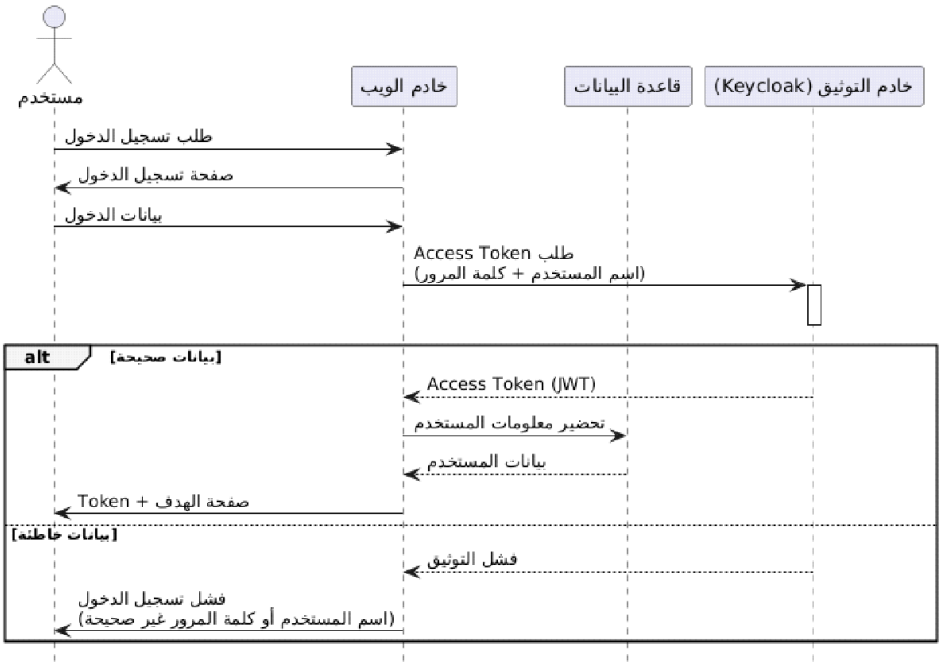
- تسجيل الدخول :

الوصف الاولي :

المطلوب : تسجيل الدخول

الاجراءات : طلب تسجيل الدخول , ادخال المعلومات اللازمة , الضغط على " تسجيل "

مخطط التسلسل :



13 : مخطط التسلسل لتسجيل الدخول

ملخص التعريف :

العنوان: تسجيل الدخول الهدف: السماح للمستخدم المصادق عليه بالوصول إلى النظام. الملخص: يدخل المستخدم بيانات الاعتماد (البريد الإلكتروني وكلمة المرور) في واجهة تسجيل الدخول. يقوم النظام بالتحقق من صحة المعلومات، وفي حال كانت صحيحة يتم منحه صلاحية الدخول وعرض المحتوى المخصص له بناءً على دوره. المعني: المستخدم

وصف السلسلة :

الشروط الابتدائية : يجب أن يكون للمستخدم حساب مسجل ومؤكد في النظام و يجب أن تكون بيانات الاعتماد صحيحة ومطابقة لما هو مخزن في النظام.

السيناريو الطبيعي : يبدأ عندما يدخل المستخدم إلى صفحة تسجيل الدخول.

التسلسل : يفتح المستخدم صفحة تسجيل الدخول.
يُدخل البريد الإلكتروني وكلمة المرور.
يضغط على زر "تسجيل الدخول".
النظام يتحقق من البيانات عبر المصادقة.
إذا كانت البيانات صحيحة، يُسجل الدخول.
يتم توجيه المستخدم إلى الصفحة الخاصة به.
إذا كانت البيانات خاطئة، يظهر له إشعار بالخطأ

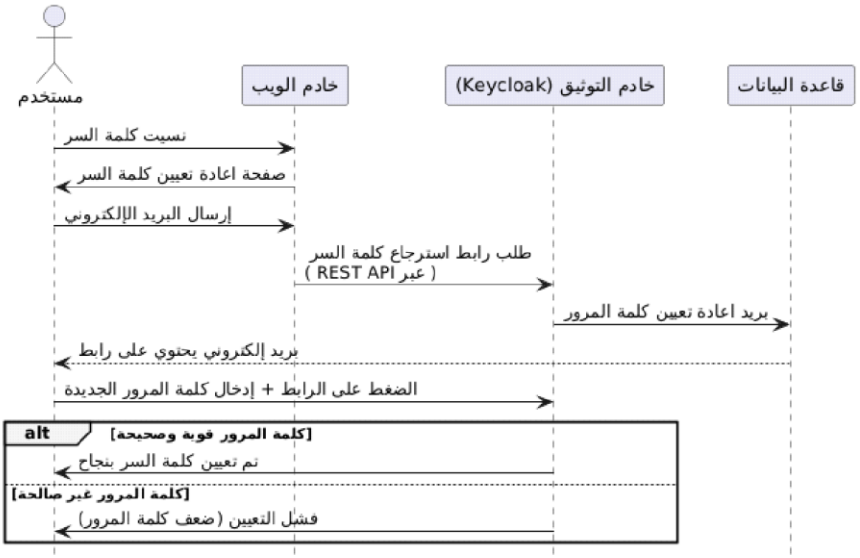
- ارجاع كلمة المرور :

الوصف الاولي :

- المطلوب : ارجاع كلمة المرور

- الاجراءات : طلب اعادة تعيين كلمة السر , ادخال كلمة سر جديدة , الضغط على " تأكيد "

مخطط التسلسل :



14 : مخطط التسلسل لارجاع كلمة السر

ملخص التعريف :

العنوان: ارجاع كلمة السر الهدف: استرجاع كلمة السر او تغييرها الملخص: يدخل المستخدم كلمة السر الجديدة , وبعد التأكد من انها توافق الشروط يتم التأكيد عليها وتحديثها المعني: المستخدم

وصف السلسلة :

الشروط الابتدائية : يجب أن يكون للمستخدم حساب مسجل ومؤكد في النظام، ويجب أن يكون البريد الإلكتروني الذي يتم إدخاله مطابقًا لما هو مخزن في قاعدة البيانات. السيناريو الطبيعي : يبدأ عندما يضغط المستخدم على رابط " نسيت كلمة السر ". التسلسل : يفتح المستخدم صفحة تسجيل الدخول و يضغط على رابط "نسيت كلمة المرور؟". يظهر حقل لإدخال البريد الإلكتروني و يقوم المستخدم بادخال بريده الإلكتروني المسجل و يضغط على زر "إرسال". إذا كان البريد الإلكتروني موجودًا في قاعدة البيانات يُرسل النظام رابطًا لإعادة تعيين كلمة المرور إلى البريد المدخل و يظهر للمستخدم إشعار بنجاح إرسال الرابط. إذا لم يكن البريد الإلكتروني موجودًا يظهر إشعار بخطأ يُفيد بأن البريد غير موجود أو غير مسجل. عند تلقي البريد الإلكتروني، يضغط المستخدم على الرابط و يُفتح نموذج جديد لإدخال كلمة مرور جديدة وتأكيدها ثم يُدخل المستخدم كلمة المرور الجديدة ويؤكدها. النظام يتحقق من صحة الإدخال، ثم يُحدث كلمة المرور. يتم توجيه المستخدم إلى صفحة تسجيل الدخول مع إشعار بنجاح العملية

3.3.2 تحديد الاحتياجات التقنية :

يغطي جمع الاحتياجات التقنية، بشكل مكمل لجمع الاحتياجات الوظيفية، جميع القيود التي لا تتعلق لا بوصف عمل المستخدمين، ولا بوصف التطبيق. يُعبَّر عن نموذج المواصفات من خلال منظورين: المواصفات البرمجية، وهندسة النظام والبنية المادية المستغلة.

1.3.3.2 تحديد حالات الاستخدام التقنية :

من أجل إعداد نموذج المواصفات البرمجية، يتم التركيز على الوظائف الخاصة بالنظام من خلال إعداد مواصفات برمجية.

في هذا السياق، يتم استخدام حالات الاستخدام التقنية.

المستفيدون من النظام هم كالتالي:

-المستخدم : وهو الشخص الذي يستخدم تطبيق النظام. وبالتالي فإن معظم الفاعلين في الجانب الوظيفي يُعتبرون مستخدمين أيضًا في البعد التقني.

- مهندس التشغيل: وهو المسؤول عن نشر النظام وصيانته وإصلاح الأعطال فيه.

يتم أولاً تحديد حالات الاستخدام التقنية من خلال النظر في التوقعات التشغيلية لكل فئة من فئات المستفيدين :

- المستخدم سيقوم بالتعامل مع الكيانات على شكل كائنات (Objects)، مما يتطلب تنفيذ آليات الاستمرارية (Persistence) وإدارة دورة حياة هذه الكائنات.

- عدة مستخدمين يمكنهم العمل بشكل متزامن. وهنا تظهر أهمية السلامة (Integrity) كآلية تمنع تحديث نفس الكيان في الوقت ذاته من قبل مستفيدَين مختلفَين.

- كل مستخدم يستفيد من آلية توزيع الأحمال على مستوى الخادم، بحيث لا تتأثر أوقات الاستجابة بعدد المستخدمين المتصلين.

- يجب على المستخدم تسجيل الدخول والتعرف عليه من قِبل النظام. وتُعد المصادقة (Authentication) هي الآلية التي تحمي النظام من التسللات الخارجية.

- يجب أن يكون النظام قابلاً للتشغيل والإدارة، ولهذا يجب أن يكون قادرًا على توليد سجلات (Logs) وتنبيهات (Alerts) تسهل صيانته ضمن بيئة نظام المعلومات في المؤسسة.

هذا التحليل التقني للمشكلة هو الذي يبرز دور مهندس التشغيل كمستفيد إضافي من النظام.

المستفيدون من النظام خاضعون لقواعد أمنية تشمل:

- المصادقة (Authentication)

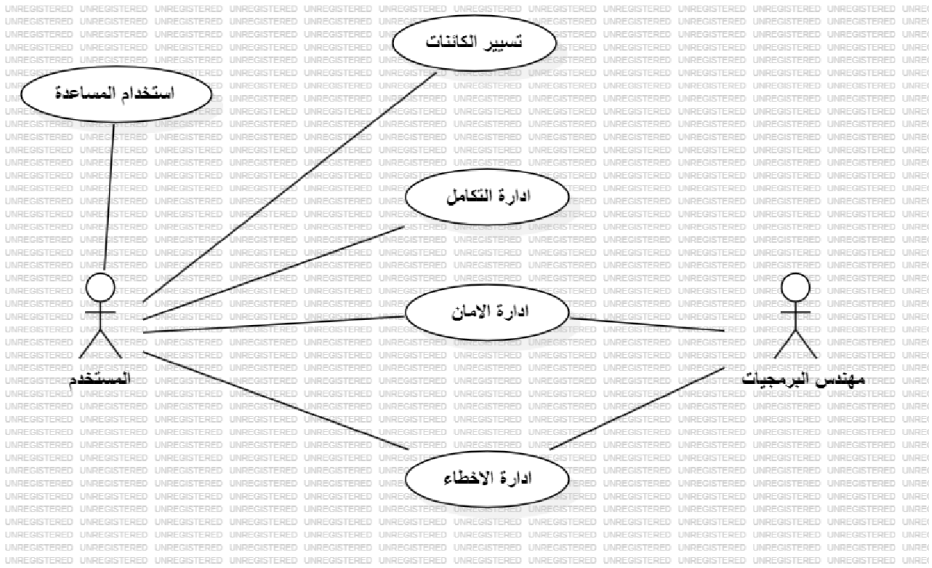
- التفويض (Habilitation)

