الجمهورية العربية السورية
المعهد العالي للعلوم التطبيقية والتكنولوجيا
قسم المعلوميات
العام الدراسي 2023/2024

مشروع تخرج

أعد لنيل درجة الإجازة في هندسة البرمجيات والذكاء الصنعي

توليد مخططات نماذج الأعمال BPMN انطلاقاً من التوصيف النصي للعمليات باستخدام النماذج اللغوية الكبيرة LLM

تقديم

محمد صالح التركي

إشراف

د. عمر حمدون

ما. محمد بشار دسوقي

23/8/2024

أهدي هذا العمل إلى من يقدر العمل.

There is an AI for that ….

كلمة شكر

أتقدم بالشكر إلى [SELECT * FROM People;]

محمد التركي

الفصل الأول

التعريف بالمشروع

يتضمن هذا الفصل التعريف بالمشروع ومتطلباته.

1.1- مقدمة

في العصر الحديث لإدارة العمليات، تعد القدرة على نمذجة عمليات الأعمال (business processes) وتصورها بكفاءة أمرًا بالغ الأهمية للمؤسسات التي تسعى إلى تحسين إنتاجيتها. وقد ظهر معيار نموذج عمليات الأعمال وترميزها (BPMN) كإطار مقبول على نطاق واسع لوصف عمليات الأعمال بتنسيق رسومي. ومع ذلك، فإن إنشاء نماذج BPMN قد يستغرق وقتًا طويلاً ويتطلب معرفة متخصصة، والتي يمكن أن تعمل كحاجز لغير الخبراء، مما يدعو إلى البحث عن حل لهذا التحدي، عن طريق إيجاد أسلوب لإنشاء هذه المخططات بطريقة مأتمتة.

2.1- هدف المشروع

يندرج عملنا في هذا المشروع ضمن سياقين اثنين: (1) الأول نتطرق فيه للمسألة من منظور الذكاء الصنعي، حيث نحاول توظيف التطورات الأخيرة في مجال نماذج اللغات الكبيرة (LLMs) للوصول لمقاربة قادرة على بناء مخططات BPMN انطلاقاً من التوصيف النصي للعمليات، (2) والثاني نتطرق فيه إلى بناء تطبيق برمجي يستفيد من المقاربة المقترحة في طرح تطبيق مفيد عملياً، حيث جرت مراعاة الأسس والمبادئ المتعارف عليها في هندسة البرمجيات وصولاً لتطبيق قابل للتوسع وسهل الصيانة. نبيّن في الفقرات التالية المتطلبات الوظيفية وغير الوظيفية لهذا التطبيق.

3.1- المتطلبات الوظيفية

يجب أن يقدم النظام للمستخدم بما يلي:

1.3.1- السماح بإنشاء حساب جديد ضمن النظام.

2.3.1- السماح بتسجيل الدخول من حساب منشأ مسبقاً.

3.3.1- السماح بإنشاء مشروع جديد.

4.3.1- السماح باستعراض المشاريع الخاصة بالمستخدم.

5.3.1- السماح بدعوة مستخدمين آخرين للمشاركة ضمن المشروع.

6.3.1- السماح بإنشاء مخطط BPMN ضمن مشروع معين.

7.3.1- السماح للمستخدم برسم مخطط BPMN ضمن الواجهة.

8.3.1- توليد مخططات BPMN انطلاقاً من التوصيف النصي للعملية.

9.3.1- توليد تقرير BPMN إنطلاقاً من التوصيف النصي للعملية.

10.3.1- السماح للمستخدم بتعديل المخطط بعد توليده.

11.3.1- السماح للمستخدم بتصدير المخطط الناتج.

12.3.1- حفظ المخطط ضمن النظام، لضمان عدم ضياعه.

4.1- المتطلبات غير الوظيفية

1.4.1- يجب أن يكون النظام آمناً، حيث يسمح فقط للمستخدمين المسجّلين باستخدامه.

2.4.1- يجب أن يوفر النظام واجهات سهلة الاستخدام وجيدة المظهر.

3.4.1- يجب أن يكون الرماز البرمجي قابلاً للتعديل والصيانة.

4.4.1- يجب أن يحقق النظام سرعة جيدة بالاستجابة للطلبات.

5.4.1- يجب أن يكون النظام قابلاً للتوسع.

5.4.1- يجب أن يكون قابلاً للتطوير والنشر المستمر دون الحاجة لأعباء إضافية.

الفصل الثاني

الدراسة النظرية

يوضّح هذا الفصل مجموعة من المفاهيم النظرية المستخدمة ضمن العمل.

BPMN -1.2

عمليات الأعمال (Business Process) هي سلسلة من الأنشطة أو المهام المنظّمة التي يقوم بها الأفراد أو الأنظمة داخل منظّمة معيّنة أو ضمن عدة منظّمات لتحقيق هدف تنظيمي محدَّد أو تقديم خدمة أو منتج معين [1].

نموذج (BPMN) هو معيار لنمذجة عمليات الأعمال يوفر ترميزًا بيانيًا لتحديد عمليات الأعمال ضمن مخطط انسيابي (flowchart diagram). إن الهدف من BPMN هو دعم نمذجة عمليات الأعمال لكل من المستخدمين الفنيّين ومستخدمي الأعمال، من خلال توفير ترميز سهل الفهم لمستخدمي الأعمال قادر على تمثيل دلالات العمليّات المعقَّدة [1].

تم تصميم BPMN ليكون مفهومًا بسهولة من قبل جميع أصحاب المصلحة. ويشمل ذلك محللي الأعمال الذين يقومون بإنشاء وتحسين العمليات، والمطورين الفنيّين المسؤولين عن تنفيذها، ومديري الأعمال الذين يراقبونها ويديرونها. وبالتالي، تعمل BPMN كلغة مشتركة، وتسد فجوة الاتصال التي تحدث بشكل متكرِّر بين تصميم عمليات الأعمال وتنفيذها [1].

BPMN NOTATIONS/SHAPES -2.2

إن الهدف الرئيسي من تطوير BPMN هو إنشاء تدوين (Notation) بسيط وسهل الفهم لإنشاء نماذج عمليات الأعمال، مع توفير الدلالات والآليات الأساسية للتعامل مع التعقيد الكامن في عمليات الأعمال. حيث أن النهج المتبع للتعامل مع هذين المتطلبين المتضاربين هو تنظيم الجوانب الرسومية للتدوين في فئات محددة. يوفر هذا مجموعة صغيرة من فئات التدوين حتى يتمكن قارئ مخطط BPMN من التعرف بسهولة على الأنواع الأساسية للعناصر وفهم المخطط [1].

نوضح في الفقرات التالية الفئات الرسومية لمخططات BPMN.

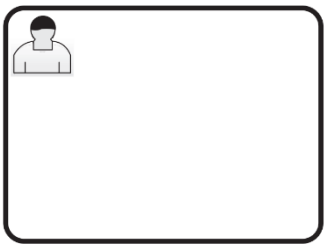
1.2.2- نمذجة المهام ضمن BPMN (Tasks)

تمثل المهمة نشاطاً وحيداً يتم تنفيذه ضمن عملية الأعمال وهي العنصر الأساسي في نموذج BPMN، إذ تمثل المهمة خطوة واحدة ضمن العملية الكلية [1].

بعض أنواع المهام:

- مهمة مستخدم (User task):

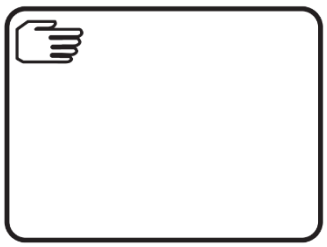
هي مهمة أو سير عمل (workflow) نموذجية يقوم فيها شخص ما بأداء المهمة بمساعدة تطبيق برمجي [1].



صورة **1**: كيفية تمثيل مهمة المستخدم **(User task)** ضمن **BPMN**.

- المهمة اليدوية (Manual task):

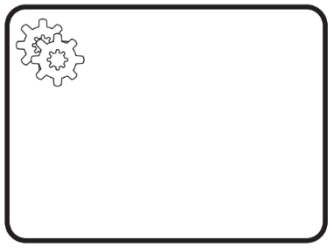
هي مهمة من المتوقع تنفيذها دون مساعدة أي محرك تنفيذ أو تطبيق [1].



صورة **2**: كيفية تمثيل المهمة اليدوية (**Manual task**) ضمن **BPMN**.

- مهمة الخدمة (Service task):

هي مهمة تستخدم خدمة من نوع معين، يمكن أن تكون خدمة ويب أو تطبيقًا آليًا [1].



صورة **2**: كيفية تمثيل مهمة الخدمة (**Service task**) ضمن **BPMN**.

2.2.2- نمذجة التدفقات ضمن BPMN (Flows)

تُمثل التدفقات سير تسلسل الأنشطة (المهام والأحداث) ضمن العملية [1]، نوضح في الفقرات التالية أنواع التدفقات ضمن BPMN.

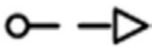
- تدفق التتابع (Sequence Flow):

يُستخدم لإظهار الترتيب (التسلسل) الذي ستُنفذ به الأنشطة في مخطط العملية [1].

صورة **3**: كيفية تمثيل تدفق التتابع (**Sequence flow**) ضمن **BPMN**.

- تدفق الرسائل (Message Flow):

يُستخدم لإظهار تدفق الرسائل بين مشاركين مختلفين في العملية (كيانات الأعمال أو أدوار الأعمال) الذين يرسلونها ويتلقونها [1].



صورة **3**: كيفية تمثيل تدفق الرسائل (**Message flow**) ضمن **BPMN**.

3.2.2- نمذجة الأحداث ضمن BPMN (Events)

تُشير الأحداث إلى شيء يحدث خلال سير العملية، تُستخدم لالتقاط وتمثيل الحوادث التي تؤثر على تدفق العملية، نوضح في الفقرات التالية أنواع الأحداث ضمن BPMN [1].

- حدث البداية (Start event):

تُشير أحداث البدء إلى بداية عملية أو عملية فرعية ولا تكون مرتبطة بتدفق تسلسل وارد. يمكن أن تحتوي العملية الرئيسية على أكثر من حدث بدء واحد، لكن العملية الفرعية تحتوي على حدث بدء واحد فقط [1].

صورة **4**: كيفية تمثيل حدث البداية (**Start event**) ضمن **BPMN**.

- الحدث الوسيط (Intermediate event):

تشير الأحداث الوسيطة إلى شيء يحدث أو قد يحدث خلال سير العملية، بين حدث البداية وحدث النهاية [1].

صورة **5**: كيفية تمثيل الحدث الوسيط (**Intermediate event**) ضمن **BPMN**.

- حدث النهاية (End event):

تشير أحداث النهاية إلى المكان الذي ينتهي فيه أحد المسارات في العملية. يمكن أن تحتوي العملية على أكثر من نقطة نهاية واحدة. تنتهي العملية عندما تنتهي جميع المسارات النشطة. لا تحتوي أحداث النهاية على تدفقات تسلسل صادرة [1].

صورة **5**: كيفية تمثيل الحدث النهائي (**End event**) ضمن **BPMN**.

4.2.2- نمذجة البوابات ضمن BPMN (Gateways)

البوابات هي عناصر تتحكم في تدفق العملية من خلال تحديد كيفية تلاقي المسارات أو تباعدها أو تقسيمها ودمجها. تساعد البوابات في اتخاذ القرارات، وإدارة المسارات المتعددة، والتحكم في تدفق الأنشطة بناءً على شروط معينة. تُعد البوابات أساسية لنمذجة منطق العملية المعقّد وضمان قدرة العمليات على التعامل مع سيناريوهات مختلفة [1]، نوضح في الفقرات التالية أنواع البوابات ضمن BPMN.

- بوابة XOR (Exclusive Gateway):

تُستخدم لتوجيه التدفق إلى واحد من عدة مسارات ممكنة بناءً على شرط. يتم اتخاذ مسار واحد فقط [1].

صورة **6**: كيفية تمثيل بوابة **XOR** (**Exclusive Gateway**) ضمن **BPMN**.

- بوابة OR (Inclusive Gateway):

تُستخدم لتوجيه التدفق إلى واحد أو أكثر من عدة مسارات ممكنة بناءً على الشروط. يمكن إتخاذ مسارات متعددة في وقت واحد [1].

صورة **7**: كيفية تمثيل بوابة **OR** (**Inclusive Gateway**) ضمن **BPMN**.

- بوابة AND (Parallel Gateway):

تُستخدم لتقسيم التدفق إلى عدة مسارات متوازية أو لمزامنة عدة مسارات متوازية في تدفق واحد. تُنفذ جميع المسارات في وقت واحد [1].

صورة **7**: كيفية تمثيل بوابة **AND** (**Parallel Gateway**) ضمن **BPMN**.

5.2.2- نمذجة المشاركين ضمن BPMN (Pools & Lanes)

تُستخدم أحواض السباحة (Pools) ومسارات السباحة (Lanes) لتنظيم وتصنيف المشاركين والأدوار المختلفة ضمن عملية الأعمال. تساعد هذه العناصر في توضيح الأدوار والمسؤوليات والتفاعلات بين الكيانات المختلفة المشاركة في العملية [1].

- أحواض السباحة (Pools):

تمثل المشاركين الرئيسيين في العملية، مثل المنظّمات. كل حوض عادةً ما يمثل كيانًا أو منظمة منفصلة تشارك في العملية. تساعد المجمعات في فصل وتمييز المشاركين المختلفين في العملية بصريًا [1].

- مسارات السباحة (Lanes):

تُستخدم مسارات السباحة داخل الحوض لتقسيم العملية بشكل أكبر إلى أدوار أو أقسام أو مجالات وظيفية مختلفة داخل نفس المشارك. تساعد مسارات السباحة في توضيح الدور أو القسم المحدد المسؤول عن كل جزء من العملية [1].

صورة **8**: كيفية تمثيل الأحواض والمسارات (**Pools & Lanes**) ضمن **BPMN**.

3.2- آليات الانتباه (Attention mechanisms)

الانتباه هو آلية في التعلم الآلي والشبكات العصبونية تمكن النماذج من التركيز على أجزاء معينة من بيانات الدخل عند توليد المخرجات. حيث تسمح للنموذج بوزن أهمية المدخلات المختلفة بشكل ديناميكي، مما يعزز قدرته على التقاط العلاقات والتبعيات داخل البيانات، بغض النظر عن المسافة بينها ضمن سلسلة الدخل [2].

مزايا الانتباه:

- تسمح بالحساب المتوازي (Parallelization): على عكس الشبكات العصبونية المتكررة (RNN)، التي تعالج البيانات بشكل متسلسل، تسمح آليات الانتباه بالمعالجة المتوازية للدخل. وهذا يسرع بشكل كبير من الحوسبة ويجعلها أكثر كفاءة، وخاصة بالنسبة لمجموعات البيانات الكبيرة [2].
- التعامل مع التبعيات طويلة المدى (Long-Range Dependencies): تسمح للنماذج بالتقاط العلاقات بين العناصر البعيدة في ضمن سلسلة الدخل بشكل أكثر فعالية من النماذج التي تعتمد فقط على الهياكل المتكررة (RNN). وهذا أمر بالغ الأهمية للمهام حيث يكون السياق من الأجزاء السابقة من التسلسل مهمًا لفهم الأجزاء اللاحقة [2].

4.2- المحولات (Transformers)

هي بنية شبكة عصبونية تُستخدم في المهام التي تتضمن معالجة بيانات متسلسلة، مثل معالجة اللغة الطبيعية وفصل الكلام. تَستخدم آليات الانتباه بشكل أساسي، حيث تسمح لها بمعالجة بيانات الإدخال بالتوازي بدلاً من التتابع، على عكس الشبكات العصبية المتكررة التقليدية (RNNs) [2].

5.2- نماذج اللغة الكبيرة (large language models)

هي نماذج لغوية إحصائية تستفيد من تقنيات التعلم العميق وخاصةً هياكل المحولات لفهم اللغة البشريّة وتوليدها. تتميز هذه النماذج بحجمها الكبير، وغالبًا ما تحتوي على عشرات إلى مئات المليارات من المعاملات (parameters)، إذ يتم تدريبها على كميّات هائلة من بيانات النصوص من مصادر متنوعة مثل الكتب ومواقع الويب وبيانات المحادثة [3].

كمثال على هذه النماذج سلسلة GPT المقدمة من شركة OpenAI، PALM المقدمة من Google و LLAMA من شركة Meta.

6.2- هندسة الأوامر Prompt engineering

الأمر أو المحفّز (Prompt) هو نص يتم تقديمه لنموذج لغوي لمساعدته على توليد استجابة.

هندسة الأوامر (Prompt engineering) هي عملية صياغة وتحسين التوجيهات للتواصل بشكل فعال مع نماذج اللغات الكبيرة (LLMs). هذه العملية مهمة للحصول على ردود دقيقة وذات صلة من النموذج. مع تطور نماذج اللغة، أصبحت مهارة هندسة الأوامر أساسية للمستخدمين الذين يريدون الاستفادة القصوى من نماذج اللغات الكبيرة وتحقيق أفضل النتائج في مختلف المجالات [4].

عند تصميم هذه التوجيهات يجب مراعاة المعايير التالية:

- الوضوح: يجب أن تكون التوجيهات واضحة وسهلة الفهم، حيث يساعد ذلك على توليد إستجابة أكثر دقة من قبل النماذج اللغوية (LLMs) [4].
- إضافة قيود صريحة: يجب إضافة إرشادات وقيود محددة عند الطلب، حيث يساعد ذلك في تضييق نطاق تركيز النموذج اللغوي مما يؤدي إلى استجابة ذات صلة بالطلب [4].
- التجريب: يجب تجريب أنواع مختلفة من التوجيهات لمعرفة ما هو الأفضل، حيث أن تجربة تنسيقات مختلفة يمكن أن يساعد في اكتشاف طرق فعالة للتفاعل النموذج اللغوي [4].
- تحسين التوجيهات باستمرار: يجب الاستمرار في تحسين التوجيهات بناءً على النتائج التي يعيدها النموذج اللغوي، حيث أن هذه العملية التكرارية يمكن أن تعمل على تحسين التوجيه بشكل كبير بمرور الوقت [4].
- التحكم بمعاملات النموذج: يمكن أن يؤدي تغيير معامل مثل درجة الحرارة الخاصة بالنموذج (temperature) التي تحدد مدى إبداع النموذج إلى نتائج مختلفة، كما أن استخدام سلسلة من التوجيهات واحدة تلو الأخرى يؤدي إلى إنشاء تفاعلات أكثر تعقيداً مع النموذج [4].
- تقديم معلومات إضافية للتوجيه: إن إضافة سياق المهمة المطلوبة إلى التوجيهات يمكن أن تساعد في إنتاج استجابات أكثر دقة وملاءمة. يكون هذا مفيدًا بشكل خاص عند التعامل مع مفاهيم مجردة أو مجالات متخصصة [4].
- تقديم أمثلة: يمكن أن يساعد إضافة مجموعة من الأمثلة للخرج المتوقع (Few-shot learning) ضمن التوجيه على الوصول لخرج أكثر دقة، حيث أن التعلم من خلال عدد قليل من الأمثلة يجعل النماذج قابلة للتكيف، خاصةً في السيناريوهات ذات البيانات المحدودة، كما أن هذا الأسلوب يقلل من الإفراط في التجهيز (overfitting) ويعزز المرونة والتخصيص والتكيف السريع مع المهام الجديدة [5].