- التشفير (Cryptage)

- عدم الإنكار (Non-répudiation)

- المراجعة (Audit)

تُنتج القيود التقنية على الاستخدام نموذجًا لمواصفات البرمجية (الوظائف الخاصة بالنظام التقني)، والذي يتم تمثيله من خلال مخطط حالة الاستخدام التالي :



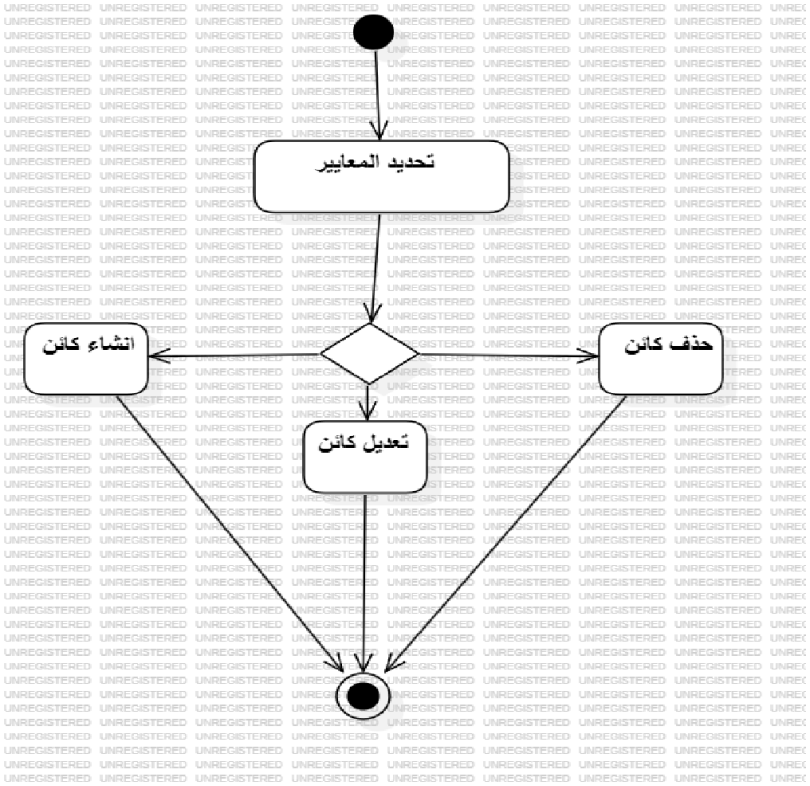
15 : نموذج مواصفات البرمجيات للنظام

- وصف حالات الاستخدام :

أ - حالة الاستخدام " التعامل مع الكائنات " :

المطلوب : يرغب المستخدم في التفاعل مع دورة حياة كائن واحد أو عدة كائنات

الاجراءات : إنشاء، تعديل، حذف كائن أو مجموعة مترابطة من الكائنات (رسم بياني من الكائنات)



16 : مخطط النشاط لحالة الاستخدام " التعامل مع الكائنات "

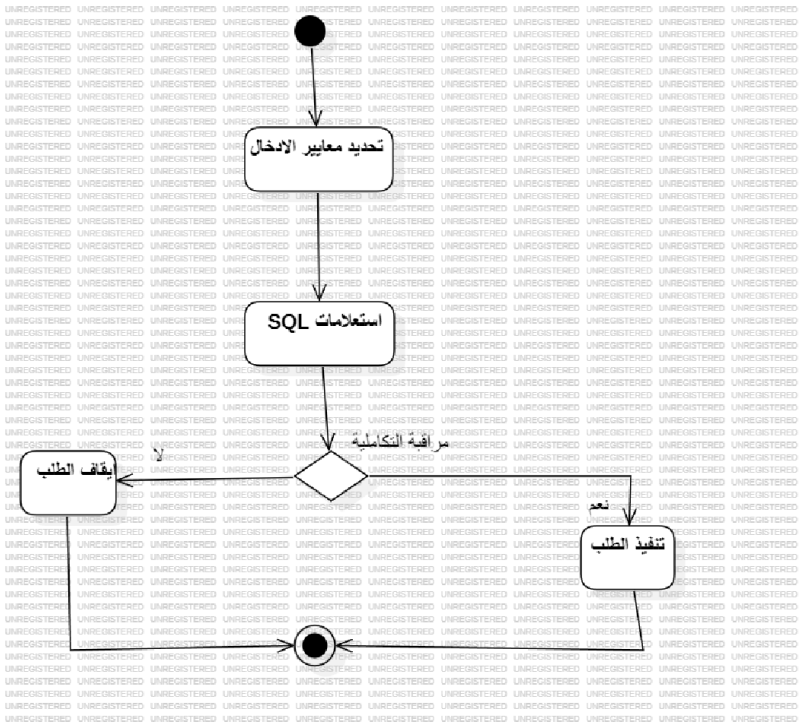
ب - حالة الاستخدام " ادارة التكامل " :

المطلوب : يمكن للمستخدمين التفاعل مع كائن واحد في الوقت نفسه، لذا يجب

منع التحديث المتزامن له

الاجراءات : السماح بالوصول المتوازي لعدة مستخدمين إلى نفس الكيان،

وإدارة سلامة (تكامل) هذه الكيانات

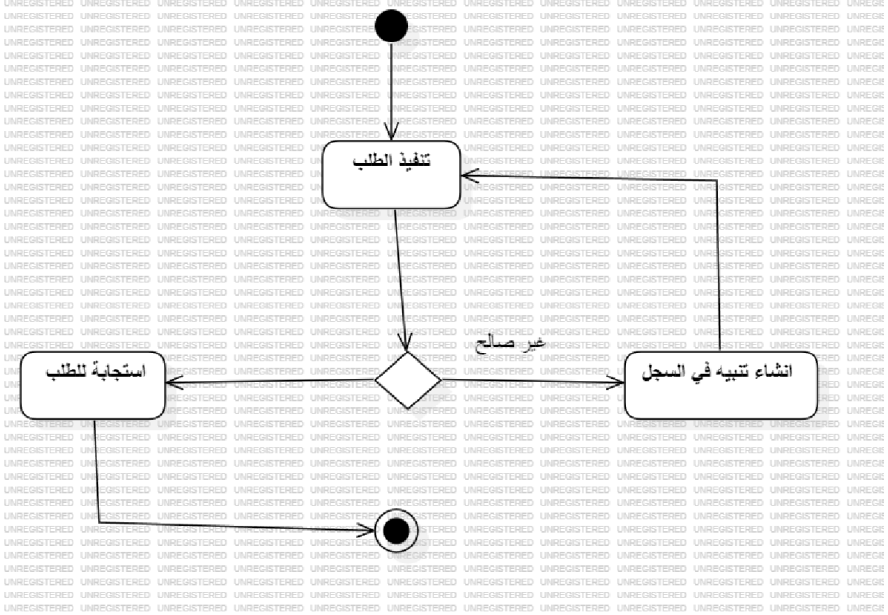


17 : مخطط النشاط لحالة الاستخدام " ادارة التكامل "

ج - حالة الاستخدام " ادارة الاخطاء "

المطلوب : القدرة على إدارة الأخطاء التي يُولدها النظام أثناء تشغيله

الاجراءات : تحرير رسائل الخطأ (تنبيهات، سجلات ...)

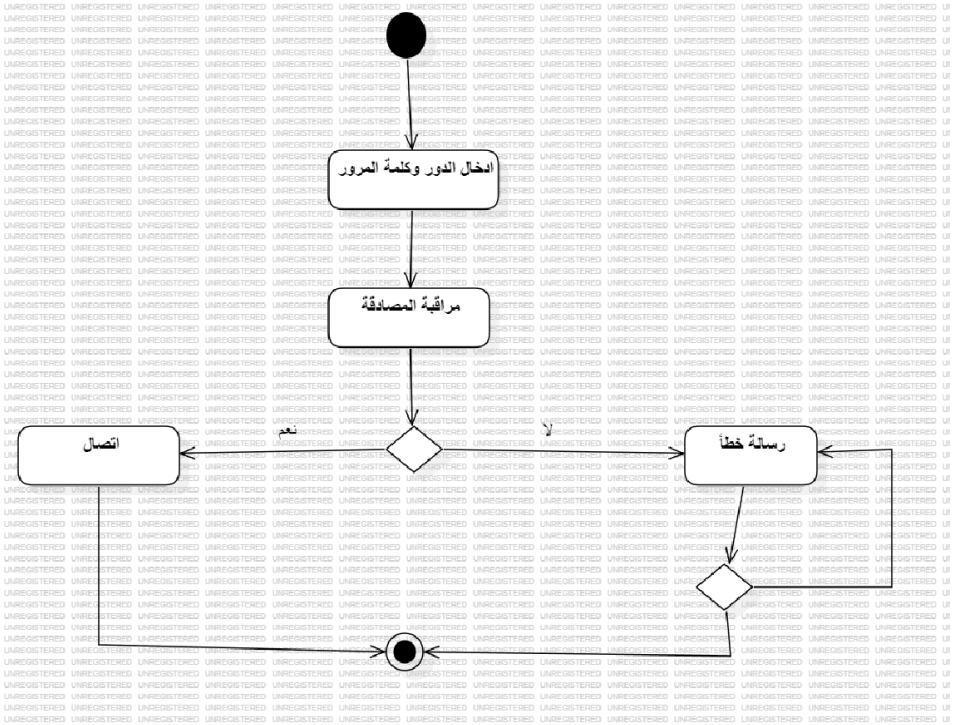


18 : مخطط النشاط لحالة الاستخدام " ادارة الاخطاء "

د - حالة الاستخدام " ادارة الامان "

المطلوب : تزويد النظام بآلية أمان تحميه من التسللات والوصولات الخبيثة

الاجراءات : التحقق من هوية كل مستغل/مشغّل للنظام عبر تعيين دور وكلمة مرور له

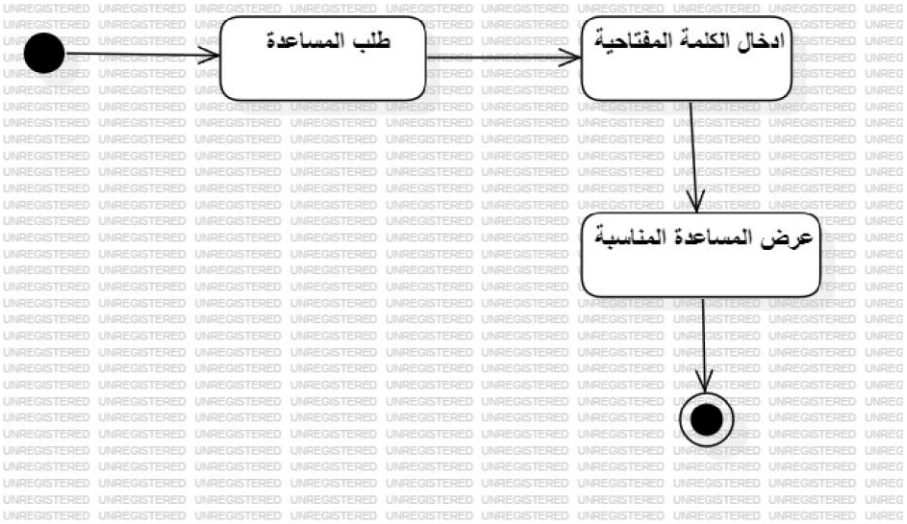


19 : مخطط النشاط لحالة الاستخدام " ادارة الامان "

هـ - حالة الاستخدام " استخدام المساعدة " :

المطلوب : تسهيل استخدام النظام للمستخدمين المختلفين

الاجراءات : توثيق النظام من خلال تقديم مساعدة سياقية

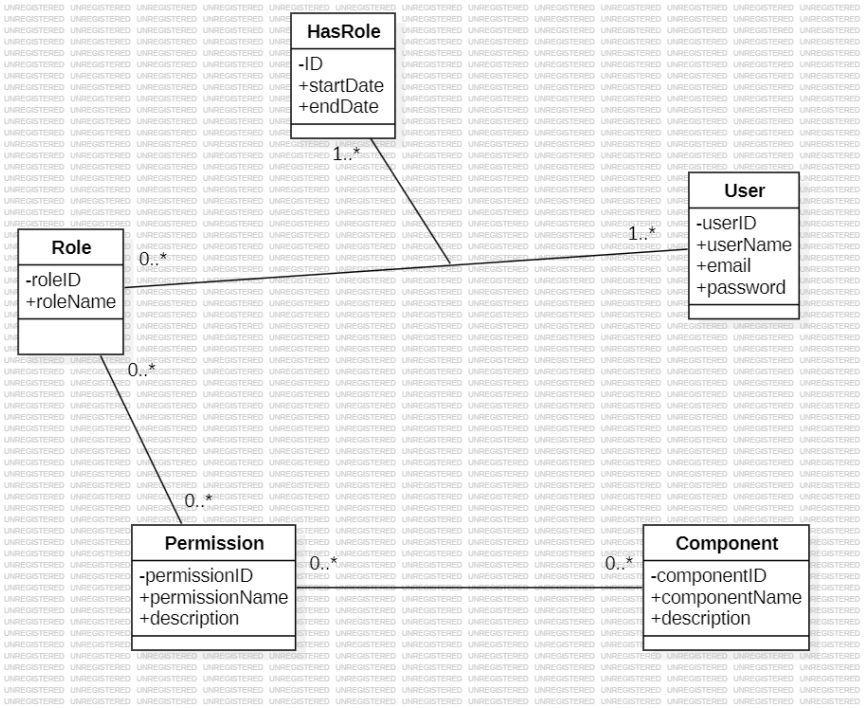


20 : مخطط النشاط لحالة الاستخدام " استخدام المساعدة "

2.3.3.2 انشاء (Static Model) :

هذه المرحلة ستُتيح لنا توضيح البُنى الرئيسية لمخطط الأصناف (Class Diagram). سيتم تفصيل هذه المخططات التي تم إعدادها بشكل موجز خلال مرحلة تحليل الاحتياجات الوظيفية، كما سيتم استكمالها وتحسينها

مخطط الفئات :



21 : مخطط الفئات للمهام

3.3.3.2 التصميم التفصيلي :

يُعدّ التصميم التفصيلي مرحلة نهائية من مراحل النمذجة، وتتمثل في بناء وتوثيق الأصناف (الكلاسات) والجداول بشكل دقيق، وهي التي تشكّل أساس ترميز (كود) الحل.

أ - وصف نموذج الاصناف :

Class	Attributs	Type
User	userID	***N[20]***
	userName	***S[30]***
	email	***S[30]***
	password	***S[50]***
Role	roleID	***N[20]***
	roleName	***S[20]***
Permission	permissionID	***N[20]***
	permissionName	***S[30]***
	description	***S[50]***
Component	componentID	***S[10]***
	componentName	***S[20]***
	description	***S[50]***

قائمة الفئات 22 :

ب - وصف نموذج العلاقات :

Associations	Attributs	Type
HasRole	ID	***N[20]***
	startDate	***D***
	endDate	***D***

قائمة نموذج العلاقات 23 :

ج - قائمة جداول قاعدة البيانات :

Table	Identifiant	Attributs
User	**userID**	**userId,** username, email, password
Role	**roleID**	**roleID**, roleName
Permission	**permissionID**	**permissionID,** permissionName, description
Component	**componentID**	**componentID,** componentName, description
HasRole	**userID, roleID, ID**	**userID, ID, roleID,** startDate, endDate

قائمة جداول قاعدة البيانات 24 :

4.2 الخاتمة :

تُعد النمذجة مرحلة أساسية ومهمة تسبق عملية تطوير النظام، حيث تتبعنا خلالها مختلف خطوات التطوير، بدءًا من جمع احتياجات المستخدم عن طريق التوثيق والتحليل، مرورًا بتصميم النظام الذي يهدف إلى معالجة المشكلات المطروحة بشكل جزئي، وذلك بدراسة الواقع وتحليل المتطلبات بهدف تلبية احتياجات المستخدم النهائي.

وعليه، تبقى هذه الدراسة مفتوحة لجميع الاقتراحات والملاحظات البنّاءة التي من شأنها أن تساهم في تحسين النظام وتطويره.

الفصل الثالث: الدراسة التقنية

3.1 مقدمة

بعد الانتهاء من الجانب النظري ومرحلة التصميم، ننتقل إلى مرحلة التنفيذ العملي للنظام الجديد. في هذا الفصل، سنتناول الأدوات والتقنيات المستخدمة، بالإضافة إلى لغات البرمجة الضرورية لإنجاز التطبيق. كما سنعرض الواجهات الرسومية التي تم تصميمها لتجسيد وظائف النظام بشكل مرئي وتفاعلي.

3.2 الادوات التقنية :

1.3.2 **IntelliJ IDEA** :

IntelliJ IDEA هو بيئة تطوير متكاملة (IDE) تم تطويرها من قبل شركة JetBrains، وتُعد واحدة من أشهر وأقوى الأدوات المستخدمة في تطوير تطبيقات Java، كما تدعم العديد من لغات البرمجة الأخرى مثل Kotlin، Groovy، Scala، SQL، TypeScript، JavaScript، وغيرها.ومن اهم مميزاته :

-دعم ذكي للغات البرمجة: حيث يوفّر تحليلًا دلاليًا (Semantic Analysis) متقدمًا.ويكشف الأخطاء أثناء الكتابة ويقترح حلولاً ذكية (Intelligent Code Completion).

-بيئة مريحة للتطوير: حيث يحتوي على واجهة مستخدم بسيطة وفعّالة و إمكانيات تخصيص عالية (Themes، Plugins، Keymaps).

-تكامل مع أدوات البناء (**Build Tools**): حيث يدعم Maven وGradle بسهولة، مع إمكانية استيراد المشاريع ومزامنتها تلقائيًا.

-تكامل مع أنظمة التحكم في الإصدارات (**VCS**): مثل Mercurial، GitHub، Git، وSubversion، مع واجهة رسومية مدمجة.

-تصحيح الأخطاء (**Debugging**) واختبار الكود: فيه Debugger متقدم مع دعم كامل لـ breakpoints، والمراقبة (Watches)، وتتبع الأداء (Profiling).ويحتوي على دعم JUnit وTestNG لاختبار الوحدات.

-إعادة هيكلة الكود (**Refactoring**): فيه أدوات قوية لإعادة تنظيم الكود دون التأثير على وظائفه.كما يتميز بتحسين هيكل الشيفرة .

-دعم قوي لتطوير الويب: حيث يدعم HTML, CSS, JavaScript, React, Angular، وغيرها. ولديه تكامل سهل مع خوادم مثل Jetty، Tomcat، وSpring Boot.

-دعم شامل لـ **Spring & Spring Boot**: أدوات مدمجة لتكوين المشاريع والتعامل مع Beans وAnnotation

Marketplace- غني بالإضافات: فيه آلاف الإضافات (Plugins) التي توسع قدرات الـ IDE حسب حاجتك.

-نسختان: Community Edition (مجانية): مفتوحة المصدر. و Ultimate Edition (مدفوعة): تحتوي على ميزات متقدمة لتطوير الويب، المؤسسات، ودعم قواعد البيانات. (3)

3.3 لغات البرمجة والتقنيات المستعملة :

1.3.3 لغة HTML :

HTML هي اختصار لـ Hyper Text Markup Language، وتعني "لغة ترميز النص الفائق". تُعد HTML اللغة القياسية المستخدمة لإنشاء صفحات الويب، كما تُستخدم أيضًا في تطوير تطبيقات هجينة للهواتف المحمولة، بالإضافة إلى تطبيقات تعمل على أنظمة التشغيل المختلفة مثل ويندوز، لينكس، macOS، وغيرها.تُستخدم HTML لبناء الهيكل الأساسي لصفحات الويب التي يتم عرضها في المتصفحات مثل: Firefox، Safari، Google Chrome وغيرها. حيث تعتمد على مجموعة من الأوامر التي يفهمها المتصفح ليقوم بعرض المحتوى للمستخدم.ومع تطور التكنولوجيا، توسّع استخدام HTML ليشمل تطوير تطبيقات الأجهزة الذكية، مما يجعلها لغة مهمة في عالم تطوير الويب والتطبيقات الحديثة. (4)

2.3.3 لغة CSS :

CSS هي اختصار لـ Cascading Style Sheets، وتُعد لغة وصفية تُستخدم لإضفاء الطابع الجمالي والتصميمي على صفحات الويب، مما يمنح كل موقع مظهره الفريد الذي يميّزه عن غيره.تُعتبر CSS الرفيقة الأساسية للغة HTML، فبينما تُستخدم HTML لبناء الهيكل العام لصفحة الويب (النصوص، الصور، الروابط... إلخ)، تُستخدم CSS لتحديد الشكل والتنسيق. وبدون CSS، ستبدو صفحات الإنترنت كصفحات نصية بسيطة ذات خلفيات بيضاء.ظهرت لغة CSS لأول مرة عام 1996 على يد اتحاد شبكة الويب العالمية (W3C)، وذلك لمواجهة بساطة تصميم الصفحات في المتصفحات القديمة، التي كانت تعرض المحتوى بشكل بدائي جدًا دون أي عناصر تصميمية.ومع تطورها، أصبحت CSS توفر إمكانيات هائلة في تصميم المواقع، مثل:تحديد الخطوط والهوامش والمسافات بين العناصر,تغيير الألوان والخلفيات، وتحديد حجم العناصر ومواقعها,تحريك العناصر وإنشاء تأثيرات مرئية وحركية جذابة,تمكين المطوّر من تطبيق تصاميم مرنة ومبتكرة تتناسب مع الهواتف والأجهزة المختلفة و اليوم، تُعد CSS جزءًا أساسيًا لا غنى عنه في تطوير أي موقع ويب عصري. (4)

3.3.3 لغة JAVASCRIPT :

JavaScript هي لغة برمجة عالية المستوى تُستخدم بشكل أساسي في متصفحات الويب لإنشاء صفحات أكثر تفاعلًا وديناميكية. تُشرف على تطويرها حاليًا لجنة 39TC التابعة لمنظمة ECMA International المتخصصة في وضع المعايير التقنية.في بداياتها، كانت JavaScript موجّهة للمبرمجين الهواة، إلا أن الاهتمام بها تزايد بشكل كبير مع مرور الوقت، خاصة بعد إدخال تقنيات حديثة مثل تقنية AJAX، التي ساعدت في تحسين تجربة المستخدم من خلال تسريع التفاعل بين الخادم (Server) والعميل (Client).اليوم، تُعد JavaScript من اللغات الأساسية في تطوير الويب، وتُستخدم في:إنشاء صفحات ويب تفاعلية وسريعة الاستجابة,تطوير تطبيقات الويب المتقدمة,برمجة الألعاب الإلكترونية القائمة على المتصفح,بناء تطبيقات تعتمد على واجهات برمجية (APIs) متقدمة وتتميز JavaScript بدعمها الكامل من جميع المتصفحات الحديثة دون الحاجة لأي إضافات أو مكونات خارجية، مما يجعلها واحدة من أكثر لغات البرمجة استخدامًا في عالم الويب. (5)