7.2- خوارزمية Split Miner

سجل الأحداث (Event log) هو ملف منظَّم يُسجِّل تسلسل الأحداث (الأنشطة) المتعلقة بعملية (business process) معيّنة.

Split Miner هي خوارزمية آلية تعمل على إنشاء نماذج BPMN دقيقة وبسيطة إنطلاقاً من سجلات الأحداث. حيث تعالج المشاكل الشائعة في بعض طرق اكتشاف العمليات، مثل إنتاج نماذج معقّدة للغاية أو نماذج لا تتناسب مع سجل الأحداث، حيث تحقق التوازن بين بساطة النموذج وملاءمته ودقته مع الحفاظ على سرعة تنفيذ عالية [6].

8.2- DevOps

هي منهجية في تطوير البرمجيات تدمج بين فِرَق التطوير (development team) والتشغيل (operation team) لتعزيز التعاون وزيادة كفاءة العمل طوال دورة حياة تطوير البرمجيات. حيث أن تبني DevOps يمكن أن يحسن بشكل كبير من أمان وجودة البرمجيات وسرعة تسليمها [7].

9.2- Continuous Integration and Continuous Delivery (CI / CD)

يعد التكامل المستمر (CI) والنشر المستمر (CD) من الممارسات الأساسية في تطوير البرمجيات الحديثة، وخاصة ضمن أطر عمل DevOps. يعمل CI/CD على أتمتة عمليات اختبار التطبيقات ونشرها، مما يمكن الفرق من تقديم التحديثات بسرعة وكفاءة بعد أي تغيير يحدث ضمن الرماز البرمجي (source code) [7].

10.2- الخدمات المصغّرة (Microservices)

تمثل الخدمات المصغَّرة أسلوبًا معماريًا حديثًا يحلل التطبيقات التقليدية ذات الكتلة الواحدة (monolithic applications) إلى مجموعة من الخدمات الصغيرة الموزعة القابلة للنشر بشكل مستقل عن بعضها البعض. يعزز هذا النهج المرونة في بناء التطبيقات وقابلية التوسع ويزيد من إنتاجيَّة المطورين من خلال السماح للفِرق بالعمل بشكل مستقل على خدمات مختلفة بدعم من ممارسات DevOps وCI / CD [8].

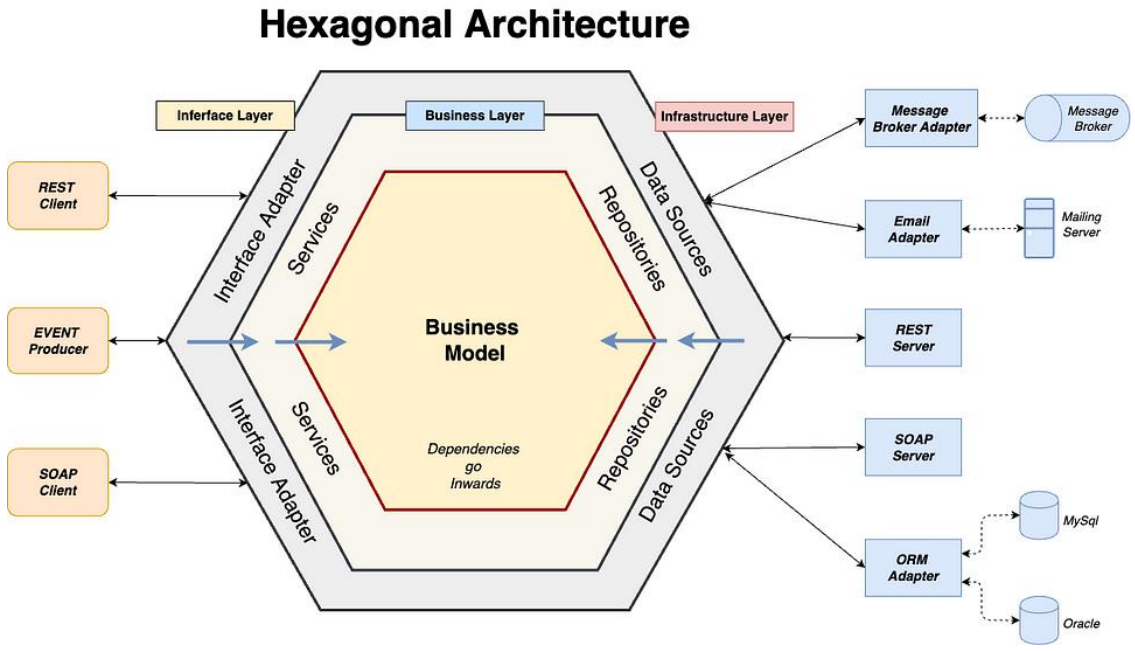
تتميَّز الخدمات المصغرة بتماسكها الداخلي العالي (high cohesion) وارتباطها الفضفاض (loose coupling)، مما يسهل تطوير التطبيقات المعقَّدة من خلال واجهات برمجة تطبيقات (APIs) بسيطة ومحددة جيدًا [8].

11.2- الهندسة المعمارية السداسية (Hexagonal Architecture)

الهندسة المعمارية السداسية، والمعروفة أيضًا باسم نمط المنافذ والمحولات (ports and adapters pattern)، هي نمط تصميمي يعزز مبدأ فصل الاهتمامات (separation of concerns) عند بناء الأنظمة البرمجية عن طريق عزل منطق العمل الأساسي للنظام عن آليات التعامل مع الأنظمة الخارجية، مثل قواعد البيانات أو واجهات المستخدم أو الخدمات الأخرى، مما يضمن بقاء النظام مرنًا وقابلًا للصيانة وقابلًا للتكيف مع التغيير [9].

1.11.2- المفاهيم الرئيسية ضمن الهندسة المعمارية السداسية

- منطق المجال الأساسي (Core Domain Logic): يحتوي المركز الأساسي من المعمارية، والذي يُشار إليه غالبًا باسم النواة (core) أو المجال (domain) على منطق العمل الأساسي للنظام (business logic). حيث يكون مركز البنية مستقلاً عن أي أنظمة خارجية، مما يضمن أن المنطق الأساسي يمكن أن يتطور دون أن يكون مرتبطًا ارتباطًا وثيقًا بالخدمات الخارجية [9].
- المنافذ (Ports): تُحدد المنافذ الواجهات البرمجية التي يتفاعل من خلالها المنطق الأساسي مع الأنظمة الخارجية. وهي تجريدات (abstractions) تسمح بتوصيل محولات (adapters) مختلفة أو استبدالها دون التأثير على المنطق الأساسي [9].
- المحولات (Adapters): المحولات هي تنفيذات (Implementations) للمنافذ. تعمل هذه العناصر كوسطاء بين المنطق الأساسي والأنظمة الخارجية، مثل قواعد البيانات أو واجهات المستخدم أو الخدمات الأخرى [9].
- يكون إتجاه التبعيات ضمن النظام من الخارج إلى الداخل، حيث أن التفاصيل التنجيزية (implementations) تعتمد على التجريدات (abstractions)، مما يسمح بتغيير هذه التنجيزات المتمثلة بالمحولات دون التأثير على نواة النظام.



2.11.2- محاسن ومساوئ هذه البنية

تقدم هذه الهندسة المعمارية العديد من الفوائد، بما في ذلك تحسين الصيانة من خلال الفصل الواضح بين المكونات، والمرونة التي تسمح باستبدال المحولات دون التأثير على المنطق الأساسي، فضلاً عن تحسين إمكانية اختبار المكونات الأساسية بشكل مستقل. ومع ذلك، فإنها تتضمن أيضًا بعض العيوب، مثل زيادة التعقيد نتيجة الطبقات الإضافية، ما قد يكون غير ضروري في المشاريع الصغيرة، فضلاً عن منحني التعلم الأكثر حدَّةً للمطورين الذين يحتاجون إلى فهم وتطبيق مبادئ المنافذ والمحولات [9].

Partially Ordered Workflow Language (POWL) -12.2

هو تمثيل بياني مرتب جزئياً يستخدم لنمذجة عمليات الأعمال، يحتوي هذا التمثيل على عناصر تساعد في نمذجة التدفق بين عناصر المخطط والقرارات ضمن العملية والحلقات [15].

يتميز هذا التمثيل بمجموعة من الخصائص أهمها:

- البساطة: تمكن الطبيعة الهرمية لـ POWL من إنشاء نموذج مبسط من خلال إنشاء نماذج متكررة ودمجها في نماذج أكبر [15].

- القوة التعبيرية: يدعم مجموعة واسعة من هياكل العمليات، حيث يسمح بنمذجة التبعيات المعقدة غير الهرمية مع الحفاظ على ضمانات الجودة للغات نمذجة العمليات الهرمية [15].

الفصل الثالث

الدراسة المرجعية

يَعرِض هذا الفصل الأبحاث المرتبطة بالعمل المقدم.

1.3- مقدمة

إن مجال توليد مخططات BPMN باستخدام نماذج اللغات الكبيرة (LLM) هو مجال جديد ويحوي على العديد من الأبحاث، سنستعرض في الفقرات التالية مجموعة من الأفكار المأخوذة من الأبحاث ضمن هذا المجال.

2.3- استخدام نماذج اللغات الكبيرة في توليد مخططات BPMN

يمكن لنماذج اللغة الكبيرة (LLMs) تنفيذ مهام BPMN بشكل فعال من خلال استغلال قدراتها المتقدمة في معالجة اللغة الطبيعية. حيث تقوم هذه النماذج بتحليل وتحويل توصيف العمليات النصية غير المنظمة إلى نماذج BPMN منظمة، حيث يمكن لنماذج اللغة الكبيرة (LLMs) تحديد المكونات الرئيسية لعمليات الأعمال مثل المهام، الأحداث ونقاط القرار من النصوص والتعرف على تسلسل الأنشطة والعلاقات بينها، مما يسمح لها ببناء رسوم بيانية لـ BPMN [10].

PET Dataset -3.3

تمثل مجموعة بيانات PET أحد الأصول المحورية التي تهدف إلى تعزيز البحث العلمي في استخراج عمليات الأعمال من المصادر النصية للغة الطبيعية. تشتمل مجموعة بيانات PET على مجموعة من 45 وثيقة تتميز بالتوضيحات السردية لعمليات الأعمال المتنوعة. يتم شرح كل وثيقة بدقة للتأكيد على المكونات الهامّة مثل الأنشطة والبوابات والجهات الفاعلة ومعلومات التدفق، إذ أنه لا غنى عن هذه العناصر لفهم العمليات الموضّحة في النص. تم إنشاء مجموعة البيانات من خلال إجراء التعليق التوضيحي المنهجي الذي شمل ثلاثة معلقين مؤهلين. قام هؤلاء الخبراء بتحديد وتصنيف عناصر العملية جنبًا إلى جنب مع علاقاتها المتبادلة، مع الالتزام بمخطط التعليقات التوضيحية المحدد مسبقًا. يضمن هذا المخطط التوحيد والدقة في التعليقات التوضيحية، مما يجعل مجموعة البيانات موردًا يمكن الاعتماد عليه للباحثين الأكاديميين [11].

4.3- توليد مخطط BPMN مباشرة من توصيف العملية النصي

بالاستفادة من نماذج اللغة الكبيرة مثلاً (GPT4)، يتم إرسال طلب وحيد (single prompt) للنموذج يحتوي على تعليمات من أجل تحويل وصف العملية المعطى مباشرة إلى ترميز JSON متوافق مع المعيار BPMN 2.0. هذا النهج فعال من حيث جهد التنفيذ ووقت التشغيل ولكن عند تجربة هذا الأسلوب لم تكن النتائج مرضية حيث أن الترميز المولد لم يكن بالتنسيق المحدد حيث أن التنسيق المطلوب معقد للغاية لأنه يحتوي على العديد من التفاصيل حول الرسم البياني، مثل معرفات محددة (id) وإحداثيات صريحة لكل عنصر رسومي. لذلك، يواجه النموذج اللغوي الكبير صعوبة في توليد الترميز بالشكل الصحيح [12].

هذا النموذج موجود فعلاً **BPMN-GPT**.

5.3- بناء مخطط BPMN عن طريق توليد تمثيل وسيط للعملية

ضمن هذا النهج يتم إرسال طلب لنموذج اللغة الكبير يحتوي على توصيف العملية ومجموعة من التعليمات التي تساعد النموذج على توليد التمثيل الوسيط الذي سيتم معالجته من أجل الوصول للمخطط النهائي، تتكون التعليمات من توصيف نصي للتمثيل الوسيط ومجموعة من الخطوات اللازم اتباعها لتوليد هذا التمثيل.

نوضح في الفقرات التالية بعض التمثيلات الوسيطة المقترحة ضمن الأبحاث.

1.5.3- تمثيل عناصر المخطط باستخدام تدوين معرف مسبقاً

يتم تمثيل عناصر BPMN ضمن النموذج المتولد الوسيط على الشكل التالي:

- تمثل المهام (Tasks) ضمن النموذج بكلمات من اللغة الطبيعية.
- يتم تمثيل التدفقات بين عناصر النموذج (Flows) كأسهم (<-).
- البوابات الحصرية تمثل بكلمة XOR والمتوازية تمثل بكلمة AND.
- يتم تمثيل الشروط الخارجة من البوابات الحصرية كنص بين قوسين (شرط) تُستخدم لتمثيل معايير القرار.
- يتم توفير تعيين الفاعل إلى المهمة بالتنسيق التالي اسم الفاعل: [مهمة أولى، مهمة ثانية ، ...].

لا يتم تضمين عناصر أخرى ضمن توصيف النموذج المتولد (على سبيل المثال، الرسائل)، ولا يتم تقديم أزواج من النصوص والنماذج الكاملة أو الجزئية المقابلة ضمن الطلب لتجنب التحيز تجاه أسلوب نمذجة معين [13].

أظهر الأسلوب السابق نتائج جيدة على عدد من النصوص التي تحوي توصيف لعمليات، حيث أن النتيجة تمثل توصيف العملية بشكل دقيق خصوصاً في حال التوصيفات النصية البسيطة، ولكن في بعض الحالات كان من الممكن تبسيط نموذج BPMN الناتج [13].

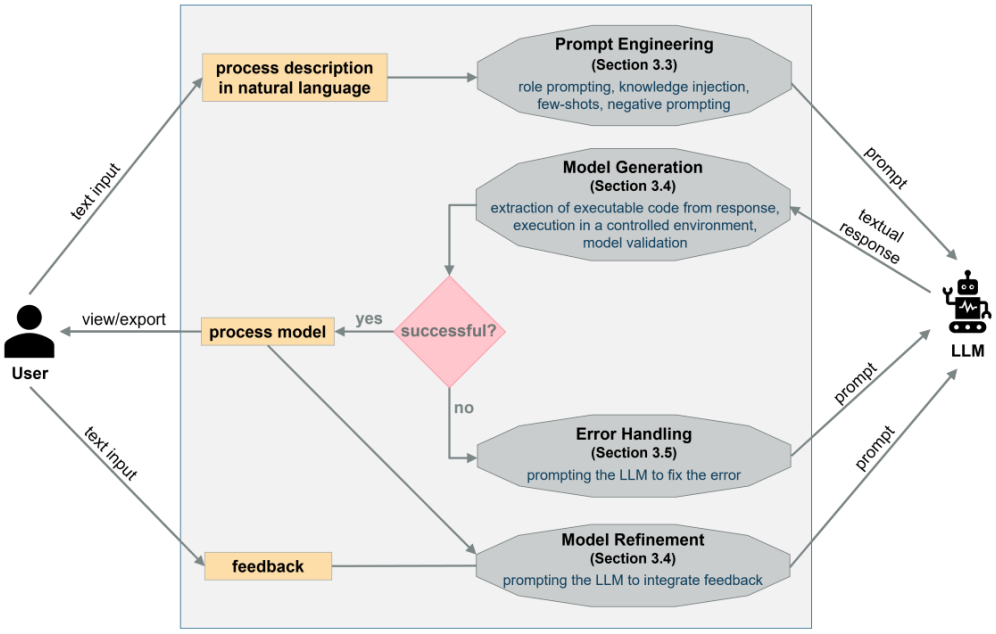
2.5.3- تمثيل العملية من خلال مجموعة من الآثار

تتمثل فكرة هذا النهج في توليد مجموعة فريدة من الآثار (تعقبات) من توصيف العملية واستخدام خوارزمية اكتشاف العمليات (Split Miner) لاستخراج نموذج العملية. في الخطوة الأولى، يتم تحفيز GPT-4 (prompt) لاستخراج وإنتاج مجموعة فريدة من الآثار بناءً على وصف العملية. تكون النتيجة قائمة من الآثار، والتي تُستخدم بعد ذلك كدفتر أحداث اصطناعي وتُدخل في خوارزمية Split Miner التي تستخرج نموذج العملية وتعيده كترميز JSON. لذلك، يتطلب النهج جهدًا بسيطًا ودائمًا ما يعيد JSON لرسم BPMN 2.0 [12].

عند اختبار هذا النهج، يقوم GPT-4 باستخراج مجموعة متماسكة وفريدة من الآثار؛ وتقوم خوارزمية Split Miner بإنتاج رسم بياني متوافق مع BPMN. بالنسبة لوصف العمليات الذي يحتوي على بنية ونحو وتعقيد مشابه للتوصيف النصي المستخدم ضمن البحث [12]، يحتوي الرسم البياني الناتج دائمًا على جميع الأنشطة المشار إليها وتدفق التحكم الصحيح، ممثلة بالأنشطة والبوابات الحصرية وعقدة البداية والنهاية، بالنسبة لوصف العمليات الأكثر غموضًا وتعقيدًا، يواجه التنفيذ الحالي لهذه المنهجية صعوبة في استخراج تدفقات التحكم المعقدة بشكل مثالي. جزئيًا، يفتقر إلى افتراض المعرفة الضمنية. ومع ذلك، لا يعني ذلك أنه لا يمكنه التعامل مع أوصاف العمليات الأكثر صعوبة، ولكن يمكن تحسين هذه المنهجية، على سبيل المثال، عن طريق تحسين الأوامر (Prompts) وتغطية بعض الحالات الخاصة [12].

6.3- توليد كود برمجي مساعد لبناء مخطط BPMN

ضمن هذه المنهجية يتم استلام التوصيف النصي الخاص بالعملية من قبل المستخدم، حيث يضاف إليه مجموعة من التعليمات تقوم بتوجيه نموذج اللغة الكبير من أجل توليد كود برمجي قادر على توليد مخطط من نوع POWL، حيث يكون هذا المخطط ترميزاً مرحلياً يمكن تحويله لمخطط BPMN. يتم أخذ الكود البرمجي وتنفيذه لبناء مخطط POWL، في حال حدوث خطأ خلال التنفيذ يتم إرسال الخطأ الناتج لنموذج اللغة ليقوم بتصحيحه. يمكن للمستخدم إرسال ملاحظات عن المخطط بعد توليده، إذ يتم إرسال هذه الملاحظات لنموذج اللغة ويتم إعادة توليد المخطط مرة أخرى.



7.3- منصات BPMN المتاحة

يوجد مجموعة من المنصات التي تساعد في التعامل مع BPMN، نستعرض في الفقرات التالية مجموعة منها.

Camunda -1.7.3

هي عبارة عن منصة مفتوحة المصدر تدعم معايير BPMN لنمذجة وتنفيذ ومراقبة عمليات الأعمال. وهي توفر أدوات مثل Web Modeler و Desktop Modeler لتصميم العمليات. تضمن Camunda إمكانية التوسع والمرونة والتكامل مع أنظمة مختلفة، مما يجعلها مثالية لتنسيق العمليات والتطبيقات عبر الخدمات. يضمن التزامها بمعايير BPMN نماذج عملية واضحة وقابلة للنقل [16].

bpmn.io -2.7.3

هي عبارة عن مجموعة أدوات مفتوحة المصدر مصممة للعمل مع مخططات BPMN 2.0، وتهدف في المقام الأول إلى تسهيل إنشاء نماذج عمليات الأعمال ومشاركتها ودمجها [17]. من أهم ميزاتها:

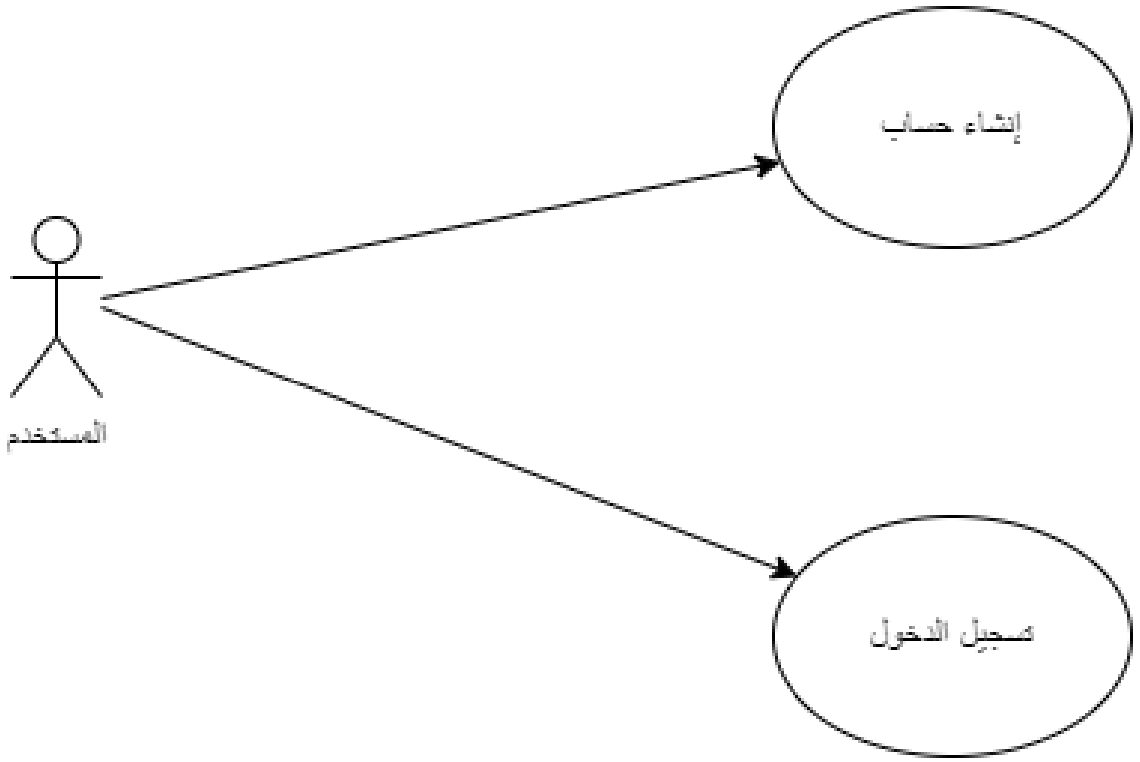
- النمذجة المستندة إلى الويب: تقدم مكون نمذجة مجانيًا قائمًا على الويب لإنشاء وتعديل مخططات BPMN 2.0، مع التركيز على قابلية الاستخدام الجيدة.
- مجموعة أدوات المطورين: بالنسبة للمطورين، توفر bpmn.io مجموعة أدوات خفيفة الوزن لتضمين مخططات BPMN 2.0 في تطبيقات الويب، مما يسمح بالرسم والتفاعل وشرح البيانات.
- إمكانية التوسع: تتضمن مجموعة الأدوات واجهة برمجة تطبيقات (APIs) قابلة للتوسع، مما يتيح تخصيص المكتبة وتحسينها.
- التوافر: مجموعة الأدوات مفتوحة المصدر ومجانية الاستخدام، ومتوفرة على GitHub، ويمكن تضمينها في تطبيقات الويب.

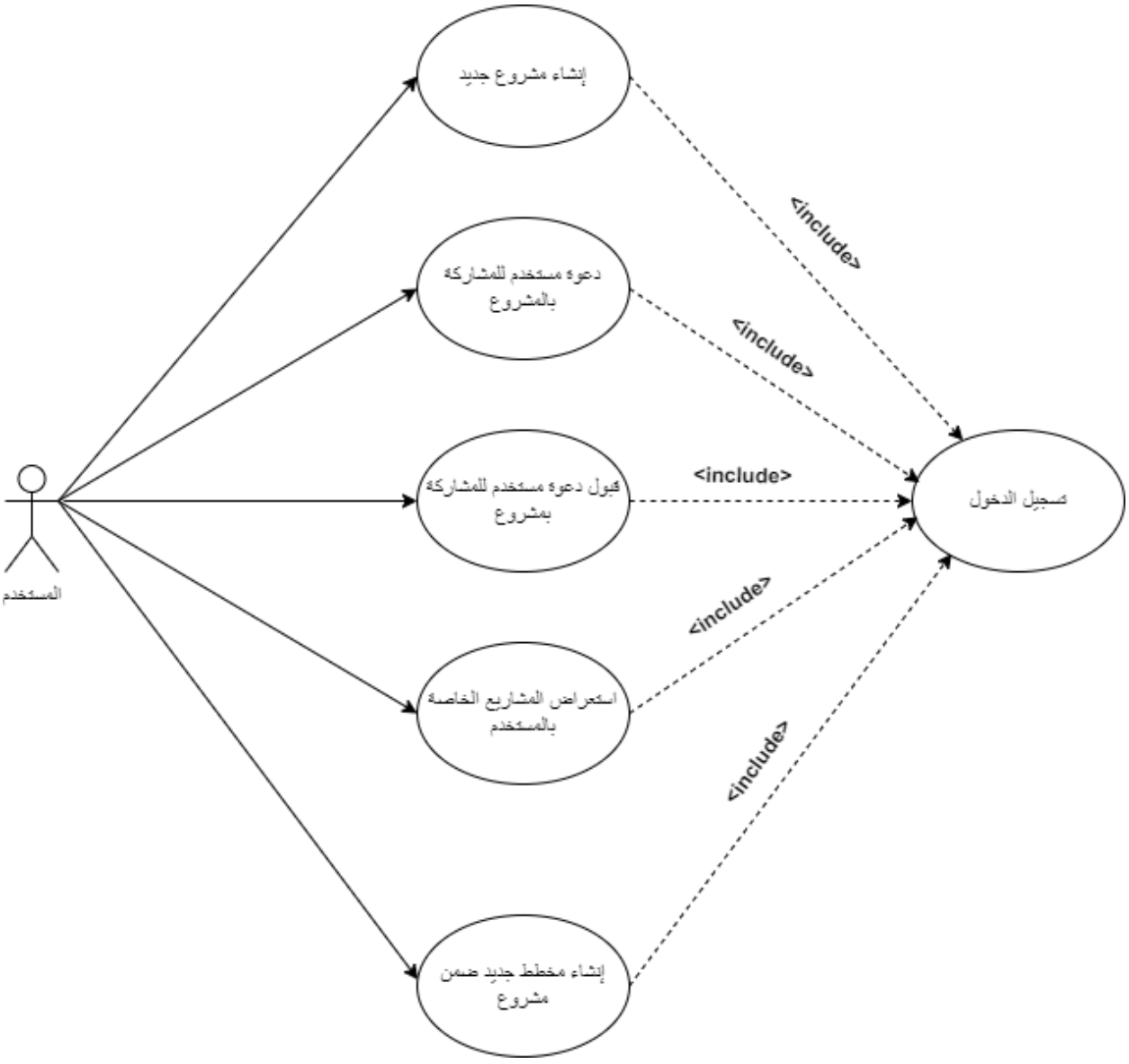
الفصل الرابع

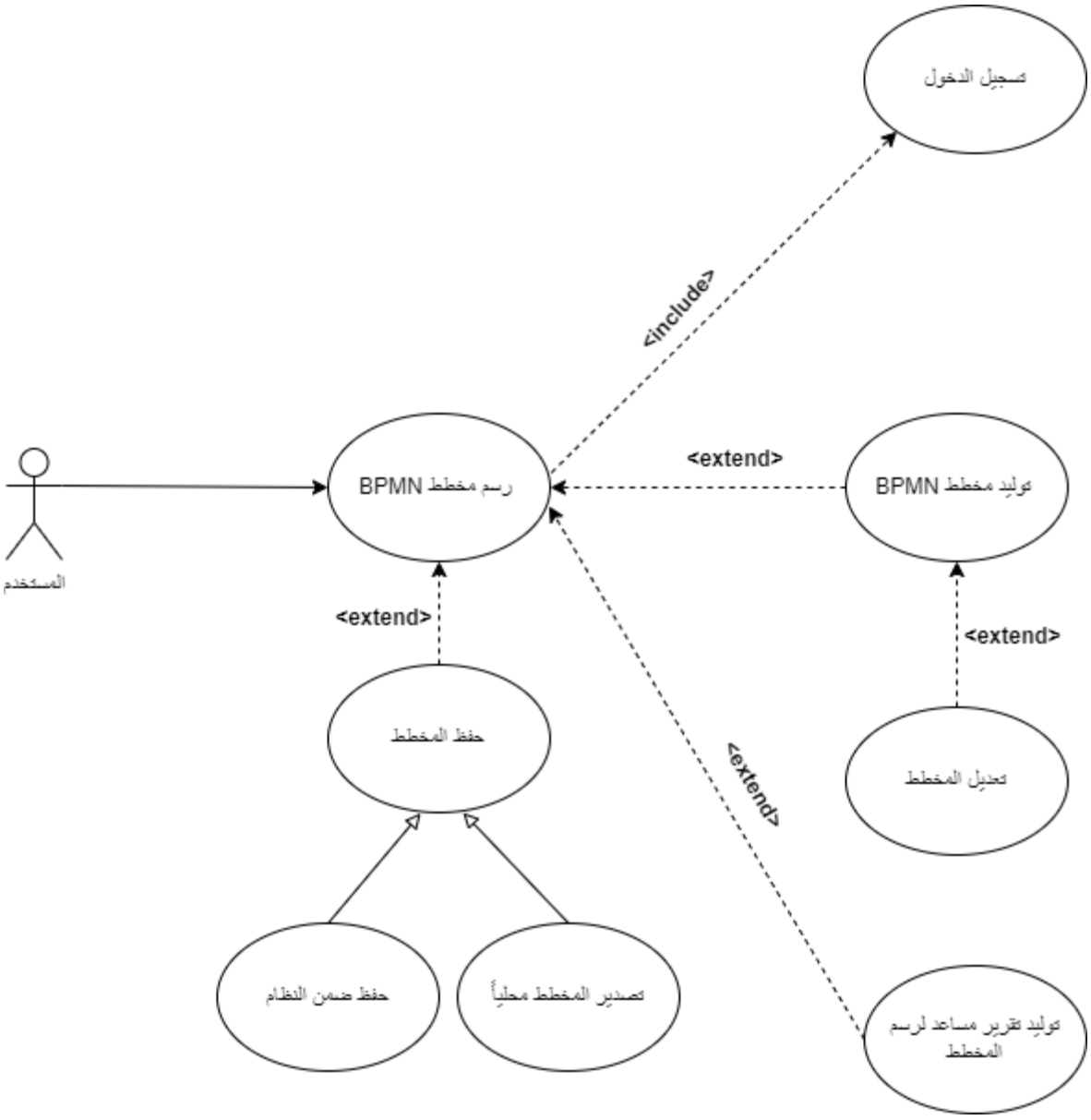
الدراسة التحليلية

يوضح هذا الفصل عملية تحليل النظام ودراسة متطلباته وصولاً للتصميم.

1.4- مخططات حالات الاستخدام




إنشاء مشروع جديد
دعوة مستخدم للمشاركة بالمشروع
قبول دعوة مستخدم للمشاركة بمشروع
استعراض المشاريع الخاصة بالمستخدم
إنشاء مخطط جديد ضمن مشروع
تسجيل الدخول
المستخدم
<include>
<include>
<include>
<include>
<include>


تسجيل الدخول
<include>
المستخدم
رسم مخطط BPMN
<extend>
توليد مخطط BPMN
<extend>
<extend>
حفظ المخطط
تعديل المخطط
<extend>
حفظ ضمن النظام
تصدير المخطط محلياً
توليد تقرير مساعد لرسم المخطط

2.4- السرد النصي لحالات الاستخدام

1.2.4- إنشاء حساب جديد

جدول 1 حالة استخدام إنشاء حساب جديد

اسم الحالة: إنشاء حساب جديد	
الوصف Description	يقوم المستخدم بإنشاء حساب جديد خاص به.
الفاعلين Actors	المستخدم
الشروط السابقة Precondition	لا يوجد.
الشروط اللاحقة Postcondition	تم إنشاء حساب جديد خاص بالمستخدم ضمن النظام.

سير الأحداث

السيناريو الأساسي الناجح

جدول 2 السيناريو الناجح لحالة استخدام إنشاء حساب جديد

النظام	المستخدم
	1. يطلب إنشاء حساب جديد.
2. يطلب النظام المعلومات التالية: **– البريد الالكتروني الخاص بالمستخدم.** **– كلمة المرور.** **– تأكيد كلمة المرور.**	
	3. يحدد المستخدم المعلومات المطلوبة ويطلب تأكيد العمليّة.

4. يتحقق النظام من صحة المعلومات ويعيد للمستخدم رسالة توضح إنتهاء العملية.	

المسارات البديلة

لا يوجد.

مسارات الأخطاء

E1: في المرحلة رقم 2 في حال تحقَّقت إحدى الحالات التالية:

- البريد الالكتروني غير صالح.
- كلمة المرور ضعيفة.
- كلمة المرور لا تتطابق مع حقل تأكيد كلمة المرور.

في حال تحققت إحدى الحالات السابقة، يتم استبدال الخطوة الثانية في السيناريو الأساسي بالخطوة التالية:

2. يعيد النظام رسالة توضّح سبب الخطأ ويَطلب تحديد المعلومات مِن جديد.

E2: في المرحلة رقم 4 في حال تحقَّقت إحدى الحالات التالية:

- البريد الالكتروني مسجل مسبقاً ضمن النظام.

في حال تحققت إحدى الحالات السابقة، يتم استبدال الخطوة الرابعة في السيناريو الأساسي بالخطوة التالية:

4. يعيد النظام رسالة توضّح سبب الخطأ ويَطلب تحديد المعلومات مِن جديد.

2.2.4- إنشاء مشروع جديد

جدول 2 حالة استخدام إنشاء مشروع جديد

اسم الحالة: إنشاء مشروع جديد	
الوصف Description	يقوم المستخدم بإنشاء مشروع جديد خاص به.
الفاعلين Actors	المستخدم
الشروط السابقة Precondition	المستخدم مسجل في الموقع مسبقاً.
الشروط اللاحقة Postcondition	تم إنشاء مشروع جديد خاص بالمستخدم ضمن النظام.

سير الأحداث

السيناريو الأساسي الناجح

جدول 2 السيناريو الناجح لحالة استخدام إنشاء مشروع جديد

النظام	المستخدم
	1. يطلب إنشاء مشروع جديد.
2. يطلب النظام المعلومات التالية: **– اسم المشروع** **– وصف عن المشروع** **– نوع المشروع (عام، خاص)**	
	3. يحدد المستخدم المعلومات المطلوبة ويطلب تأكيد العمليّة.

4. يتحقق النظام من صحة المعلومات ويعيد للمستخدم رسالة توضح إنتهاء العملية.	

المسارات البديلة

لا يوجد.

مسارات الأخطاء

E1: في المرحلة رقم 2 في حال تحقَّقت إحدى الحالات التالية:

- اسم المشروع فارغ.