4.3.3 اطار العمل **BOOTSTRAP** :

Bootstrap هو إطار عمل (Framework) مجاني ومفتوح المصدر، مخصص لمطوري واجهات المستخدم (Front-End Developers). يهدف إلى تسهيل وتسريع عملية إنشاء مواقع ويب متجاوبة تتناسب مع مختلف أحجام الشاشات والأجهزة.تم تطوير Bootstrap عام 2010 على يد مهندسي شركة Twitter، وسرعان ما أثبت كفاءته وانتشر على نطاق واسع. يُستخدم حاليًا في العديد من المواقع والتطبيقات الشهيرة مثل LinkedIn وSpotify، بالإضافة إلى العديد من قوالب WordPress.يتيح Bootstrap للمطورين إنشاء صفحات HTML أنيقة ومتناسقة بفضل ما يوفره من: عناصر تصميم جاهزة مثل: الأزرار، الجداول، القوائم، الشرائح (Carousels)، النوافذ المنبثقة (Modals)، وغيرها.نظام شبكي (Grid System) لتصميم واجهات متجاوبة بسهولة.مكتبة ضخمة من الأصناف الجاهزة (CSS Classes) التي توفر الوقت والجهد.تكامل سلس مع JavaScript لإضافة تفاعلية على الصفحة دون الحاجة لكتابة الكثير من الكود.يُعتبر Bootstrap مناسبًا للمبتدئين والمحترفين على حد سواء، ولا يتطلب خبرة برمجية متقدمة. ومع ذلك، من المفيد أن يمتلك المستخدم خلفية أساسية في HTML وCSS وJavaScript، بالإضافة إلى فهم لمبادئ التصميم المتجاوب (Responsive Design). (6)

5.3.3 اطار العمل **ANGULAR** :

Angular هو إطار عمل (Framework) مفتوح المصدر لتطوير تطبيقات الويب وتطبيقات الأجهزة الذكية، تم تطويره وصيانته من قبل شركة Google. يُستخدم لإنشاء تطبيقات أحادية الصفحة (Single Page Applications - SPA) بطريقة منظمة وقابلة للتوسعة، باستخدام لغة TypeScript (وهي نسخة مطوّرة من JavaScript).تم إطلاق الإصدار الأول من AngularJS عام 2010، ثم أعيد بناء الإطار بالكامل وصدر Angular 2 عام 2016، ومنذ ذلك الحين يُشار إليه ببساطة باسم Angular.ومن اهم مميزاته :

- مبني على **TypeScript**: يوفّر ميزات متقدمة مثل التحقق من الأنواع (Type Checking) وواجهة أكثر تنظيماً لكتابة الكود.

- تطوير تطبيقات أحادية الصفحة (**SPA**): يسمح بإنشاء تطبيقات ديناميكية وسريعة، يتم تحميل الصفحة مرة واحدة ويتم تحديث المحتوى دون إعادة تحميل الصفحة بالكامل.

- مبدأ المكوّنات (**Components**): يُقسم التطبيق إلى وحدات صغيرة (مكونات) سهلة الصيانة وإعادة الاستخدام.

- الربط بين البيانات (**Data Binding**): يدعم الربط ثنائي الاتجاه (Two-Way Data Binding) لتحديث الواجهة تلقائيًا عند تغير البيانات.

- توجيه (**Routing**): يتيح التنقل بين الصفحات داخل التطبيق دون الحاجة لإعادة تحميل الصفحة.

- أدوات قوية للاختبار (**Testing**): مدمج فيه دعم لاختبار الوحدات (Unit Testing) واختبار التكامل.

- حقن التبعيات (**Dependency Injection**): يسهّل إدارة الكود وتحسين الأداء من خلال تمرير الخدمات بشكل ديناميكي.

- دعم قوي للمجتمع والتحديثات: مدعوم من Google مع مجتمع نشط وتحديثات منتظمة.

CLI - (واجهة أوامر قوية): يوفّر أداة Angular CLI لإنشاء المشاريع وإدارتها بسرعة وسهولة.

- متعدد المنصات: يمكن استخدام Angular لتطوير تطبيقات ويب، تطبيقات موبايل باستخدام Ionic، وحتى تطبيقات سطح المكتب. (7)

6.3.3 اطار العمل **Spring Boot** :

Spring Boot هو إطار عمل (Framework) مفتوح المصدر مبني على منصة Spring الشهيرة بلغة Java. يُستخدم لتطوير تطبيقات الويب والخدمات الخلفية (Back-End APIs) بسهولة وسرعة، ويهدف إلى تقليل الإعدادات والتعقيد اللازم لتشغيل تطبيق Spring.تم إطلاق Spring Boot من قبل Pivotal (حاليًا جزء من VMware) لتبسيط تطوير تطبيقات Java Enterprise، وذلك من خلال توفير إعدادات تلقائية (Auto Configuration)، وخوادم مضمنة (Embedded Servers)، واعتماد نهج "الجاهزية للإنتاج".ويتميز ب:

- إعداد تلقائي (**Auto Configuration**): يقلل من الإعداد اليدوي لملفات XML أو Java Config من خلال توفير إعدادات ذكية حسب المكتبات المستخدمة.

- تشغيل سريع (**Standalone**): يمكن تشغيل التطبيق مباشرة كـ JAR بدون الحاجة إلى خادم خارجي (مثل Tomcat أو Jetty)، لأنه يأتي مدمجًا بخوادم داخلية.

- إدارة التبعيات (**Dependency Management**): يوفر تكاملًا سهلًا مع Maven وGradle لتسهيل إدارة المكتبات.

- هيكلية واضحة للتطبيق: يعتمد على مفهوم الطبقات (Layers)، مما يسهّل تنظيم الكود وفصله (Controllers, Services, Repositories).

- سهولة بناء **RESTful APIs**: يجعل من السهل جدًا إنشاء واجهات برمجية REST متكاملة مع قواعد البيانات والتعامل مع JSON.

- أمان مدمج (**Spring Security**): يوفر آليات أمان جاهزة للتكامل مع OAuth2, JWT, Keycloak، وغيرها.

- تكامل ممتاز مع قواعد البيانات: يدعم JDBC ,JPA, Hibernate، ويحتوي على طبقة Spring Data لتسهيل التعامل مع قواعد البيانات.

- سهولة الاختبار (**Testing**): مدمج فيه أدوات قوية لاختبار وحدات الكود (Unit Testing) واختبار الوظائف (Integration Testing).

- مناسب للتطبيقات السحابية (**Cloud Ready**): متوافق مع Spring Cloud لإنشاء خدمات صغيرة (Microservices) قابلة للتوسيع بسهولة.

- وثائق ممتازة ومجتمع قوي: يتمتع Spring Boot بتوثيق رسمي شامل ومجتمع كبير من المطورين. (8)

4.3 Keycloak (نظام إدارة الهوية و الوصول) :

1.4.3 تعريف :

Keycloak هو نظام مفتوح المصدر لإدارة الهوية والوصول (Identity and Access Management - IAM)، تم تطويره من قبل شركة Red Hat. يهدف إلى تسهيل عمليات المصادقة (Authentication) والتفويض (Authorization) في التطبيقات الحديثة، سواء كانت تطبيقات ويب، تطبيقات جوّال، أو خدمات RESTful ويُوفر Keycloak خدمات مثل: تسجيل الدخول الأحادي (SSO) , إدارة المستخدمين والمجموعات , دعم بروتوكولات OpenID، OAuth 2.0 Connect، وSAML , الربط مع LDAP وActive Directory , تسجيل الدخول الاجتماعي (Google, Facebook...) و إدارة الأدوار والصلاحيات

2.4.3 المفاهيم الاساسية في Keycloak :

- **Realm** (العالم) : هو نطاق أو مساحة تحتوي على مجموعة من المستخدمين، العملاء (clients)، الأدوار، وسياسات الوصول. يمكن اعتبار كل Realm كبيئة مستقلة (مثلاً: Realm لبيئة التطوير وآخر للإنتاج).
- **Client** (العميل) : هو أي تطبيق يحتاج إلى المصادقة باستخدام Keycloak. يمكن أن يكون: تطبيق ويب (مثل Angular أو React) , خدمة REST API (مثل Spring Boot) او تطبيق جوّال , كل Client يتم تسجيله داخل Realm ويُعطى إعدادات خاصة مثل نوع الوصول، التوكنات، والسياسات.
- **User** (المستخدم) : يمثل الشخص الذي يسجل الدخول. يمكن إنشاؤه يدويًا، أو من خلال التسجيل الذاتي، أو المزامنة مع LDAP.
- **Role** (الدور) : هو صلاحية أو مجموعة من الصلاحيات تُمنح للمستخدم. يمكن أن تكون أدوار عامة (Realm Roles) أو خاصة بتطبيق معين (Client Roles).
- **Group** (المجموعة) : تُستخدم لتجميع المستخدمين ومنحهم أدوارًا مشتركة.
- **Identity Provider** (مزود الهوية) : يمكن لـ Keycloak أن يتصل بمزود هوية خارجي (مثل Google أو Azure AD) للسماح بتسجيل الدخول من خلاله.

3.4.3 الميزات الاساسية :

- المصادقة والتفويض : Keycloak يُدير المصادقة باستخدام جلسات آمنة وتوكنات JWT. كما يسمح بإدارة صلاحيات دقيقة باستخدام الأدوار والسياسات.
- تسجيل الدخول الأحادي (**SSO**) : يسمح للمستخدم بتسجيل الدخول مرة واحدة للوصول إلى جميع التطبيقات المرتبطة بـ Keycloak دون الحاجة لتكرار تسجيل الدخول.
- تسجيل الدخول الاجتماعي : يدعم التكامل مع Google, Facebook, Twitter وغيرها لتسجيل الدخول السريع.

- تسجيل الخروج الموحد (**Single Logout**) : عند تسجيل الخروج من أحد التطبيقات، يتم تسجيل الخروج من كل التطبيقات المرتبطة.
- التخصيص : يمكن تخصيص صفحة تسجيل الدخول، البريد الإلكتروني، شاشة التسجيل، والمزيد باستخدام قوالب Freemarker و.Themes
- دعم البروتوكولات القياسية : OAuth , OpenID Connect (OIDC) 2.0 و SAML 2.0

4.4.3 بنية **Keycloak** :

يتكوّن Keycloak من مكونات رئيسية:

- **Keycloak Server :** هو التطبيق الرئيسي الذي يُشغّل لوحة التحكم ويوفّر API المصادقة والتفويض.
- **Keycloak Admin Console :** واجهة ويب رسومية لإدارة المستخدمين، الأدوار، العملاء، السياسات... إلخ.
- **Keycloak Account Console :** واجهة يستخدمها المستخدم النهائي لتعديل معلوماته الشخصية وكلمات المرور والروابط الاجتماعية

5.4.3 انواع التوكنات المستخدمة :

Keycloak يعتمد على JWT ويُصدر أنواعًا مختلفة من التوكنات :

- **Access Token :** يُستخدم للوصول إلى الموارد المحمية (مثل API).
- **Refresh Token :** يُستخدم للحصول على Access Token جديد بعد انتهاء صلاحيته دون الحاجة لإعادة تسجيل الدخول.
- **ID Token :** يحتوي على معلومات تعريفية عن المستخدم (اسم، بريد...)