-توصيف المشروع فارغ.

-لم يتم تحديد نوع المشروع.

في حال تحققت إحدى الحالات السابقة، يتم استبدال الخطوة الثانية في السيناريو الأساسي بالخطوة التالية:

2. يعيد النظام رسالة توضّح سبب الخطأ ويَطلب تحديد المعلومات مِن جديد.

3.2.4- دعوة مستخدم للمشاركة ضمن مشروع

جدول 3 حالة استخدام دعوة مستخدم للمشاركة ضمن مشروع

اسم الحالة: دعوة مستخدم للمشاركة ضمن مشروع	
الوصف Description	يقوم المستخدم بدعوة مستخدم إخر للمشاركة ضمن مشروع.
الفاعلين Actors	المستخدم
الشروط السابقة Precondition	المستخدم مسجل في الموقع مسبقاً، وقد أنشأ مشروعاً.
الشروط اللاحقة Postcondition	تم إرسال دعوة للمستخدم الآخر للمشاركة ضمن المشروع.

سير الأحداث

السيناريو الأساسي الناجح

جدول 2 السيناريو الناجح لحالة استخدام دعوة مستخدم للمشاركة ضمن مشروع

النظام	المستخدم
	1. يطلب إنشاء دعوة.
2. يطلب النظام المعلومات التالية: **– جزء من البريد الالكتروني للمستخدم الثاني على الأقل ثلاث محارف.**	
	3. يحدد المستخدم المعلومات المطلوبة.
4. يقوم النظام بإظهار مجموعة من عناوين البريد الموجودة ضمن النظام التي تحوي الجزء النصي الذي حدده المستخدم.	
	5. يختار المستخدم البريد المناسب من القائمة ويطلب تأكيد العمليّة.
5. يتحقق النظام من صحة المعلومات ويعيد للمستخدم رسالة توضح إنتهاء العملية.	

المسارات البديلة

لا يوجد.

مسارات الأخطاء

E1: في المرحلة رقم 4 في حال تحقَّقت إحدى الحالات التالية:

– لا يوجد بريد يحتوي على تسلسل الحروف المكتوبة من قبل المستخدم.

في حال تحققت إحدى الحالات السابقة، يتم استبدال الخطوة الرابعة في السيناريو الأساسي بالخطوة التالية:

4. يعيد النظام رسالة توضّح سبب الخطأ ويَطلب تحديد المعلومات مِن جديد.

4.2.4- قبول دعوة للمشاركة ضمن مشروع

جدول 4 حالة استخدام قبول دعوة للمشاركة ضمن مشروع

اسم الحالة: قبول دعوة للمشاركة ضمن مشروع	
الوصف Description	يقوم المستخدم بقبول الدعوة للمشاركة ضمن مشروع معين.
الفاعلين Actors	المستخدم
الشروط السابقة Precondition	المستخدم مسجل في الموقع مسبقاً، وقد قام أحد المستخدمين الآخرين بدعوته.
الشروط اللاحقة Postcondition	تم قبول الدعوة للمشاركة ضمن مشروع.

سير الأحداث

السيناريو الأساسي الناجح

جدول 2 السيناريو الناجح لحالة استخدام قبول دعوة للمشاركة ضمن مشروع

النظام	المستخدم

	1. يطلب استعراض الدعوات.
2. يقوم النظام بإظهار مجموعة الدعوات التي تلقاها المستخدم.	
	3. يحدد المستخدم الدعوة التي يريد قبولها ويطلب تأكيد العمليّة.
4. يعيد النظام للمستخدم رسالة توضح إنتهاء العملية، وأن المستخدم أصبح مشاركاً ضمن المشروع.	

المسارات البديلة

P1: في المرحلة رقم 3 في حال تحقَّقت إحدى الحالات التالية:

– طلب المستخدم رفض الدعوة.

في حال تحققت إحدى الحالات السابقة، يتم استبدال الخطوة الرابعة في السيناريو الأساسي بالخطوة التالية:

4. يعيد النظام رسالة توضّح أنه الدعوة قد تم رفضها.

مسارات الأخطاء

لا يوجد.

5.2.4- رسم مخطط BPMN ضمن مشروع معين

جدول 5 حالة استخدام رسم مخطط BPMN ضمن مشروع معين

اسم الحالة: رسم مخطط BPMN ضمن مشروع معين	
الوصف Description	يقوم المستخدم برسم مخطط BPMN ضمن مشروع معين.
الفاعلين Actors	المستخدم
الشروط السابقة Precondition	المستخدم مسجل في الموقع مسبقاً، وقد قام باختيار المشروع الذي يريد العمل عليه.
الشروط اللاحقة Postcondition	تم عرض واجهة الرسم.

سير الأحداث

السيناريو الأساسي الناجح

جدول 2 السيناريو الناجح لحالة استخدام رسم مخطط BPMN ضمن مشروع معين

النظام	المستخدم
	1. يطلب البدء برسم مخطط.
2. يقوم النظام بإظهار الواجهة الخاصة برسم المخططات.	

المسارات البديلة

لا يوجد.

مسارات الأخطاء

لا يوجد.

6.2.4- توليد مخطط BPMN

جدول 6 حالة استخدام توليد مخطط BPMN

اسم الحالة: توليد مخطط BPMN	
الوصف Description	يقوم النظام بتوليد مخطط BPMN وعرضه ضمن الواجهة.
الفاعلين Actors	المستخدم
الشروط السابقة Precondition	المستخدم مسجل في الموقع مسبقاً، وقد بدأ برسم مخطط.
الشروط اللاحقة Postcondition	تم عرض المخطط ضمن الواجهة.

سير الأحداث

السيناريو الأساسي الناجح

جدول 2 السيناريو الناجح لحالة استخدام توليد مخطط BPMN

النظام	المستخدم

	1. يطلب عملية توليد المخطط.
2. يقوم النظام بطلب توصيف العملية المراد توليد المخطط لها.	
	3. يحدد المستخدم المعلومات المطلوبة ويطلب تأكيد العمليّة.
4. يقوم النظام بعرض المخطط ضمن المخطط.	

المسارات البديلة

لا يوجد.

مسارات الأخطاء

E1: في المرحلة رقم 4 في حال تحقَّقت إحدى الحالات التالية:

- في حال كان التوصيف النصي فارغاً.

في حال تحققت إحدى الحالات السابقة، يتم استبدال الخطوة الرابعة في السيناريو الأساسي بالخطوة التالية:

4. يعيد النظام رسالة توضّح سبب الخطأ ويَطلب تحديد المعلومات مِن جديد.

7.2.4- حفظ المخطط

جدول 7 حالة استخدام توليد مخطط BPMN

اسم الحالة: توليد مخطط BPMN

الوصف Description	يقوم المستخدم بطلب حفظ المخطط، إذ يتم حفظه بأسلوبين إما محلياً عن طريق تحميله أو ضمن النظام.
الفاعلين Actors	المستخدم
الشروط السابقة Precondition	المستخدم مسجل في الموقع مسبقاً، وقد قام برسم مخطط.
الشروط اللاحقة Postcondition	تم حفظ المخطط.

سير الأحداث

السيناريو الأساسي الناجح

جدول 2 السيناريو الناجح لحالة استخدام توليد مخطط BPMN

النظام	المستخدم
	1. يطلب حفظ المخطط ضمن النظام.
2. يقوم النظام بحفظ المخطط ضمن قاعدة المعطيات.	

المسارات البديلة

E1: في المرحلة رقم 1 في حال تحقَّقت إحدى الحالات التالية:

– طلب المستخدم حفظ المخطط محلياً ضمن جهازه.

في حال تحققت إحدى الحالات السابقة، يتم استبدال الخطوة الثانية في السيناريو الأساسي بالخطوة التالية:

4. يقوم النظام بتحميل المخطط إلى جهاز المتسخدم.

مسارات الأخطاء

لا يوجد.

8.2.4- توليد تقرير مساعد لرسم المخطط

جدول 8 حالة استخدام توليد تقرير مساعد لرسم المخطط

اسم الحالة: رسم مخطط BPMN ضمن مشروع معين	
الوصف Description	يقوم النظام بتوليد تقرير مساعد يمكن الاستفادة منه لتوليد مخطط BPMN.
الفاعلين Actors	المستخدم
الشروط السابقة Precondition	المستخدم مسجل في الموقع مسبقاً، وقد بدأ برسم مخطط.
الشروط اللاحقة Postcondition	يتم عرض التقرير.

سير الأحداث

السيناريو الأساسي الناجح

جدول 2 السيناريو الناجح لحالة استخدام رسم مخطط BPMN ضمن مشروع معين

النظام	المستخدم
	1. يطلب إنشاء تقرير انطلاقاً من التوصيف النصي للعملية.

2. يقوم النظام بإنشاء التقرير وعرضه.	

المسارات البديلة

لا يوجد.

مسارات الأخطاء

E1: في المرحلة رقم 2 في حال تحقَّقت إحدى الحالات التالية:

- في حال كان التوصيف النصي فارغاً.

في حال تحققت إحدى الحالات السابقة، يتم استبدال الخطوة الثانية في السيناريو الأساسي بالخطوة التالية:

2. يعيد النظام رسالة توضّح سبب الخطأ ويَطلب تحديد المعلومات مِن جديد.

الفصل الخامس

المنهجية المقترحة

يَعرض هذا الفصل المنهجية المقترحة ضمن المشروع لتوليد مخططات BPMN.

1.5- مقدمة

بعد النظر إلى الأبحاث في مجال توليد مخططات BPMN تم اتخاذ منهجية استخدام هندسة الأوامر لتوليد هذه المخططات انطلاقاً من توصيف العملية النصي وذلك لعدة أسباب أهمها:

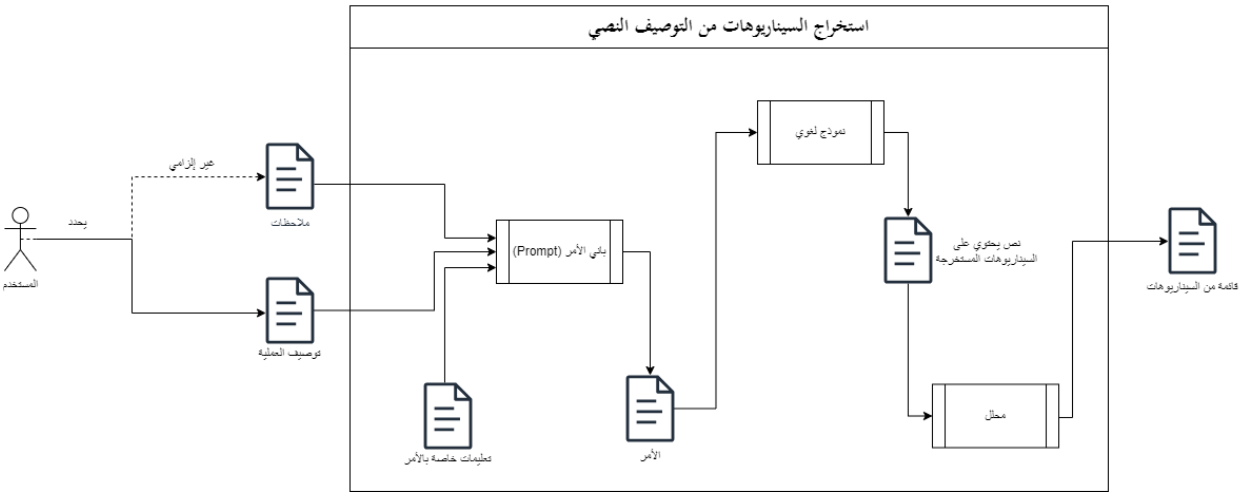
- عدم توفر مجموعة بيانات كبيرة لتدريب نموذج من البداية، حيث تم توضيح أن أكبر مجموعة بيانات (annotated) لهذه المسألة تحتوي على 45 عملية فقط [11].
- بعد أن أظهرت الأبحاث قدرة النماذج اللغوية الكبيرة على حل مسائل استخراج معلومات العمليات من التوصيف النصي مباشرة نجد أن استثمار هذه النماذج مباشرة مفيد لحل هذه المسألة [13].
- لم يتم استخدام منهجية تحسين النموذج اللغوي لملاءمة المسألة (fine-tuning) بسبب عدم توفر البيانات والموارد الحاسوبية الكافية، حيث تم توجيه النموذج لأداء المهام من خلال منهجية التعلم من خلال عدد قليل من الأمثلة (Few-shot learning). (يحتاج إلى بيانات تدريب لتعديل أوزان النموذج اللغوي والموارد المطلوبة هي حسب حجم النموذج وحجم بيانات التدريب المراد استخدامها)

2.5- النموذج المقترح

بدايةً نعرف دخل النموذج، حيث يتكون الدخل من التوصيف النصي للعملية المراد توليد مخطط BPMN خاص بها، ومجموعة من الملاحظات التي يحددها المستخدم (في حال أراد ذلك) لتساعد النموذج في فهم العملية. يتم التوليد على عدة خطوات، سيتم توضيحها في الفقرات التالية.

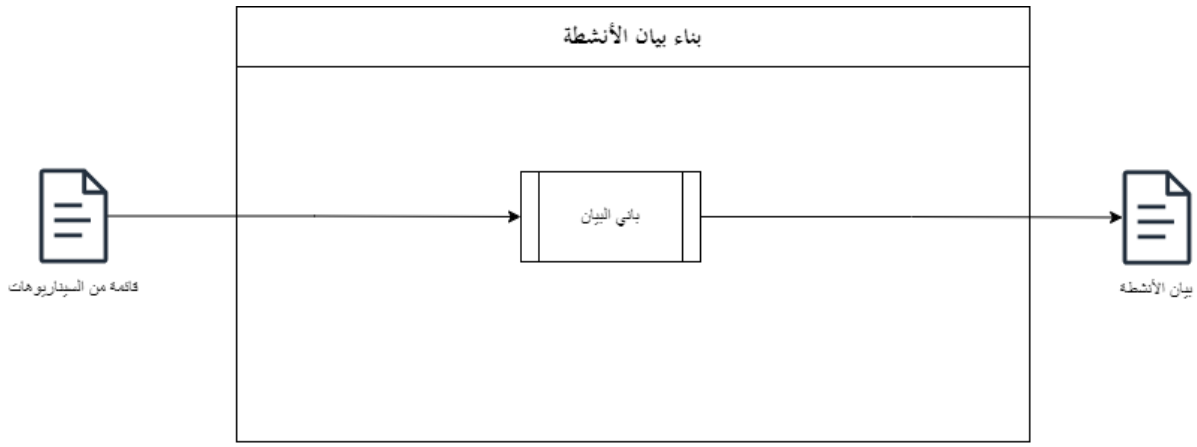
1.2.5- استخراج السيناريوهات من التوصيف النصي

يتم إرسال أمر (Prompt) لنموذج لغة كبير (LLM)، يحتوي هذا الأمر على التوصيف النصي للعملية والملاحظات المقدمة من قبل المستخدم، ومجموعة من التعليمات من أجل توليد كل السيناريوهات الممكنة لتدفقات الأنشطة الموجودة ضمن التوصيف النصي للعملية، يتم استخراج هذه التدفقات من النص المعاد من قبل النموذج اللغوي عن طريق إرسال النص لمحلل (Parser) يقوم بالبحث ضمن النص عن الخرج الموضوع بين علامتي (```) ويقوم ببناء قائمة من السيناريوهات الكاملة (تسلسل من الأنشطة من البداية إلى النهاية) التي تعبر عن خرج هذه الخطوة.



2.2.5- بناء بيان الأنشطة

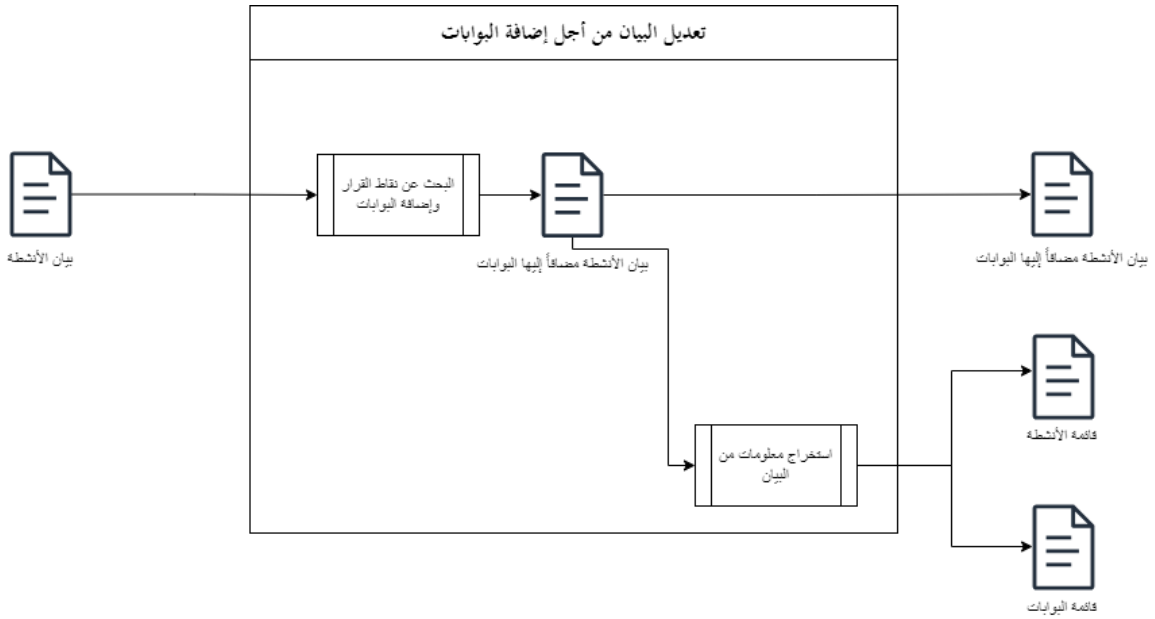
يتم بناء بيان بناءً على التدفقات المستخرجة من المرحلة السابقة، تعبر عقد هذا البيان عن الأنشطة الموصَّفة ضمن العملية وحوافه عن تسلسل التدفق ضمن العملية. من خلال هذا الأسلوب يتم استخراج أهم عناصر مخطط BPMN المتمثلة بالأنشطة والتدفقات فيما بينها، حيث يتم حل مشكلة وجود تدفقات متولدة ليس لها نهاية كما في البحث [13].



3.2.5- معالجة البيان

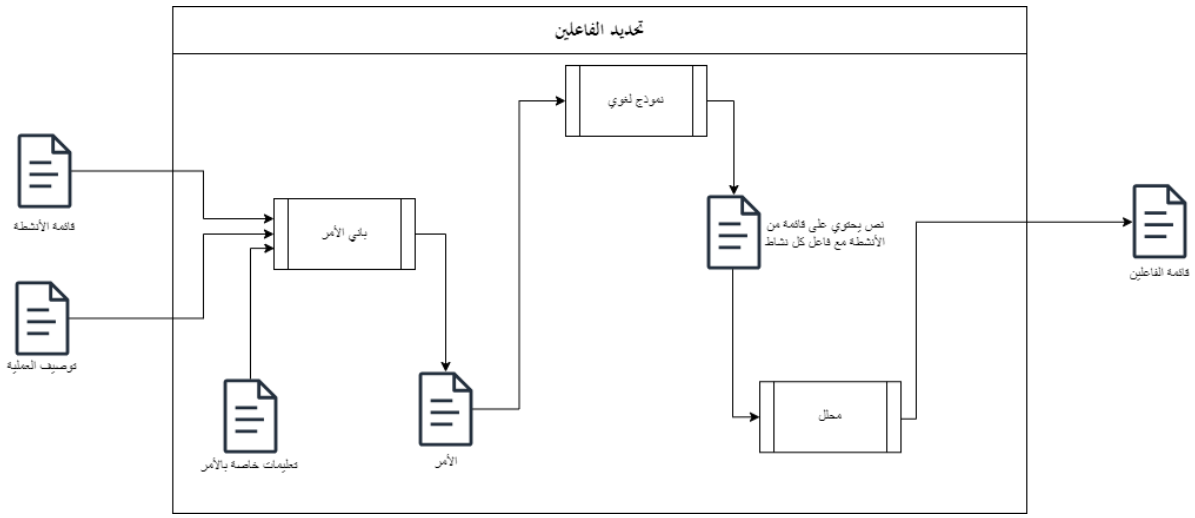
نقوم بتعديل البيان حيث يتم إضافة البوابات (gateways) التي تعبر عن نقاط القرار ضمن العملية المرادة. إذ أنَّ أي نقطة يحدث فيها انتقال من نشاط واحد إلى عدة أنشطة تعتبر نقطة قرار. لذا، نقوم بإضافة بوابة في هذا الموقع ونعطيها رقماً تسلسلياً فريدًا، حيث يصبح تدفق الانتقال من النشاط الأب إلى البوابة ثم إلى النشاطات الأبناء.

يتم استخراج أسماء الأنشطة، وأسماء البوابات مع دخل، وخرج كل بوابة انطلاقاً من البيان المعدل من أجل استخدامها في المراحل اللاحقة.



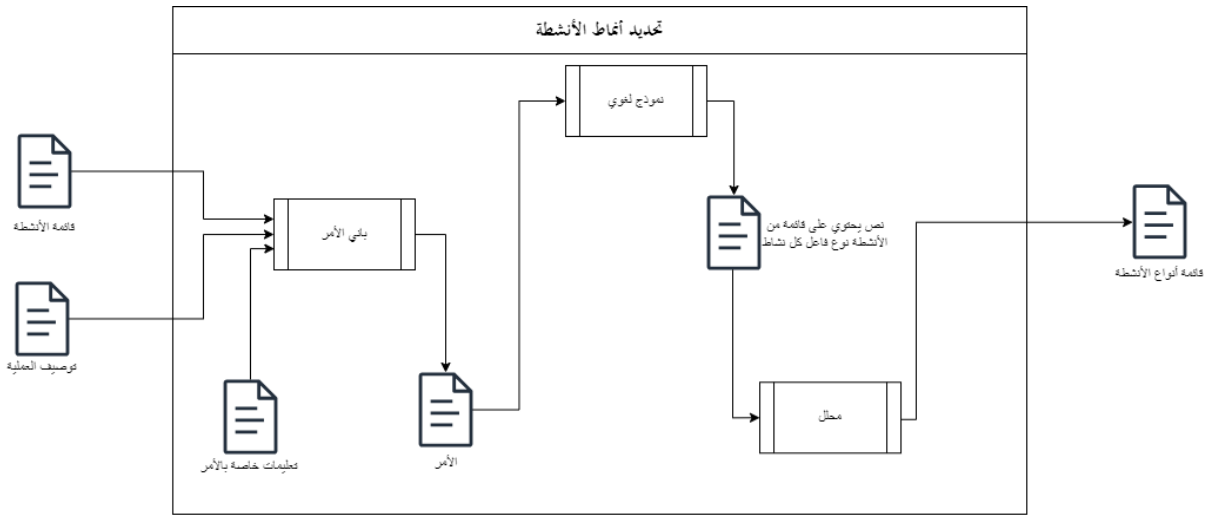
4.2.5- تحديد الفاعلين

نقوم بتحديد الفاعل لكل نشاط من الأنشطة المستخرجة، عن طريق بناء أمر يحتوي على التوصيف النصي للعملية والأنشطة المراد تحديد الفاعل لكل منها ومجموعة من التعليمات والأمثلة من أجل استخراج هؤلاء الفاعلين، ثم يتم إرسال هذا الأمر للنموذج اللغوي. يُعيد النموذج نصاً يحتوي على قائمة تربط كل فاعل مع الأنشطة التي يقوم بها، نقوم بإرسال هذا النص للمحلل الذي يقوم بالبحث ضمن النص عن الخرج الموضوع بين علامتي (```) ويقوم ببناء قائمة الفاعلين.



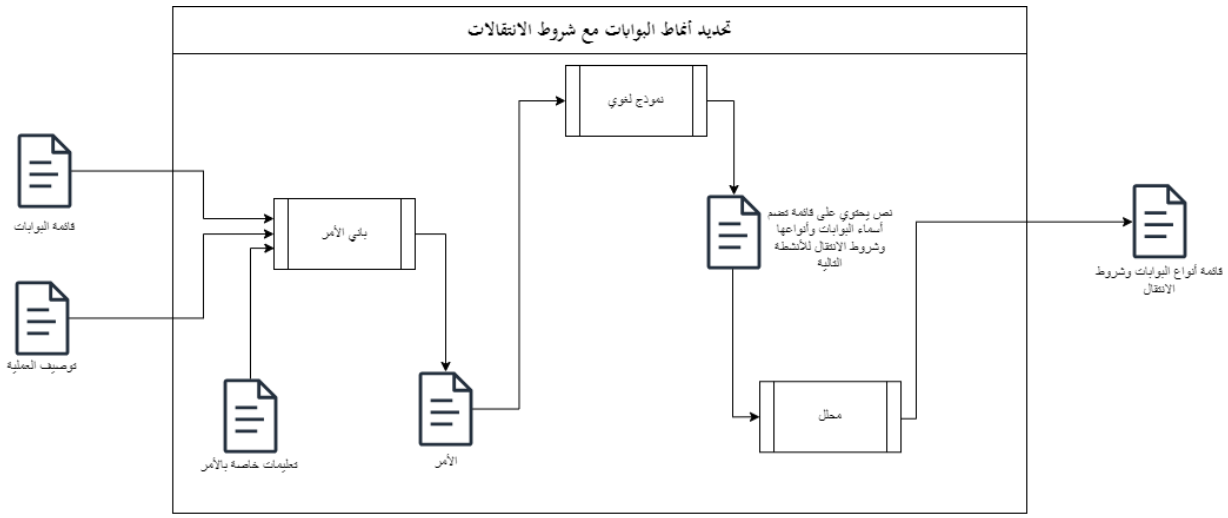
5.2.5- تحديد أنواع الأنشطة

نقوم بتحديد نوع كل نشاط من الأنشطة المستخرجة (.... user task, system task, start event) عن طريق بناء أمر يحتوي على التوصيف النصي للعملية والأنشطة المراد تحديد نوعها، ومجموعة من التعليمات والأمثلة من أجل استخراج هذه الأنواع، ثم يتم إرسال هذا الأمر للنموذج اللغوي. يُعيد النموذج نصاً يحتوي على قائمة تربط كل نشاط مع نوعه، نقوم بإرسال هذا النص للمحلل الذي يقوم بالبحث ضمن النص عن الخرج الموضوع بين علامتي (```) ويقوم ببناء قائمة الأنواع.



6.2.5- تحديد أنواع البوابات مع شروط الانتقالات

يتم تحديد نوع كل بوابة (or, xor, and) مع شروط التدفق بين هذه البوابة والأنشطة المرتبطة بها عن طريق بناء أمر يحتوي على التوصيف النصي للعملية والبوابات المراد تحديد نوعها مع دخل وخرج كل بوابة ومجموعة من التعليمات والأمثلة من أجل استخراج هذه الأنواع، ثم يتم إرسال هذا الأمر للنموذج اللغوي. يُعيد النموذج نصاً يحتوي على قائمة تضم أسماء البوابات وأنواعها وشروط الانتقال للأنشطة التالية، نقوم بإرسال هذا النص للمحلل الذي يقوم بالبحث ضمن النص عن الخرج الموضوع بين علامتي (```) ويقوم ببناء قائمة أنواع البوابات.



ملاحظة: يتم تحديد الفاعلين وأنواع الأنشطة والبوابات على التوازي إذ لا يوجد ترتيب فيما بينها.

7.2.5- بناء المخطط النهائي

انطلاقاً من بيان الأنشطة المعدل نقوم باستخراج التدفقات (Flows).

انطلاقاً من قائمة أنواع الأنشطة نقوم باستخراج المهام (Tasks) والأحداث (Events).

انطلاقاً من قائمة الفاعلين نقوم بوضع كل نشاط ضمن حوض السباحة (Pool) المناسب.

انطلاقاً من قائمة أنواع البوابات نقوم بتحديد نوع كل بوابة ووضع الشروط للانتقال للأنشطة التالية.

يتم إسناد معرف فريد لكل عنصر من عناصر المخطط المستخرجة ويتم إرسالها للواجهة (تكون الوجهة هي المسؤولة عن اسناد موقع لكل عنصر ضمن المخطط) ثم يتم عرض المخطط.

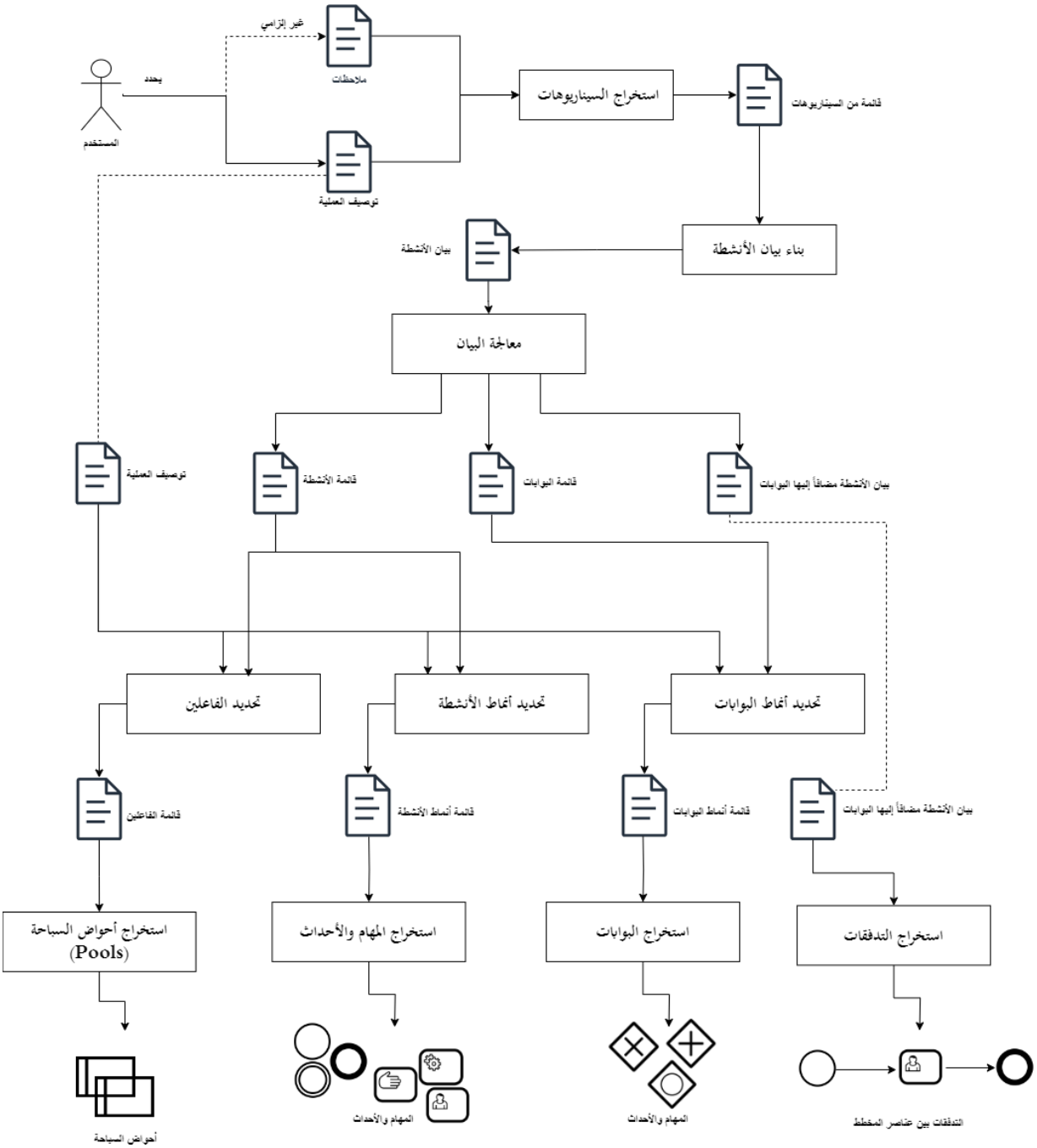


Figure 1 المخطط العام للمنهجية المقترحة

الفصل السادس

تصميم النظام

يَعرض هذا الفصل القرارات التصميمية التي بني من خلالها النظام.

1.6- مقدمة

تم اعتماد بنية الخدمات المصغرة (micro-services) لبناء التطبيق، إذ يتكون النظام من مجموعة من الخدمات التي تعمل مع بعضها بشكل متكامل لخدمة المستخدم النهائي. تقدم هذه البنية مجموعة من الفوائد أهمها:

- التوسع المستقل: يمكن توسيع كل خدمة مصغرة بشكل مستقل بناءً على الطلب، حيث تسمح هذه المرونة باستخدام الموارد بكفاءة، حيث يتم توسيع أجزاء التطبيق التي تحتاج إلى المزيد من الموارد فقط، بدلاً من النظام بأكمله.
- المرونة في استخدام التقنيات: يمكن استخدام لغات برمجة وأطر عمل وقواعد بيانات وتقنيات أخرى مختلفة للخدمات المصغرة المختلفة، وتحسين كل خدمة بناءً على متطلباتها المحددة، إذ يمكن أن يؤدي هذا إلى أداء أفضل وحلول أكثر تخصيصًا لكل خدمة.
- النشر المستقل: يمكن نشر الخدمات المصغرة بشكل مستقل، مما يسمح بالتكامل المستمر والتسليم المستمر (CI/CD) وهذا يعني أنه يمكن إصدار التحديثات وإصلاح الأخطاء بسرعة دون انتظار دورة إصدار النظام الكاملة.

نعرض في الفقرات التالية شرحاً لتصميم كل خدمة.

2.6- خدمة توليد مخططات BPMN

تم اتخاذ معمارية vertical slice architecture لبناء هذه الخدمة، إذ تقوم هذه المعمارية على تقسيم التطبيق إلى شرائح تحتوي كل شريحة على جميع طبقات الهندسة المعمارية (على سبيل المثال واجهة المستخدم، الاتصال مع قاعدة المعطيات ومنطق العمل...) لتوفير حالة استخدام كاملة مستقلة عن باقي حالات الاستخدام ضمن النظام.

تقدم هذه المعمارية مجموعة من الفوائد أهمها:

- تحترم مبدأ single responsibility حيث أن كل شريحة (slice) تحتوي على المنطق الكامل الخاص بميزة معينة (feature) أي حالة استخدام، مما يسمح باختبارها وصيانتها بسهولة بمعزل عن الميزات الأخرى التي يقدمها النظام.
- تحترم مبدأ High Cohesion بين مكونات الشريحة حيث أن كل مكونات الشريحة متماسكة فيما بينها وتخدم هدفاً واحداً.

من الجدير بالذكر أن هذه البنية تعاني من بعض المشاكل أهمها:

تكون مكونات الشريحة غير قابلة لإعادة الاستخدام ضمن شرائح أخرى، مما قد يؤدي إلى إعادة بناء بعض المكونات في شرائح أخرى كان من الممكن إعادة استخدامها من الشريحة الأساسية.

3.6- خدمة التخزين

هذه الخدمة مسؤولة بشكل أساسي عن تخزين الموارد الخاصة بالمستخدم من مخططات BPMN مع التوصيفات النصية الخاصة بها من أجل إعادة العمل عليها في وقت لاحق.

تعتبر هذه الخدمة بسيطة من حيث التنجيز إذ تقدم واجهتي تخاطب (get, set) يتم من خلالها تخزين واسترجاع الموارد، ومنه لا داعي لتعقيد التنجيز نكتفي باستخدام منهجية 3-tier الشهيرة التي تعتمد على وجود api يطلب خدمة منطق العمل (Business Logic Service) التي تتخاطب بدورها مع قاعدة المعطيات لتخزين واسترجاع الموارد.

من خلال فصل تخزين الموارد في خدمة مستقلة نحقق قابلية التوسع من حيث عدد المستخدمين من جهة، وقابلية التوسع في أنماط البيانات القابلة للتخزين ضمن النظام، حيث أن معالجة أي نوع جديد من الموارد سيتم ضمن هذه الخدمة بمعزل عن بقية أجزاء النظام مما يزيد من قابلية الصيانة ويسرع عملية التطوير.