6.4.3 التكامل مع التطبيقات :

- مع تطبيقات الويب (مثل **Angular) :** استخدام مكتبة Keycloak JS أو مكتبة keycloak-angular,

تسجيل الدخول، حفظ التوكن، توجيه المستخدم حسب الأدوار

- مع **Spring Boot** : استخدام مكتبة spring-boot-starter-oauth2-resource-server

أو spring-security-oauth2 لتأمين الـ APIs , تحليل التوكن المرسل من Angular والتحقق من الصلاحيات

7.4.3 Keycloak Admin API :

يوفر Keycloak واجهة REST قوية لإدارة :

- إنشاء مستخدم

- تعيين دور

- إرسال بريد إعادة تعيين كلمة المرور

- إدارة التوكنات

و يمكن استخدام RestTemplate أو WebClient في Java للتواصل مع هذا الـ API.

8.4.3 الية العمل عند تسجيل الدخول :

- الخطوة الاولى : يرسل المستخدم بياناته (اسم المستخدم/الإيميل وكلمة المرور).

- الخطوة الثانية : يقوم Keycloak بالتحقق من الهوية.

- الخطوة الثالثة : إذا كانت البيانات صحيحة:

- الخطوة الرابعة : يصدر Access Token وID Token وRefresh Token.

- الخطوة الخامسة : يعيد التوجيه إلى التطبيق.

- الخطوة السادسة : يستخدم التطبيق الـ Access Token للوصول إلى الـ Backend المحمي

9.4.3 الأمان :

Keycloak يُوفر:

- تشفير التوكنات

- دعم FA2 (المصادقة الثنائية)

- حماية ضد هجمات CSRF وXSS

- إدارة صلاحيات دقيقة (سياسات الوصول)

10.4.3 استخدامات Keycloak في المشاريع :

- بوابة المستخدمين لتطبيقات التجارة الإلكترونية

- تسجيل دخول موحد في أنظمة الشركات

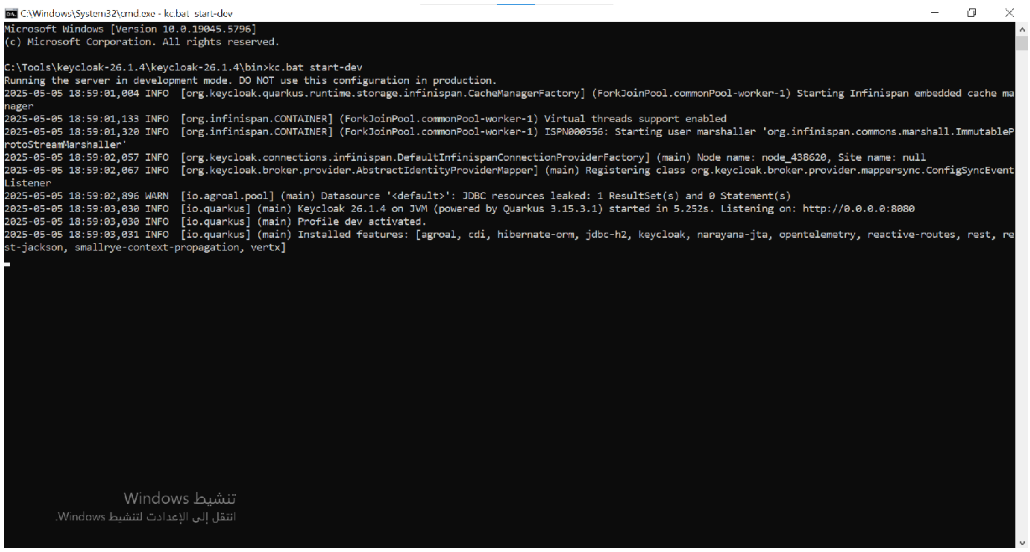
- إدارة المستخدمين في المنصات التعليمية

- تأمين واجهات REST APIs. (9)

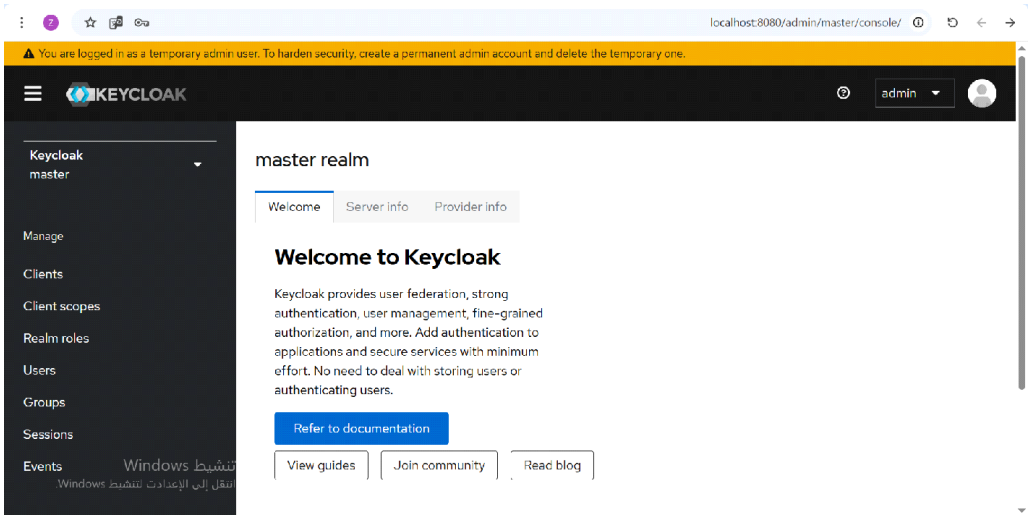
11.4.3 استخدام Keycloak في اطارنا البرمجي (تسيير الحسابات في برامج الويب) :

بعد تشغيل Keycloak :

- الدخول الى لوحة التحكم عبر http://localhost:8080

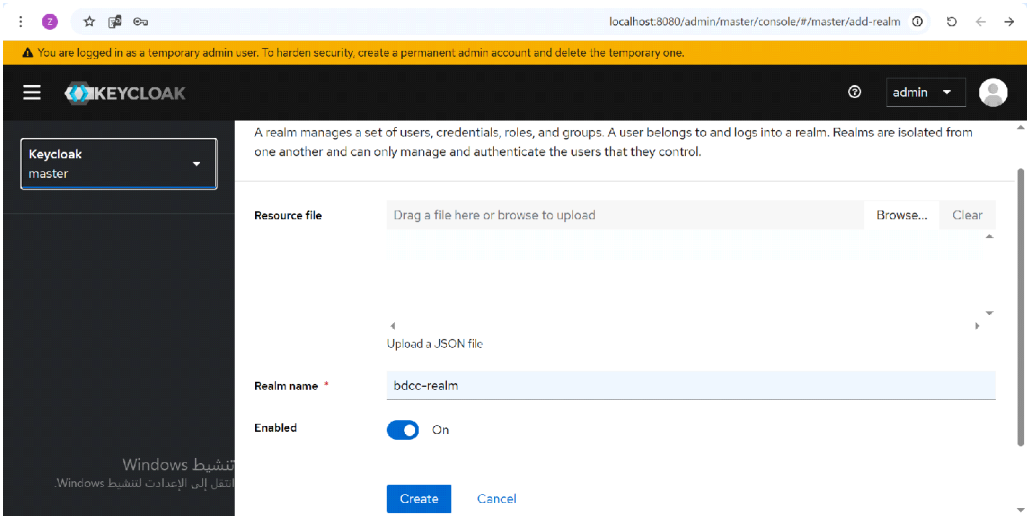


25 : تشغيل Keycloak



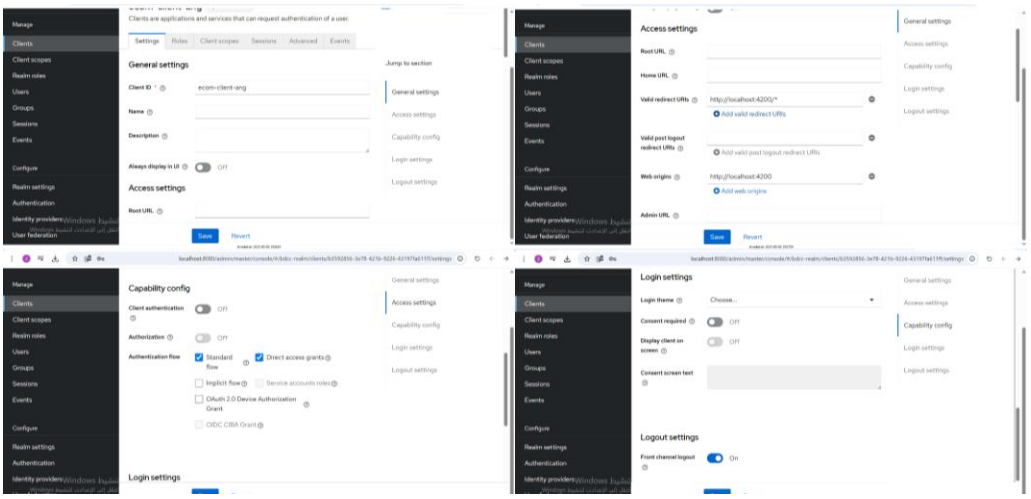
26 : الصفحة الرئيسية ل Keycloak (لوحة التحكم)

- انشاء **Realm** في مشروعنا سميناه : "**bdcc-realm** "



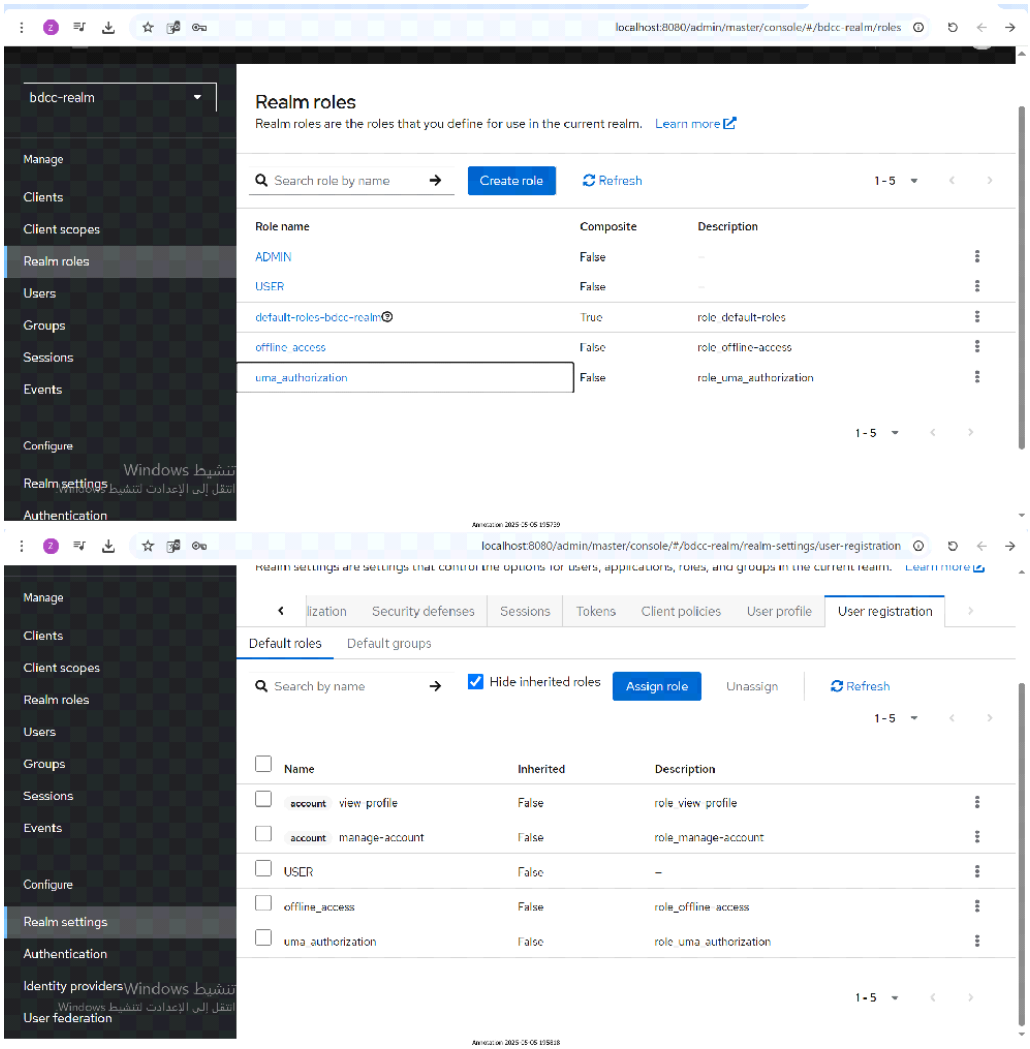
27 : صفحة انشاء Realm

- انشاء **Client** و ضبط الاعدادات الخاصة به كي نتمكن من العمل عليه و لكي تكون البرمجة الخاصة به سلسة و تتم بشكل صحيح في مشروعنا سميناه : " **ecom-client-ang** "