4.6- خدمة إدارة عمليات النظام

تُعتبر هذه الخدمة هي الخدمة الأساسية ضمن النظام، حيث تقوم بإدارة إنشاء مشاريع ودعوة مستخدمين للمشاركة في هذه المشاريع من أجل بناء مخططات BPMN وإدارة المعلومات الخاصة بكل مشروع من مخططات وتوصيفات نصية للعمليات.

تم استخدام منهجية البنية سداسية الشكل (hexagonal architecture)، التي تقدم مجموعة من الخواص أهمها:

- عدم الاعتماد على التفاصيل التنجيزية الخارجية في معالجة منطق العمل (Business logic) حيث نستطيع تغيير هذه التفاصيل (مثل قاعدة المعطيات، مخدمات الرسائل، كيفية التواصل مع الخدمات الأخرى ...) دون الحاجة لتعديل باقي أجزاء النظام.
- تكون مركبات النظام غير مرتبطة مع بعضها البعض بقوة (Loosely coupled) مما يقدم إمكانية التعديل والصيانة بسهولة.

5.6- خدمة البوابة (Gateway)

بعد النظر إلى الخدمات التي تكوِّن النظام الكلي، نجد أن تواصل المستخدم مع كل خدمة على حدا يشكل مجموعة من المشاكل أهمها:

- عدم وجود واجهة موحدة تمثل النظام ككل يمكن للمستخدم التواصل معها.
- يجب على كل خدمة أن تقوم بإدارة الصلاحيات والتأكد من مصادقة المستخدم، مما يؤدي إلى تكرار تنجيز هذه الإدارة ضمن كل خدمة.

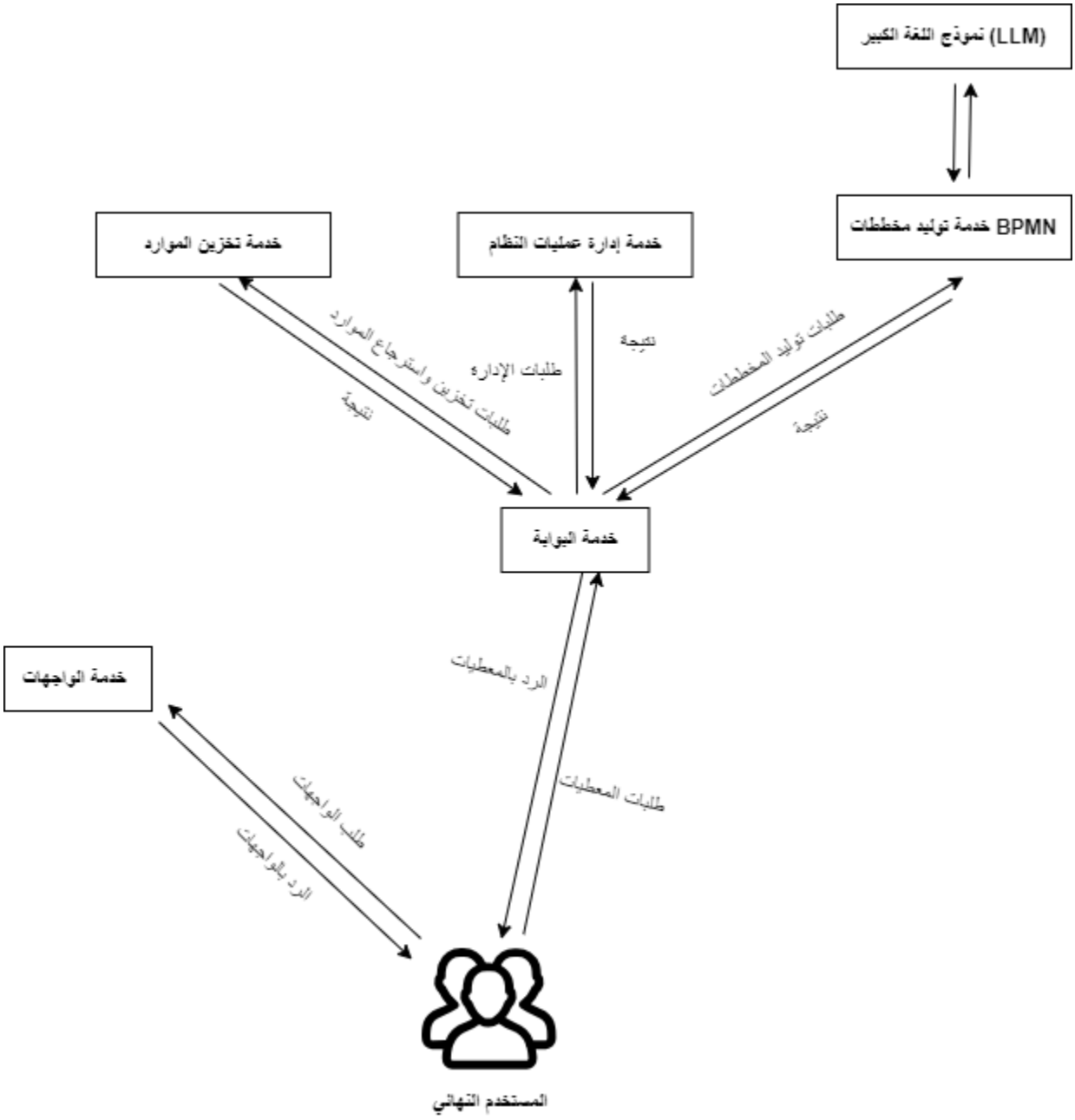
لذا ظهرت الحاجة لوجود مخدم يعمل كواجهة تخاطبية للنظام الكلي يتأكد من مصادقة المستخدمين ويقوم بإدارة الصلاحيات وتوجيه الطلبات للخدمات الداخلية حسب الطلب.

6.6- خدمة واجهة المستخدم (Front-End)

تم بناء واجهة المستخدم باستخدام بنية قائمة على المكونات مما يوفر العديد من الفوائد أهمها:

- النسيقية (Modularity): تسمح بتقسيم موقع الويب إلى وحدات أصغر (مكونات) مستقلة. يغلف كل مكون وظيفة محددة، مما يجعل إدارته وفهمه أسهل.
- قابلية إعادة الاستخدام: بمجرد إنشاء أحد المكونات، يمكن إعادة استخدامه عبر أجزاء مختلفة من موقع الويب أو حتى في مشاريع مختلفة. يقلل هذا من التكرار ويسرع التطوير ويضمن الاتساق في واجهة المستخدم والسلوك.
- سهولة تصحيح الأخطاء: تعمل المكونات المعزولة على تبسيط عملية تصحيح الأخطاء، حيث يمكن تتبع المشكلات إلى مكونات معينة بدلاً من البحث ضمن قاعدة الرماز الكبيرة (huge codebase).

7.6- مخطط النظام التصميمي



الفصل السابع

الأدوات المستخدمة

يَعرض هذا الفصل الأدوات المستخدمة لتنجيز المشروع.

Docker -1.7

هي منصة مفتوحة مصممة لأتمتة عملية نشر التطبيقات وتوسيع نطاقها (scaling) وإدارتها في حاويات خفيفة الوزن (containers). الحاوية (container) هي بيئة معزولة تجمع بين التطبيق وجميع تبعياته، مما يسمح بتشغيله بشكل متسق عبر بيئات حوسبة مختلفة، مثل أنظمة تشغيل مختلفة أو مزودي الخدمات السحابية.

PostgreSQL -2.7

هو نظام إدارة قواعد بيانات علائقية مفتوح المصدر، معروف بمتانته وقابليته للتوسع والالتزام بمعايير SQL. يدعم أنواع بيانات متقدمة واستعلامات معقدة، مما يجعله مناسباً للتعامل مع مجموعات البيانات الكبيرة والمعقدة. يُستخدم PostgreSQL على نطاق واسع في تطبيقات الويب خصوصاً تلك التي تتطلب تخزينًا واسترجاعًا موثوقًا للبيانات.

React -3.7

هي مكتبة JavaScript مفتوحة المصدر، تُستخدم لبناء واجهات المستخدم، خاصة لتطبيقات الويب. تم تطويرها بواسطة شركة Facebook، وتستخدم على نطاق واسع لإنشاء واجهات مستخدم ديناميكية وتفاعلية. تتبع React البنية القائمة على المكونات، مما يساعد المطورين على بناء مكونات واجهة المستخدم التي يمكن إعادة استخدامها وتكوينها معًا لإنشاء واجهات مستخدم معقدة.

NextJS -4.7

هو إطار عمل لمكتبه React مصمم لبناء تطبيقات ويب كاملة (full stack). حيث يبسط التطوير من خلال إدارة الأدوات اللازمة لـ React مثل التجميع (bundling)، حتى يتمكن المطورون من التركيز على بناء تطبيقاتهم.

يقدم إطار العمل NextJS مجموعة من المزايا أهمها:

- التوجيه (Routing): يوفر موجهًا قائمًا على نظام الملفات مع دعم للتخطيطات (Layouts) وصفحات معالجة الأخطاء والتحميل.
- العرض (Rendering): يدعم العرض من جانب العميل (client-side) ومن جانب الخادم (server-side)، مع تحسينات للمحتوى الثابت والديناميكي، بعكس React التي لا تدعم إلا طريقة العرض من جانب العميل.
- التحسينات (Optimizations): توفر أدوات لتعزيز الأداء عند التعامل مع الصور والخطوط والبرامج النصية، مما يؤدي إلى تجربة مستخدم أفضل.

MongoDB -5.7

MongoDB هو برنامج قاعدة بيانات NoSQL مفتوح المصدر شائع الاستخدام. يندرج ضمن فئة قواعد البيانات الموجهة للمستندات (document-oriented databases)، مما يعني أنه يخزن البيانات بتنسيق يشبه JSON يُعرف باسم BSON (JSON الثنائي). تم تصميم MongoDB ليكون مرنًا وقابلًا للتوسع وسهل التعامل معه لكل من المطورين والمسؤولين (administrators).

Go (Golang) -6.7

هي لغة برمجة ذات أنماط ثابتة (statically typed) ومترجمة (compiled) كما أنها مفتوحة المصدر، تم بناؤها بواسطة شركة Google. تتميز بدعمها للتزامن (Concurrency) والتشغيل التفرعي، حيث يمكن استخدامها لبناء تطبيقات يمكنها التعامل مع أعداد كبيرة من المستخدمين مع حركة مرور عالية (high traffic).

ASP.NET Web API -7.7

هو إطار عمل طورته شركة Microsoft يتيح إنشاء خدمات تعتمد على HTTP. يمكن استخدام هذه الخدمات بواسطة مجموعة متنوعة من العملاء، بما في ذلك متصفحات الويب وتطبيقات الأجهزة المحمولة وخدمات الويب الأخرى. إنه جزء من إطار عمل ASP.NET الأكبر وهو مصمم خصيصًا لإنشاء واجهات برمجة تطبيقات RESTful.

Json Web Token (JWT) -8.7

الرمز (JWT) هو غرض JSON يُستخدم 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 تطبيقات الويب، تطبيقات الجوال وتطبيقات سطح المكتب (Desktop application).

ReactFlow -9.7

React Flow هي مكتبة لبناء تطبيقات تعتمد على العقد (Nodes) باستخدام المكتبة React. ويمكن أن تكون هذه التطبيقات مخططات بسيطة أو تصورات للبيانات (data visualization) أو حتى محررات مرئية معقدة. تقدم المكتبة مجموعة جيدة من المكونات التي تسرع عملية التطوير، كما تسمح بإضافة مكونات مخصصة حسب الحاجة.

Git -10.7

هو نظام تحكم في الإصدارات (version control system) مفتوح المصدر يستخدم لتتبع التغييرات في الرماز المصدري (source code) أثناء تطوير البرمجيات. حيث يسمح Git لمطورين متعددين بالتعاون في مشروع واحد من خلال إدارة التغييرات التي تم إجراؤها على الرماز المصدري، مما يجعل من السهل تنسيق العمل وتتبع التغيرات ضمن المشروع.

GitHub -11.7

هي منصة تعمل ضمن الويب توفر أدوات للتحكم في الإصدارات والتعاون بين المطورين لتطوير البرامج، حيث أنها مبنية على نظام التحكم في الإصدارات Git. تسمح هذه المنصة بتخزين الكود البرمجي ومشاركته بين مجموعة من المطورين كما تقدم مجموعة من الأدوات للتعامل مع بيئات التطوير المستندة إلى السحابة (cloud) وإستضافة مواقع الويب الثابتة.

Jenkins -12.7

هو خادم أتمتة مفتوح المصدر يستخدم لتحقيق التكامل المستمر (CI) والتسليم المستمر (CD) في تطوير البرمجيات. فهو يساعد في أتمتة أجزاء من عملية تطوير البرمجيات من خلال إنشاء خطوط أنابيب (Pipelines) تعمل على بناء واختبار ونشر الرماز البرمجي تلقائيًا عند اكتشاف التغييرات في نظام التحكم في الإصدار (source control system) مثل Git، حيث يقدم Jenkins واجهة ويب يمكن إدارة عمليات الأتمتة من خلالها.

Ubuntu Server -13.7

هو أحد إصدارات نظام التشغيل Ubuntu مخصص للإستخدام ضمن المخدمات (Servers)، يوفر منصة مستقرة وموثوقة لاستضافة تطبيقات المخدمات المختلفة مثل خوادم الويب وقواعد البيانات. يقدم إدارة قوية للحزم ويركز بشكل أساسي على الأمان، مما جعله مستخدماً على نطاق واسع في مجال المخدمات. يتم إدارته بشكل أساسي من خلال واجهة سطر الأوامر (command line interface).

Grafana K6 -14.7

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

الفصل الثامن

تنجيز النظام

يَعرض هذا الفصل كيفية تنجيز النظام مع تفصيل كل جزء من أجزاءه.

1.8- مقدمة

خلال عملية تطوير الخدمات يتم رفع الرماز المصدري الخاص بكل خدمة باستخدام الأداء Git إلى مستودعات (Repositories) ضمن الموقع GitHub، حيث يتم تجميع الرماز المصدري ضمن فرع (branch) يسمى dev يَرمز إلى مرحلة التطوير (development). بالطبع ليست كل عمليات الدفع (push) يتم إرسالها مباشرةً إلى الفرع dev، في بعض الحالات يتم إنشاء فروع جديدة بناءً على الفرع dev يتم دمجها مع الفرع dev في وقت لاحق عند انتهاء العمل على الميزة.

يحقق هذا الأسلوب مبدأ (Continues Integration) حيث يتم تجميع الرماز البرمجي لكل خدمة ضمن مستودع مركزي يمكن جميع المطورين (حالياً المتمثلين بمحمد التركي فقط) من الوصول والعمل على نفس الرماز. كما يمكننا من اختباره عن طريق دمج الرماز البرمجي ضمن الفرع test، حيث يتم تفعيل خط الانابيب (Pipeline) الخاص بالاختبارات (إن وجدت) من أجل التأكد من صحة التنجيز.

نعرض في الفقرات التالية شرحاً تفصيلياً لكل خدمة.

2.8- خدمة توليد مخططات BPMN

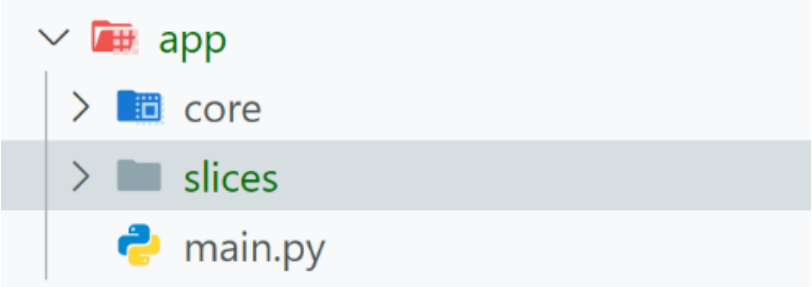
1.2.8- مقدمة

تم بناء هذه الخدمة باستخدام إطار العمل fastapi بلغة البرمجة python، تم اختيار python عن غيرها من لغات البرمجة بسبب الدعم الكبير لها في مجال التعامل والاتصال مع نماذج اللغات الكبيرة إذ تحتوي على العديد من المكتبات أهمها: (langchain, crewAi, langFlow, groq, …) التي تساعد على بناء التطبيق بشكل سريع وتحقيق مبدأ don’t repeat yourself (DRY).

تم الاعتماد على إطار العمل fastapi بدلاً من flask و Django، لكونه إطار العمل الأسرع كأداء ضمن هذه الإطارات ولكونه مدعوماً (من حيث المكتبات والمجتمع) أكثر من flask في المقام الأول، وبالرغم من أنه مدعوم بشكل أقل من Django ولكنه يعد كافياً في مجال بناء RESTAPI ويقدم المطلوب.

2.2.8- تفاصيل التنجيز

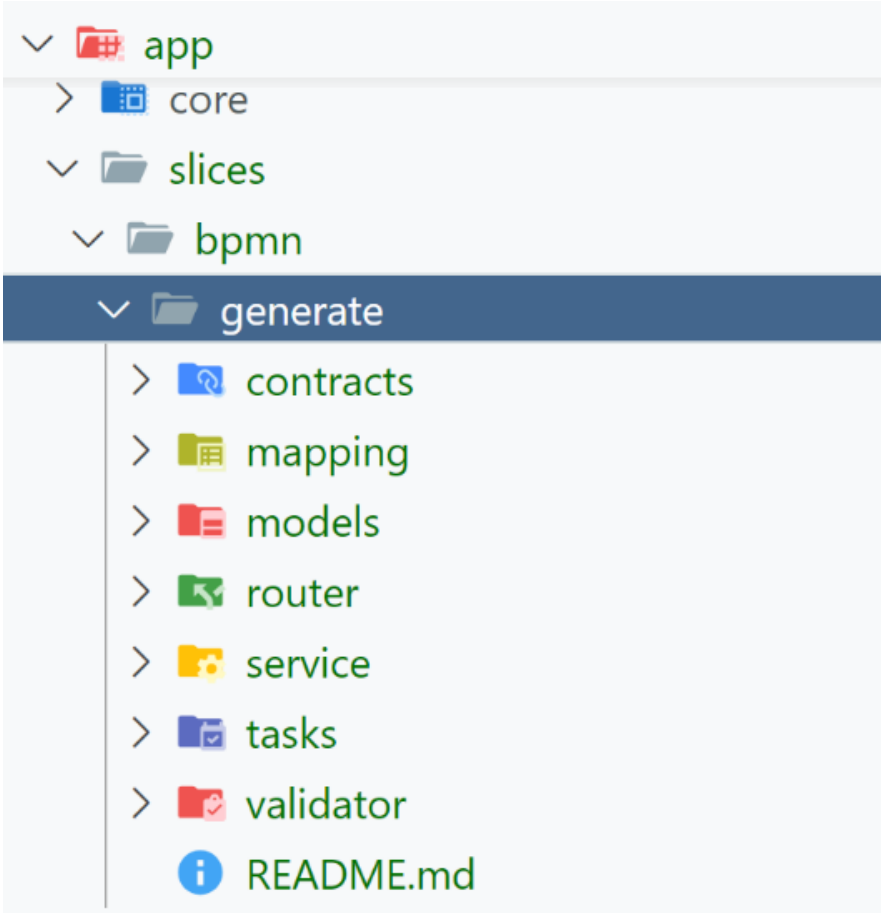
يتألف النظام من قسمين، الأول هو نواة المشروع (core) الذي يحتوي على مجموعة من المكونات التي لا تنتمي بشكل خاص لأي شريحة تساعد جميع الشرائح على أداء مهامها، وتُعتَبر مكونات قابلة لإعادة الإستخدام (reuseable components).



تَتْبَع شرائح النظام جميعا إلى نفس الهيكلية، إذ تتألف كل شريحة من المكونات التالية:

- ReadMe file: يحتوي على توصيف نصي للشريحة وما تقوم به.
- Contracts: يحتوي على الأنماط التي تستقبلها وتعيدها هذه الشريحة عند الطلب.
- Validator: يحتوي على المعالجات التي يتم تطبيقها على الطلب الوارد للتأكد من صحته.
- Models: يحتوي على الأنماط التي ستتعامل معها الشريحة لتطبيق منطق (Business Logic) معين.
- Mapping: يحتوي على منطق تحويل الأنماط بين Contracts وModels وبالعكس.
- Tasks: تحتوي على ملفات تحتوي على التوجيهات النصية التي سيتم إرسالها للنموذج اللغوي، حيث أن وضعها في ملفات يسهل عملية تعديلها من أجل الوصول لتوجيهات أفضل بدون الحاجة لتعديل الرماز المصدري (source code).

- Service: تحتوي على المنطق الأساسي للشريحة إذ تعتبر المكان الأساسي الذي يتم فيه تطبيق منطق العمل الخاص بالشريحة فيه (Business Logic).
- Router: يتم فيه استدعاء هذه المكونات وتنسيق التواصل فيما بينها من خلاله، كما يتم تعريف مسار طلب Http من خلال هذا المكون.



صورة **1** مكونات الشريحة الواحدة ضمن النظام.

نشرح في الفقرات التالية الشرائح التي تعبر عن الميزات (features) المقدمة ضمن هذه الخدمة، كل الشرائح تتبع للبنية ذاتها، لذا نكتفي بشرح الميزة التي تقدمها.

3.2.8- شريحة توليد مخططات BPMN

تقوم بتوليد مخططات BPMN باستخدام المنهجية المقترحة في الفصل الثالث، حيث تم استخدام النموذج (Llama-3-70B) كنموذج لغوي من خلال التواصل مع منصة Groq التي تقدم API Key مجاناً مع حدود طلبات مناسبة، 30 طلب بالدقيقة.

4.2.8- شريحة تصحيح أخطاء الكتابة ضمن توصيف العملية

تقوم بتصحيح الأخطاء اللغوية ضمن توصيف العملية، حيث يتم إرسال أمر مباشر للنموذج اللغوي الذي يقوم بدوره بتصحيح الأخطاء في حال وجودها.

5.2.8- شريحة توليد التقرير المساعد في رسم مخطط BPMN

نقوم في هذه الشريحة بإجراء محادثة بين نموذجين لغويين، كل منهما يصف نفسه على أنه خبير BPMN. هدف المحادثة هو توليد تقرير يحتوي على عناصر BPMN الخاصة بالتوصيف النصي للعملية، حيث أن هذا الأسلوب يرينا قدرة النماذج اللغوية على فهم العمليات واستخراج عناصر BPMN. يتم إرسال المحادثة للمستخدم على شكل تقرير PDF، يمكنه الاستفادة منه من أجل رسم المخطط.

3.8- خدمة التخزين

تم بناء هذه الخدمة باستخدام لغة البرمجة go التي تعتبر أحد أسرع اللغات من ناحية الأداء في مجال الويب. تم اعتماد هذه اللغة بشكل أساسي لأن عنق الزجاجة (bottleneck) ضمن النظام سيكون حفظ واسترجاع المخططات التي يعمل عليها مستخدمو النظام، لذا كان لابد من استخدام هذه اللغة ذات الأداء القوي.

تم استخدام قاعدة المعطيات mongodb التي تتبع منهجية nosql لتخزين هذه الموارد، حيث يعتبر نمط المعطيات المراد تخزينه غير قابل للهيكلة باستخدام نماذج قواعد المعطيات العلائقية التقليدية. لذا نجد أن الحاجة أدت إلى استخدام مثل هذا النوع من قواعد المعطيات.

4.8- خدمة إدارة عمليات النظام

1.4.8- مقدمة

تم بناء هذه الخدمة باستخدام إطار العمل ASP.NET الذي يعتبر من أفضل أطر العمل خصوصاً في المشاريع الكبيرة التي تحتوي على منطق عمل كبير ويجب معالجته بأسلوب قابل للتوسع والاختبار.

تم استخدام قاعدة المعطيات postgresql العلائقية التي تعتبر أحد أقوى قواعد المعطيات مفتوحة المصدر، حيث أن تصميم النظام يفرض استخدام قاعدة معطيات علائقية لتخزين المعطيات الخاصة به.

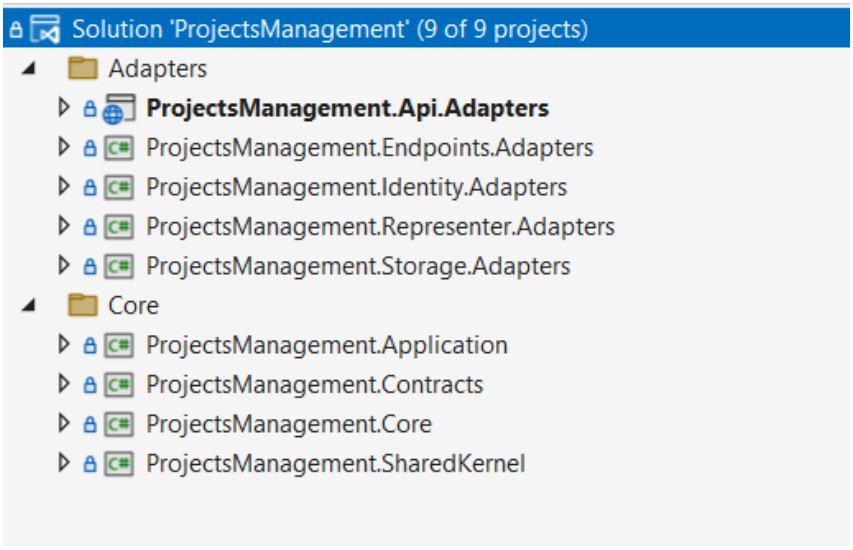
2.4.8- تنجيز الخدمة

تتكون الخدمة من مكونين أساسيين، نواة المشروع (Core) الذي يحتوي على المعالجات الأساسية لمنطق العمل والواجهات البرمجية التي تعبر عن الخدمات الخارجية (Ports) والمحولات (Adapters) التي تقوم بتنجيز واجهات الخدمات الخارجية.

تقدم الخدمة واجهة تخاطب Rest API يمكن الاتصال لها من خلال إرسال طلبات Http، وتتألف من مجموعة من الطبقات وهي:

- Shared Kenel: تحتوي على مجموعة من المكونات القابلة لإعادة الاستخدام ضمن أي مشروع.
- Core: تحتوي على الكيانات الموجودة ضمن النظام.
- Application: تحتوي على تنجيز منطق العمل الأساسي للخدمة، وهي مبنية باستخدام النمط التصميمي CQRS.
- Contracts: تحتوي على العقود التي ستستخدمها الطبقات الأعلى من أجل التخاطب مع الطبقات الأدنى واستدعائها.
- Storage: تحتوي على التنجيز من أجل الاتصال مع قاعدة المعطيات.

- Representers: تحتوي على طرق Methods تقوم بتحويل الأنماط المعادة من الطبقات الأدنى من أجل إرسالها عبر الويب.
- Identity: تحتوي على التنجيز من أجل التواصل مع خدمة إدارة المستخدمين (لم يتم شرحها بعد).
- End Points: تحتوي على الطرق والصفوف التي ستقوم بمعاجلة طلبات Http.
- API: يعتبر المشغل الأساسي للمشروع، مهمته الوحيدة هي تشغيل الطبقات جميعها.

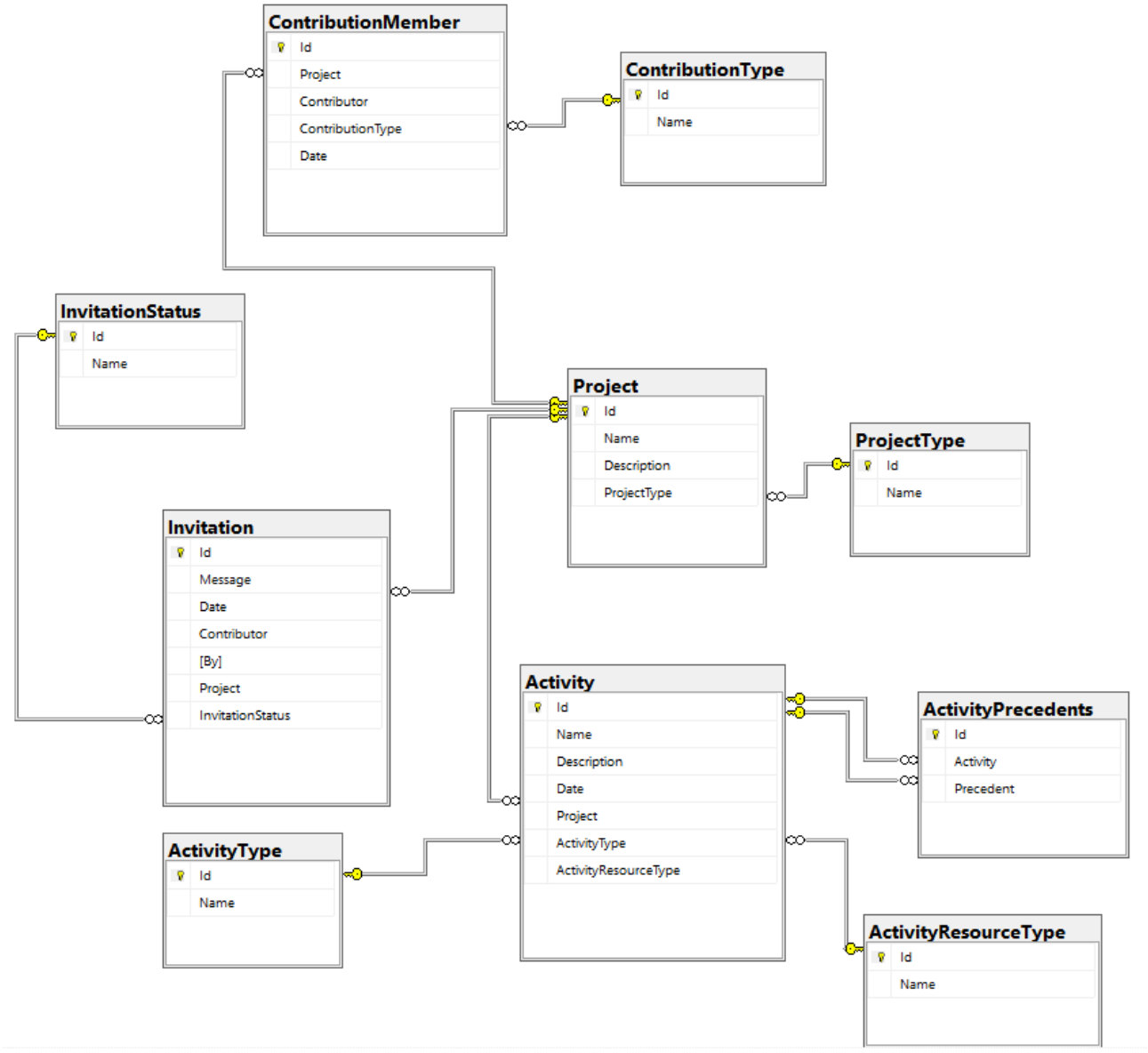


صورة **2** طبقات النظام

3.4.8- بعض الأنماط التصميمية الداخلية

تم استخدام النمط التصميمي الوسيط (Mediator) ضمن طبقة التطبيق (Application)، حيث أن جميع الطلبات الداخلة للنظام تمر عبر الوسيط الذي يقوم بمقاطعة الطلب ويقوم بتسجيله (Auditing) من أجل متابعة سلوك المستخدمين.

4.4.8- تصميم قاعدة المعطيات



تم مراعاة وضع القيم الثابتة عادةً التي توضع ضمن الكود البرمجي كنوع المشروع (public, private) ضمن جداول، هذه الجداول هي:

- ContributionType
- ActivityType

- ActivityResourceType
- InvitationStatus
- ProjectType

كل مشروع يحتوي على عدد من الأنشطة وعدد من المشاركين، وكل نشاط له نشاط سابق واحد أو أكثر يتم نمذجة هذه العلاقة ضمن الجدول ActivitiesPresedent.

5.8- خدمة البوابة (Gateway)

تم تنجيز خدمة البوابة باستخدام إطار العمل ASP.NET، حيث تم استخدام المكتبة Yarp لتنجيز عمليات توجيه الطلبات إلى الخدمات الأخرى والمكتبة ASP.NET Identity لتنجيز إدارة المستخدمين وصلاحياتهم وتسجيل الدخول وتوليد معرفات خاصة بكل مستخدم (JWT tokens).

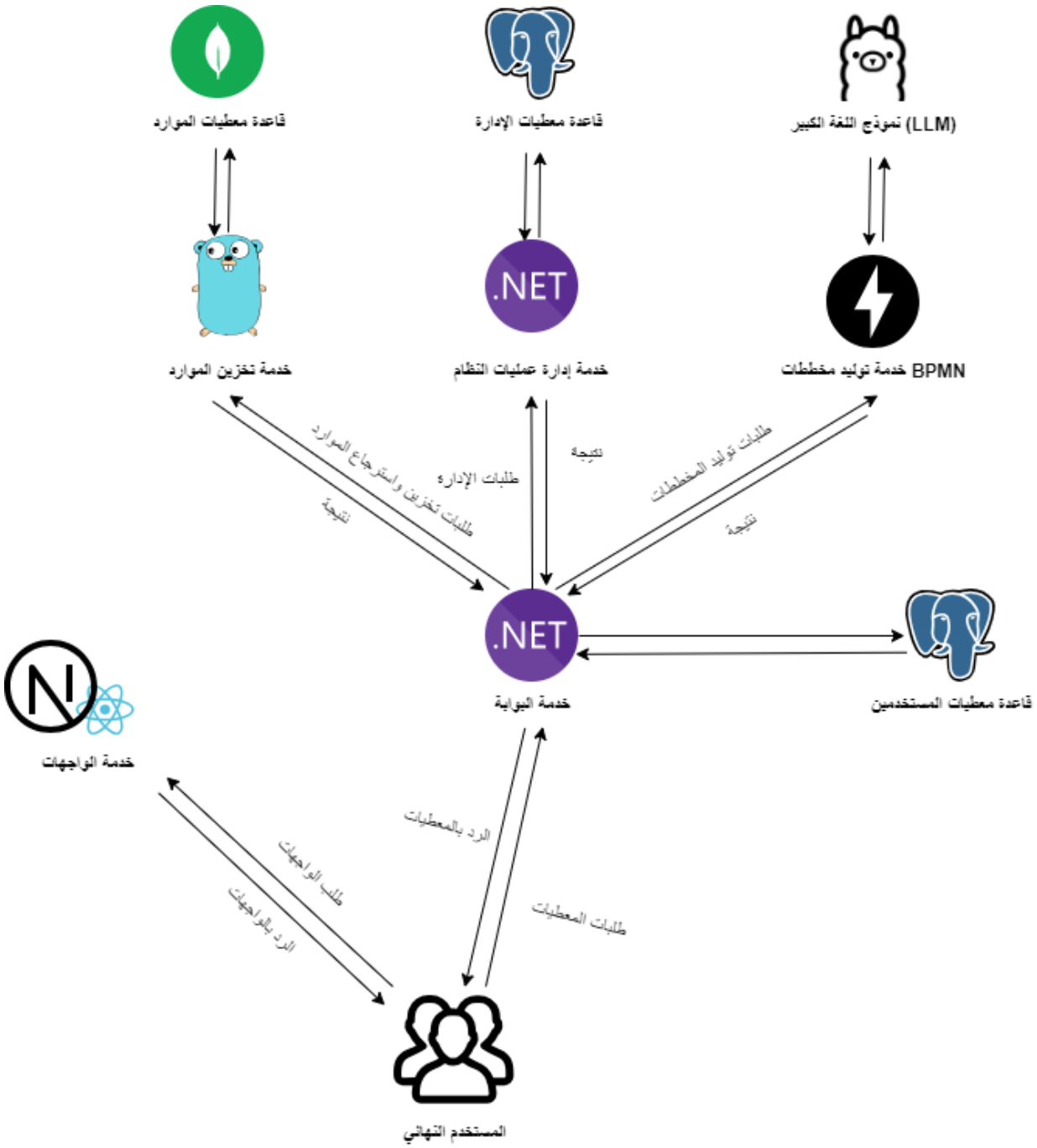
6.8- خدمة واجهة المستخدم

تم بناء واجهة المستخدم بإستخدام إطار العمل NextJS، حيث نستطيع الاستفادة من جميع خواص مكتبة React مع الخواص الإضافية التي يقدمها NextJS. إذ لا بد من استخدام إطار عمل يساعد على بناء واجهات تفاعلية من أجل رسم مخططات BPMN. إذ تم اختيار مكتبة ReactFlow لبناء الواجهة التي يتم عرض المخططات المتولدة من خلالها.