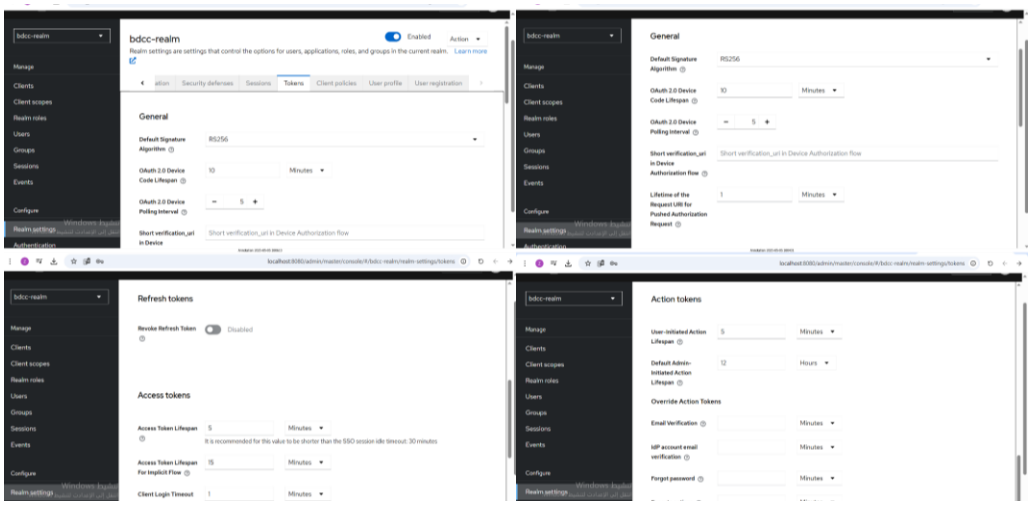
28 : انشاء وضبط اعدادات ال Client

- انشاء الادوار التي نحتاجها في مشروعنا نحتاج الى مسؤول " **ADMIN** " و دور اخر للمستعملين الاخرين على حسب الحاجة هنا حددنا " **USER** " وسيكون هو الدور الافتراضي الذي يعطيه **Keycloak** لاي مستخدم جديد



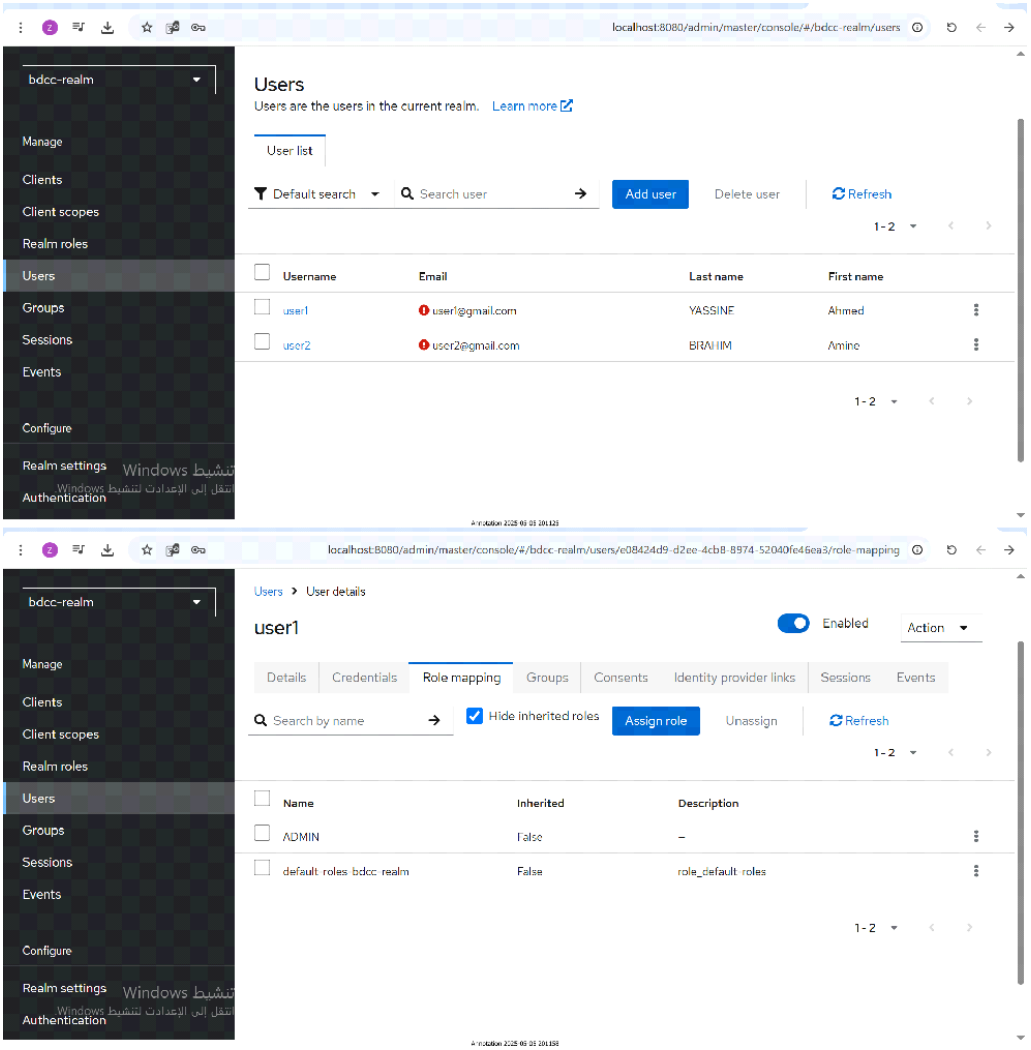
29 : انشاء الادوار المطلوبة وتحديد الدور الافتراضي

- تحديد خصائص التوكن كالمدة الزمنية لصلاحيته و طريقة تشفيره و انواعه وغيرها من الميزات