تم بناء واجهة عرض المخططات من البداية ولم يتم استخدام مكتبات جاهزة لعدة أسباب أهمها:

- من أجل تحقيق التحكم الكامل بالمخطط من أجل إضافة خواص الذكاء الصنعي وإدارة التكامل معها.
- التحكم بتنسيق المخطط الناتج، حيث أن معظم محررات BPMN تتعامل فقط مع أنماط XML.

7.8- مخطط النظام التنجيزي



8.8- نشر الخدمات

تم تجهيز البنية التحتية لنشر الخدمات عن طريق استخدام مخدم Ubuntu Server، حيث تم تثبيت تطبيقي Docker و Jenkins ضمن هذا المخدم.

من أجل أتمتة عملية نشر الخدمات، تم إنشاء خط أنابيب مؤتمت (Pipeline) باستخدام الأداة Jenkins، يعمل بشكل تلقائي عند حدوث تعديل ضمن الفرع master الخاص بمستودع الخدمة، إذ يتم تحميل الرماز المصدري الخاص بالخدمة وبناؤه ضمن حاوية (container) وتشغيله باستخدام بيئة العمل Docker.

من خلال اعتماد هذه المنهجية في نشر التطبيق تم تحقيق مجموعة من الميزات أهمها:

- الحفاظ على الرماز البرمجي من الضياع وإمكانية مشاركته ضمن فريق العمل بسهولة، حيث تم تحقيق ذلك من خلال استخدام الأداة Git والموقع GitHub.
- عدم الحاجة إلى إعادة بناء الخدمات بشكل يدوي عند تعديلها ضمن المستودعات، إذ يعمل Jenkins على أتمتة هذه العملية، مما يسرع عملية التطوير والنشر ويحقق هذا مبدأ (Continues Delivery).
- عدم الحاجة إلى إدارة الاعتماديات الخاصة بكل خدمة ودراسة تأثيرها على الخدمات الأخرى، حيث أن استخدام Docker يساعد على إنشاء حاوية معزولة وخاصة لكل خدمة تحتوي على الخدمة مع جميع تبعياتها.

الفصل التاسع

اختبارات النظام ومناقشة النتائج

يوضح هذا الفصل الاختبارات التي تم تطبيقها للتأكد من تلبيته للمتطلبات.

1.9- اختبار المنهجية المقترحة

تم تقييم النموذج المقترح باستخدام مجموعة البيانات PET [11]، إذ تم اتخاذ المنهجية المتبعة ضمن البحث [13] لتقييم النموذج المقترح، حيث تم استخدام ستة توصيفات نصية من مجموعة المعطيات كدخل للنموذج، وتم استخدام التعليقات التوضيحية لتحديد دقة النموذج. نقوم بتقييم النموذج بعدد المكونات الصحيحة (Recall) التي استطاع استخراجها من التوصيف النصي، كما نقتصر في تقييمنا على عدد المكونات التي تم اكتشافها من النوع (التدفقات، الفاعلين، البوابات، المهام والأحداث).

يتم التقييم يدوياً بعد معالجة مجموعة المعطيات لتحويلها إلى تنسيق مفهوم (مع الحفاظ على جميع العلاقات) لتسهيل فهم العلاقات. تم إرفاق النتائج ضمن مستودع github MohamadAlturky/bpmn-results (github.com).

2.9- اختبار خدمة التخزين

من أجل التأكد من قدرة خدمة التخزين على الاستجابة لعدد كبير من الطلبات، تم إنشاء اختبارات لأداء هذه الخدمة باستخدام الأداة k6، سيتم استعراض كيفية التجهيز للاختبارات مع نتائجها في الفقرات التالية.

1.2.9- التجهيز للاختبارات

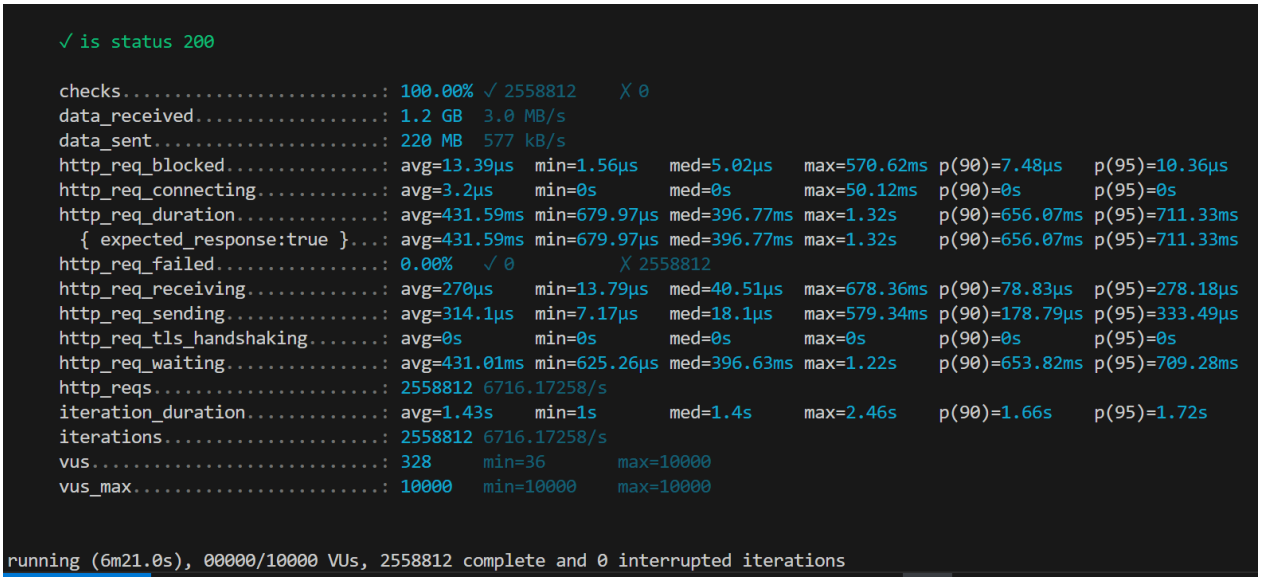
تم إضافة 15 مليون مخطط عشوائي المحتوى ضمن قاعدة المعطيات MongoDB، باستخدام go routines.

```
ubuntu@restaurant:~/fifthproj/Resources/stats$ go run main.go
The 'activities' collection contains 15280707 documents.
ubuntu@restaurant:~/fifthproj/Resources/stats$
```

2.2.9- اختبار الحمل (Load test)

```
export const options = {
  stages: [
    { duration: '5s', target: 1000 },
    { duration: '5s', target: 10000 },
    { duration: '360s', target: 10000 },
    { duration: '10s', target: 0 }
  ],
};
```

توضح محددات الاختبار عن البدء بالصعود لمدة 5 ثواني لألف مستخدم، ثم لمدة خمس ثواني أخرى لعشرة آلاف مستخدم، حيث يبقى عدد المستخدمين ثابتاً لمدة ست دقائق من أجل اختبار سلوك الخدمة للحمل المتوقع لعدد المستخدمين (الذي تم افتراضه على أنه عشرة آلاف). ننهي الاختبار بتقليل عدد المستخدمين تدريجياً لمدة عشر ثواني.



تظهر النتائج قدرة النظام على تحمل العبء بدون خسارة أي طلب، حيث أن متوسط الزمن اللازم للرد على الطلب هو 1.43s وأقل زمن استغرقته الخدمة للرد هو 1s وأعلى زمن هو 2.46s، مما يبين تلبية الخدمة للمتطلبات الموكلة لها من حيث تحمل العبئ. تم تخديم 2558812 طلب خلال زمن تنفيذ الاختبار.

3.2.9- اختبار الضغط

```
export const options = {
  stages: [
    { duration: '5s', target: 1000 },
    { duration: '5s', target: 20000 },
    { duration: '60s', target: 20000 },
    { duration: '10s', target: 0 }
  ],
};
```

توضح محددات الاختبار عن البدء بالصعود لمدة 5 ثواني لألف مستخدم، ثم لمدة خمس ثواني أخرى لعشرة آلاف مستخدم، حيث يبقى عدد المستخدمين ثابتاً لمدة دقيقة من أجل اختبار سلوك الخدمة عند حدوث ضغط يساوي ضعف عدد المستخدمين المتوقع. ننهي الاختبار بتقليل عدد المستخدمين تدريجياً لمدة عشر ثواني.

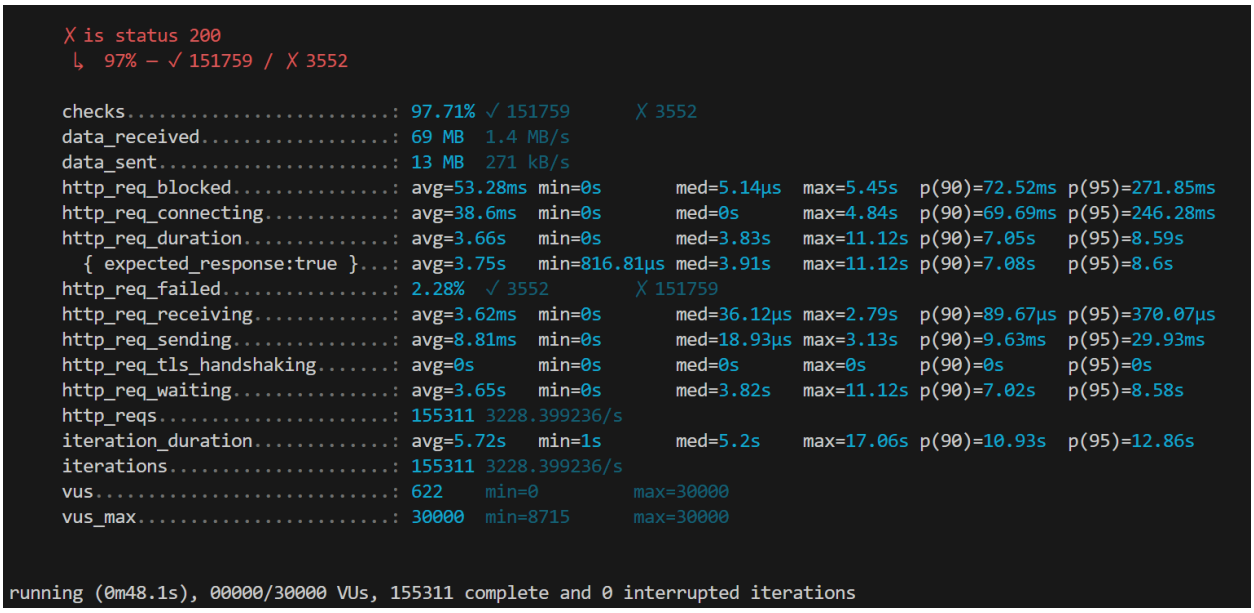
```
    ✓ is status 200

    checks.....................: 100.00% ✓ 396403      ✗ 0
    data_received..............: 180 MB  2.2 MB/s
    data_sent..................: 34 MB   420 kB/s
    http_req_blocked...........: avg=8.37ms min=1.47µs   med=5.22µs  max=2.45s  p(90)=11.93µs  p(95)=181.2µs
    http_req_connecting........: avg=7.57ms min=0s       med=0s      max=1.8s   p(90)=0s       p(95)=80.65µs
    http_req_duration..........: avg=2.3s   min=808.14µs med=1.82s   max=8.11s  p(90)=4.66s    p(95)=5.18s
      { expected_response:true }...: avg=2.3s   min=808.14µs med=1.82s   max=8.11s  p(90)=4.66s    p(95)=5.18s
    http_req_failed............: 0.00%   ✓ 0           ✗ 396403
    http_req_receiving.........: avg=2.02ms min=14.77µs  med=41.43µs max=1.45s  p(90)=105.49µs p(95)=311.36µs
    http_req_sending...........: avg=2.95ms min=6.9µs    med=19.02µs max=1.57s  p(90)=214.52µs p(95)=750.92µs
    http_req_tls_handshaking...: avg=0s     min=0s       med=0s      max=0s     p(90)=0s       p(95)=0s
    http_req_waiting...........: avg=2.3s   min=756.42µs med=1.82s   max=8.03s  p(90)=4.65s    p(95)=5.18s
    http_reqs..................: 396403  4889.935156/s
    iteration_duration.........: avg=3.36s  min=1s       med=2.82s   max=10.31s p(90)=5.77s    p(95)=6.29s
    iterations.................: 396403  4889.935156/s
    vus........................: 577     min=0         max=20000
    vus_max....................: 20000   min=11387     max=20000


running (1m21.1s), 00000/20000 VUs, 396403 complete and 0 interrupted iterations
```

تظهر النتائج قدرة النظام على تحمل الضغط المضاعف لعدد المستخدمين بدون خسارة أي طلب، حيث أن متوسط الزمن اللازم للرد على الطلب هو 2.3s وأقل زمن استغرقته الخدمة للرد هو 1.82s وأعلى زمن هو 8.03s، مما يبين تلبية الخدمة للمتطلبات الموكلة لها من حيث تحمل ضغط يساوي ضعف المتوقع. تم تخديم 396403 طلب خلال زمن تنفيذ الاختبار.

4.2.9- اختبار الارتفاع المفاجئ للطلبات

```
export const options = {
  stages: [
    { duration: '5s', target: 100 },
    { duration: '2s', target: 30000 },
    { duration: '30s', target: 30000 },
    { duration: '10s', target: 0 }
  ],
};
```

توضح محددات الاختبار عن البدء بالصعود لمدة 5 ثواني لمئة مستخدم، ثم لمدة ثانيتين نصعد لعشرة آلاف مستخدم، حيث يبقى عدد المستخدمين ثابتاً لمدة ثلاثين ثانية من أجل اختبار سلوك الخدمة عند حدوث ضغط مفاجئ يساوي ثلاث أضعاف المتوقع. ننهي الاختبار بتقليل عدد المستخدمين تدريجياً لمدة عشر ثواني.



تظهر النتائج قدرة النظام على تحمل الضغط المفاجئ لعدد المستخدمين مع خسارة 2.29% من الطلبات التي لم يتم تخديمها فقط، حيث أن متوسط الزمن اللازم للرد على الطلب هو 5.72s وأقل زمن استغرقته الخدمة للرد هو 1s وأعلى زمن هو 17.06s، مما يبين تلبية الخدمة للمتطلبات الموكلة لها من حيث تحمل الضغط الذي يساوي ثلاث أضعاف المتوقع. تم تخديم 151759 طلب خلال زمن تنفيذ الاختبار.

5.2.9- ملاحظات حول طبيعة الاختبار

تم تنفيذ الاختبارات ضمن مخدم ubuntu server، حيث أن الخدمة وأداة الاختبار تعملان ضمن نفس المخدم، مما يستهلك من موارد المخدم من جهة، ولكن بسبب أن الاتصال لم يحدث عبر الشبكة (اتصال محللي) فإن هذا الاختبار لا يختبر قدرات الشبكة، إنما يختبر فقط إمكانية تحمل للخدمة لعدد مستخدمين عالي.

3.9- اختبار خدمة إدارة عمليات النظام

تم إنشاء مجموعة من اختبارات الوحدة (Unit tests)، واختبارات التكامل (Integration tests) لاختبار مراعاة الخدمة للمتطلبات الموكلة إليها، تم إنشاء اختبارات وحدة لحالات الاستخدام الأساسية كما تم اختبار التكامل عند طلب خدمة المستخدمين من أجل معرفة أسماء المشاركين ضمن مشروع معين، يتم تفعيل هذه الاختبارات بشكل أوتوماتيكي عند القيام بدمج الرماز المصدري ضمن الفرع test للمستودع الخاص بالخدمة.

المراجع

[1] Von Rosing, M., White, S., Cummins, F., & De Man, H. (2015). Business Process Model and Notation-BPMN.

[2] Vaswani, A. (2017). Attention is all you need. *arXiv preprint arXiv:1706.03762*.

[3] Minaee, S., Mikolov, T., Nikzad, N., Chenaghlu, M., Socher, R., Amatriain, X., & Gao, J. (2024). Large language models: A survey. *arXiv preprint arXiv:2402.06196*.

[4] Ekin, S. (2023). Prompt engineering for ChatGPT: a quick guide to techniques, tips, and best practices. *Authorea Preprints*.

[5] Wang, Y., Yao, Q., Kwok, J. T., & Ni, L. M. (2020). Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)*, *53*(3), 1-34.

[6] Augusto, A., Conforti, R., Dumas, M., La Rosa, M., & Polyvyanyy, A. (2019). Split miner: automated discovery of accurate and simple business process models from event logs. *Knowledge and Information Systems*, *59*, 251-284.

[7] Paul, A., & Haldar, M. (2023). Continuous Integration and Continuous Delivery/Continuous Deployment. In *Serverless Web Applications with AWS Amplify: Build Full-Stack Serverless Applications Using Amazon Web Services* (pp. 233-256). Berkeley, CA: Apress.

[8] Liu, G., Huang, B., Liang, Z., Qin, M., Zhou, H., & Li, Z. (2020, December). Microservices: architecture, container, and challenges. In *2020 IEEE 20th international conference on software quality, reliability and security companion (QRS-C)* (pp. 629-635). IEEE.

[9] Cockburn, A. (2005). Hexagonal architecture. *The Pattern: Ports and Adapters*.

[10] Grohs, M., Abb, L., Elsayed, N., & Rehse, J. R. (2023, September). Large language models can accomplish business process management tasks. In *International Conference on Business Process Management* (pp. 453-465). Cham: Springer Nature Switzerland.

[11] Bellan, P., van der Aa, H., Dragoni, M., Ghidini, C., & Ponzetto, S. P. (2022, September). PET: an annotated dataset for process extraction from natural language text tasks. In *International Conference on Business Process Management* (pp. 315-321). Cham: Springer International Publishing.

[12] Zirnstein, B. Extraction of BPMN process models from unstructured textual descriptions.

[13] https://doi.org/10.1007/978-3-031-50974-2_34

[14] https://doi.org/10.1007/978-3-031-61007-3_18

[15] Kourani, H., & van Zelst, S. J. (2023, September). POWL: partially ordered workflow language. In *International Conference on Business Process Management* (pp. 92-108). Cham: Springer Nature Switzerland.

[16] BPMN Tutorial: Learn Business Process Model and Notation | Camunda

[17] About | bpmn.io

[18]

[19]

[20]