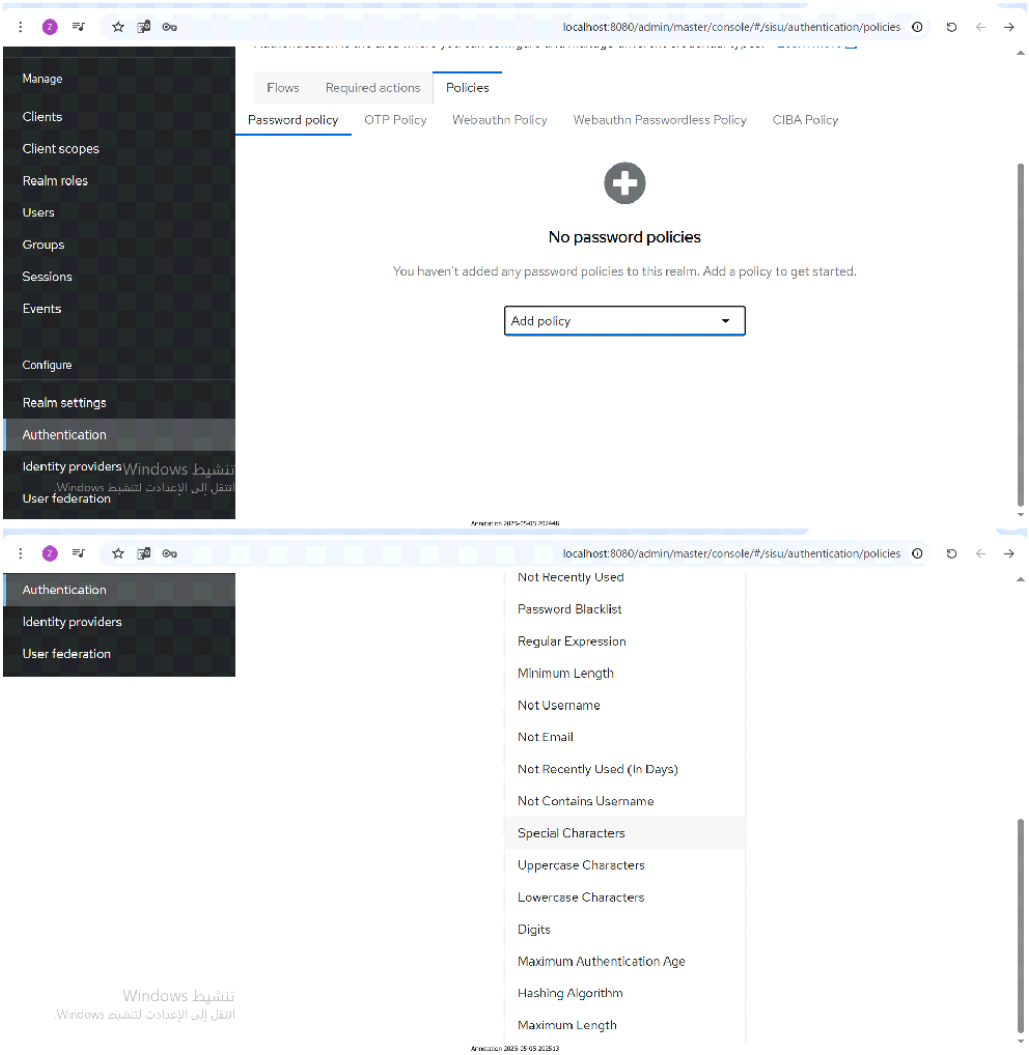
30 : تحديد خصائص التوكن

- تحديد الدور للمستخدمين وذلك عن طريق اختيار حساب المستخدم وبعدها اسناد الدور له



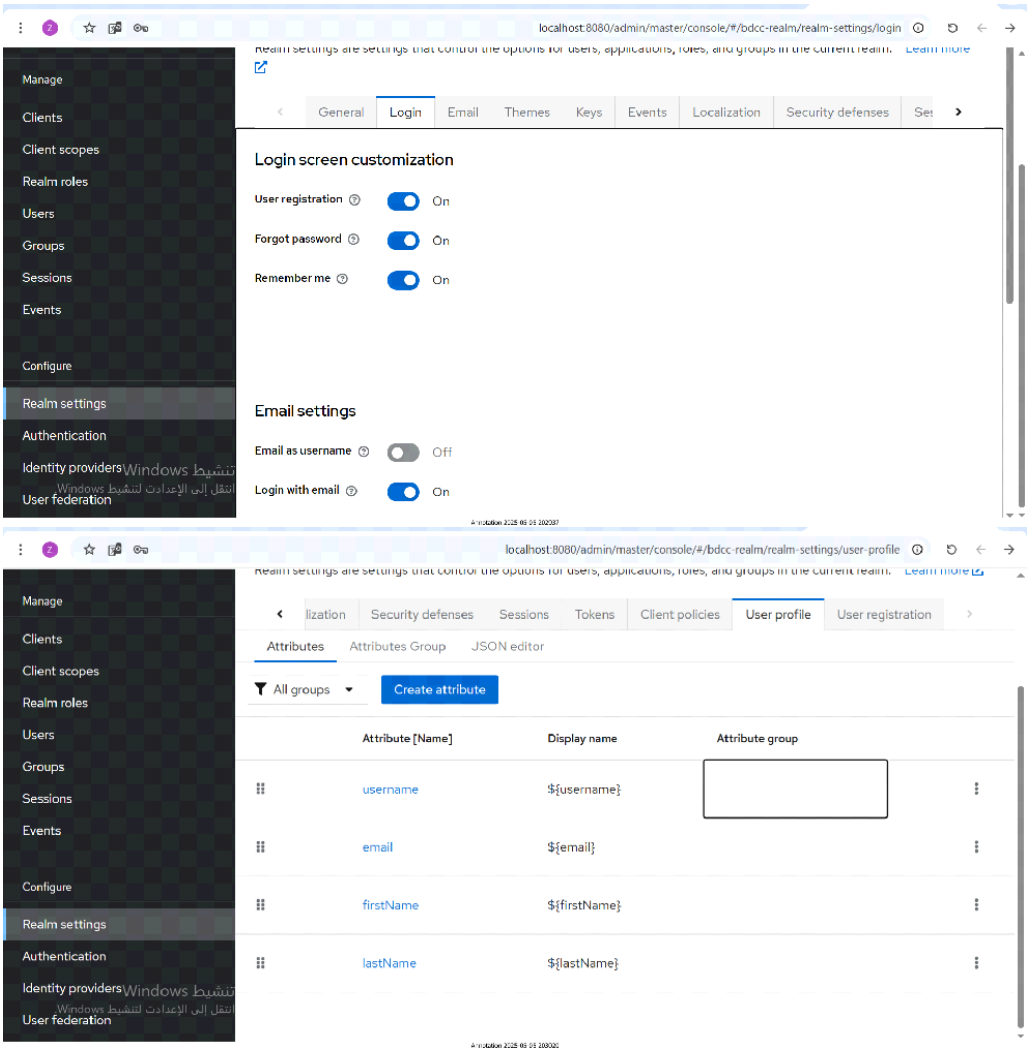
31 : تحديد دور المستخدم

- التحكم في خصائص كلمة المرور كعدد الاحرف اللازمة و الرموز وغيرها



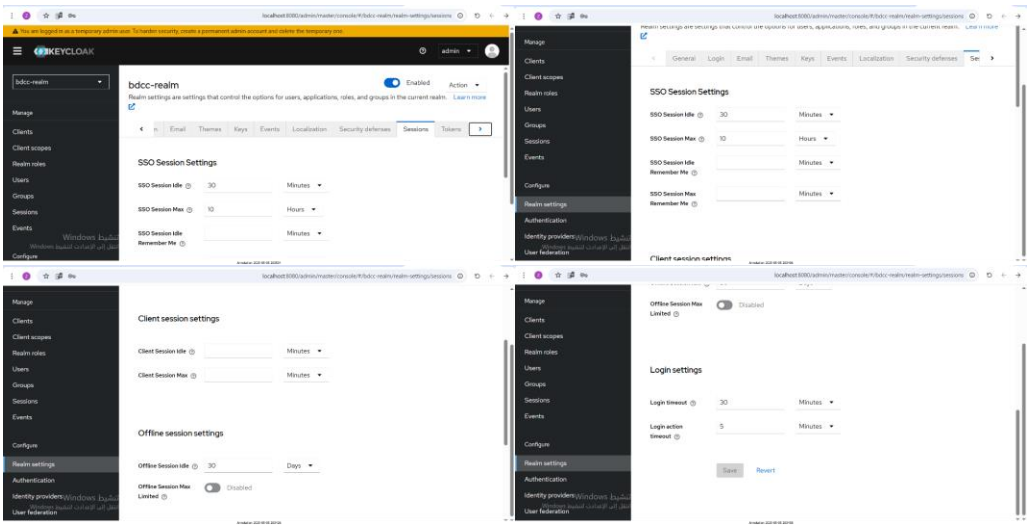
32 : التحكم في خصائص كلمة السر

- التحكم في خصائص تسجيل الدخول وانشاء حساب مثل المعلومات التي يجب ادخالها و خاصية ارجاع كلمة المرور و غيرها



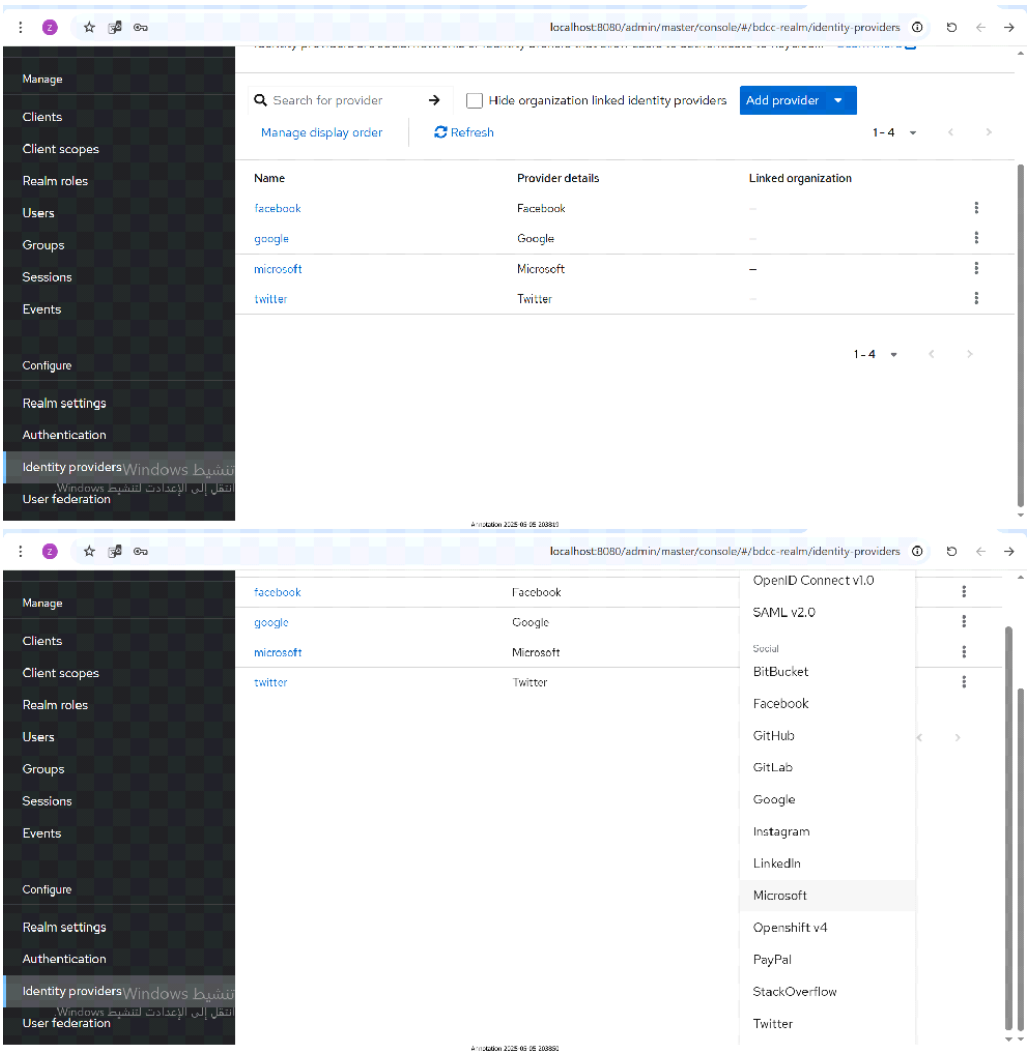
33 : التحكم في خصائص الحساب المطلوبة

- التحكم في خصائص الجلسة كالمدة الزمنية وتسجيل الحسابات من خلالها وغيرها



34 : التحكم في خصائص الجلسة

- امكانية التحكم في المواقع التي يمكن للمستخدم التسجيل منها او ربط حسابه بها مثل فيسبوك و قوقل وغيرها

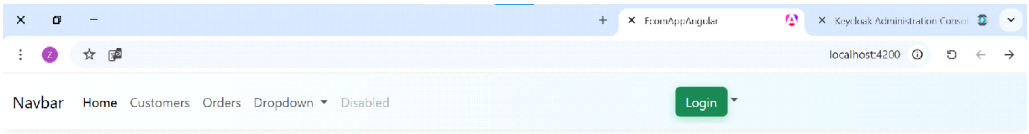


35 : التحكم في الوصول للمواقع العالمية

5.3 واجهات الاطار البرمجي :

1.5.3 واجهة الصفحة الرئيسية :

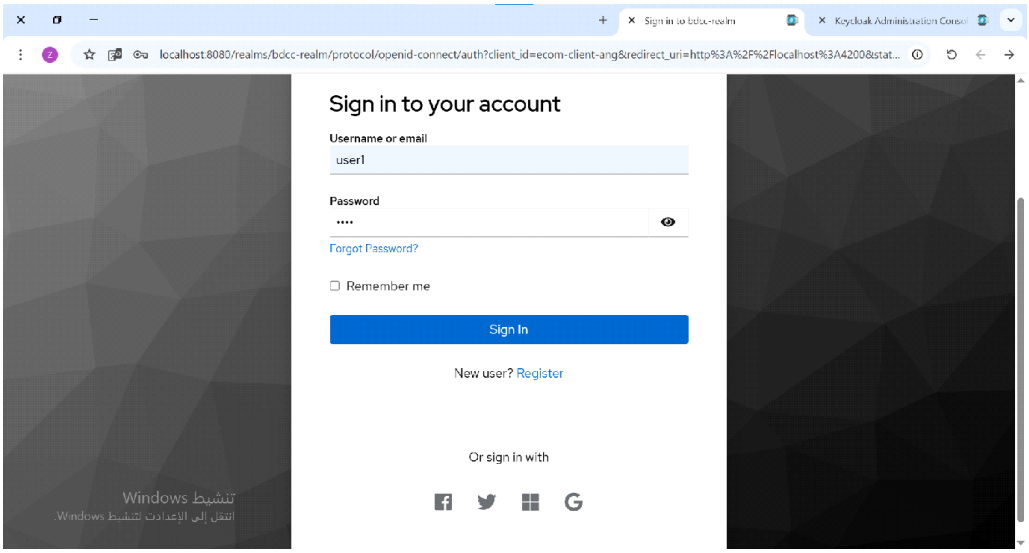
يبين الشكل رقم 36 الواجهة الرئيسية للاطار البرمجي حيث يمر عليها جميع الزوار . حيث من خلالها يمكننا معرفة اسماء المكونات (Component) الخاصة بجميع المستعملين .



36 : الصفحة الرئيسية

2.5.3 واجهة تسجيل الدخول :

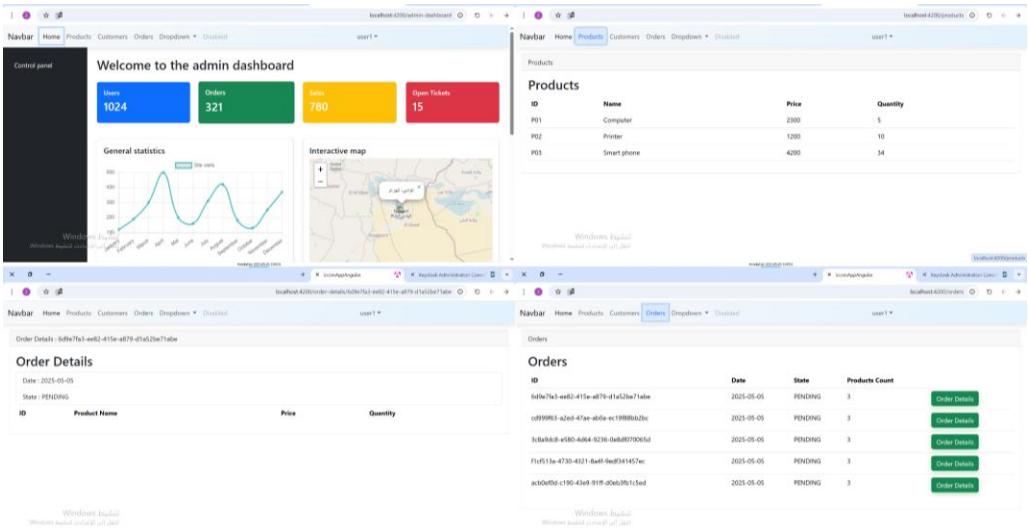
يبين الشكل 37 واجهة تسجيل الدخول بحيث تسمح بتسجيل الدخول للمستعمل سواء مسؤول او اي مستخدم اخر .



37 : صفحة تسجيل الدخول

3.5.3 المكونات الخاصة بالمسؤول :

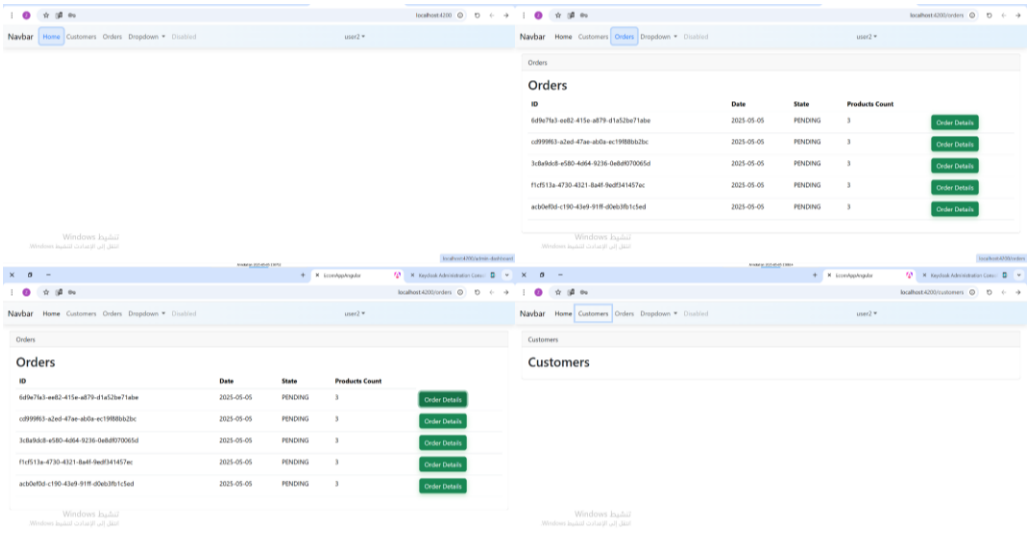
يبين الشكل 38 جميع المكونات التي تظهر للمستخدم هو وحده بحيث لا يمكن لاي مستعمل اخر لا يملك دور المسؤول ان تظهر له هذه المكونات مثل لوحة التحكم و قائمة المنتجات (Products) و تفاصيل العروض وغيرها



38 : المكونات الخاصة بالمسؤول

4.5.3 المكونات الخاصة بالمستخدم :

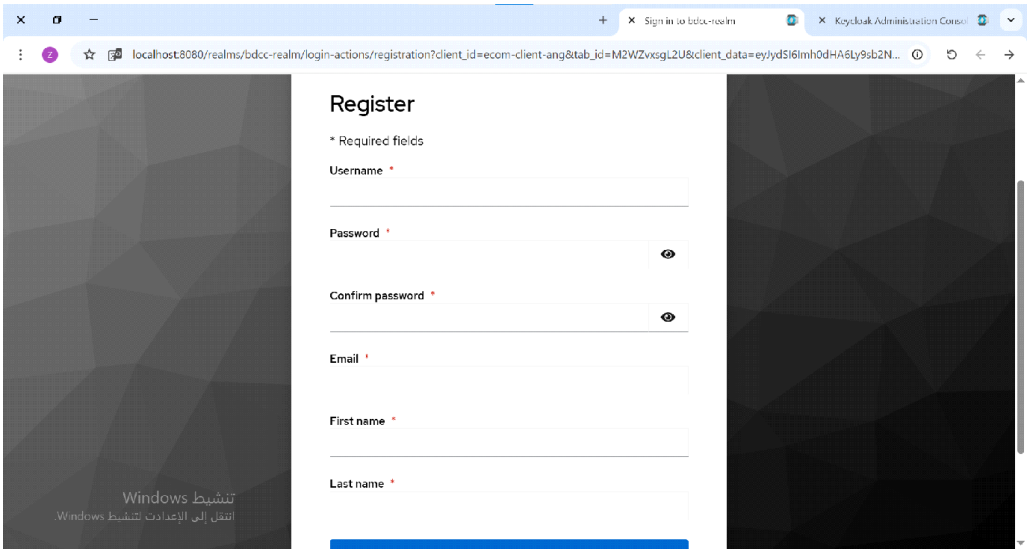
يبين الشكل 39 المكونات التي تظهر لاي مستخدم اخر غير المسؤول لان المسؤول تكون له صلاحيات اوسع في اي اطار برمجي اما عامة المستعملين الاخرين تكون لديهم صلاحيات محدودة اقل حيث في هذا الاطار لا يمكن للمستخدم ان يرى لوحة التحكم او ان يرى المنتجات (Products) ولا يمكنه رؤية تفاصيل العروض (orders-details) بينما يمكنه رؤية العروض وحسب (orders) .



39 : المكونات الخاصة بالمستخدم

5.5.3 واجهة انشاء حساب :

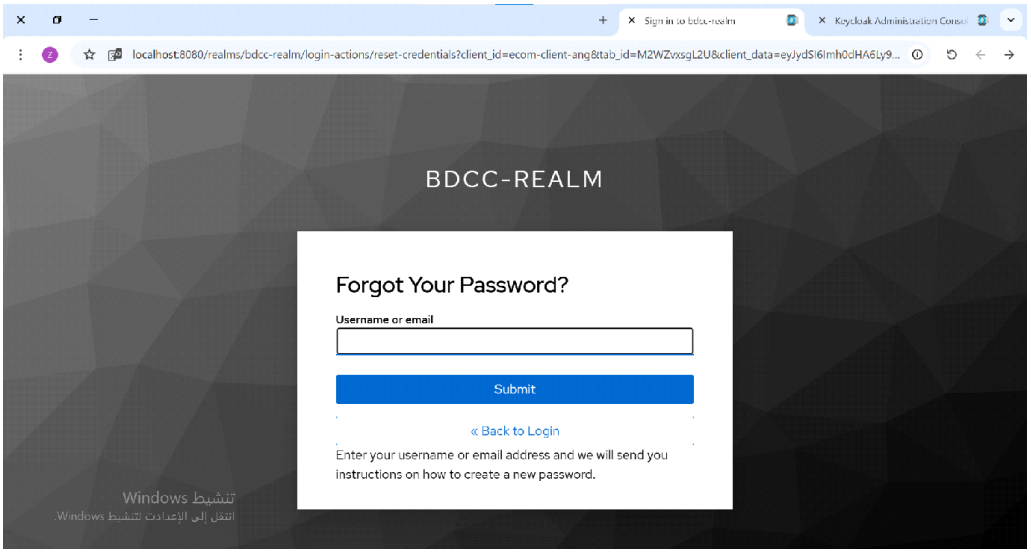
يبين الشكل 40 الواجهة المستعملة لانشاء حساب .



40 : صفحة انشاء حساب

6.5.3 واجهة ارجاع كلمة السر :

يبين الشكل 40 الواجهة المستعملة لارجاع كلمة السر .



41 : صفحة ارجاع كلمة السر

6.3 الخاتمة :

وفي هذا الفصل قمنا بعرض الواجهات والصفحات الرئيسية الخاصة بإطارنا البرمجي وبيننا جميع الوظائف المتعلقة بها وكيفية القيام بها .

خاتمة عامة :

بعد اكمالنا للفترة المحددة قدمنا هذا العمل المتواضع الذي كان خلاصة لمجهوداتنا المكثفة والمتواصلة والتي فتحت امامنا افاقا كبيرة استطعنا من خلالها انجاز المشروع المطلوب . فالهدف الاساسي منه هو انشاء منصة برمجية (اطار برمجي) لتسيير وادارة الحسابات في برامج الويب ويتكون هذا العمل من مرحلتين : المرحلة النظرية والتي تشمل الفصلين السابقين , و المرحلة العملية التي تناولناها في الفصل الاخير . وبالرغم من صعوبة تركيب Angular مع spring bootو Keycloak الا اننا توفقنا في تخطي عتبة كبيرة و هذا يفتح لنا افاقا مستقبلية كبيرة في مجال الامن المعلوماتي .

وفي الاخير نتمنى اننا وفقنا في انجاز هذا النظام , كما ان توقفنا عند هذا الحد لا يعني تمام الدراسة في هذا المشروع , بل العكس فهو يمثل انطلاقة لاتمام انظمة اخرى مستقبلية اكثر عمقا ونفعا . وعليه نقترح التحسينات التالية :

- دعم اللغات المتعددة
- نظام متكامل لاسترجاع كلمة المرور
- نظام تنبيهات لتسجيلات الدخول

المراجع

1. لغة النمذجة الموحدة (it-solutions.center) UML. [متصل]
2. الدليل البسيط لرسم UML تمثيليا ونمذجة قاعدة البيانات (microsoft.com). [متصل]
3. JetBrains. IntelliJ IDEA Features. https://www.jetbrains.com/idea/features/. [متصل]
4. *Duckett, Jon. HTML and CSS: Design and Build Websites. Wiley, 2011.*
5. *Flanagan, David. JavaScript: The Definitive Guide, 7th Edition. O'Reilly Media, 2020.*
6. Jacob. Bootstrap: Responsive Web Development. Packt Publishing, 2013. Spurlock.
7. Angular Team. (2024). Introduction to Angular. Angular Docs. Retrieved from https://angular.io/docs. [متصل]
8. Spring Team. (2024). Spring Boot Reference Documentation. Retrieved from https://spring.io/projects/spring-boot. [متصل]
9. Keycloak Team. (2024). Keycloak Documentation. Retrieved from https://www.keycloak.org/documentation.html. [متصل]