الجمهورية العربية السورية
جامعة البعث
كلية الهندسة المعلوماتية
قسم هندسة البرمجيات ونظم المعلومات

آفاق وتحديات استخدام منهجية DevOps في عالم تطوير البرمجيات

دراسة أُعِدَّت لنيل درجة الماجستير في كلية الهندسة المعلوماتية
باختصاص هندسة البرمجيات ونظم المعلومات

إعداد

المهندسة: دانه حافظ مدور

إشراف

الدكتورة: أليدا اسبر

أُستاذ مساعد في قسم هندسة البرمجيات ونظم المعلومات
كلية الهندسة المعلوماتية – جامعة البعث

1442 هـ – 2021 م

فهرس المحتويات

فهرس المحتويات III
الملخّص VII
Abstract VIII
قائمة الأشكال والمخطّطات IX
قائمة الجداول XI
الفصل الأوّل مقدّمة 12
1.1 مشكلة البحث 13
2.1 أهداف البحث وأهميته 13
3.1 مجال البحث 14
4.1 منهجية البحث المستخدمة 14
5.1 مخطط الأطروحة 15
الفصل الثّاني الدّراسة النظريّة 18
1.2 منهجيات تطوير البرمجيات Software Development Methodologies 18
أسباب ظهور منهجيات تطوير البرمجيات 19
نسبة الفشل في المشاريع البرمجية 19
1.1.2 نماذج التطوير التقليدية أو المقادة بخطة Plan-Driven Development Methodologies 20
2.1.2 تطوير البرمجيات السريع Agile Software Development 23
3.1.2 تطوير البرمجيات الرشيق Lean Software Development 28
4.1.2 منهجية DevOps 32
2.2مقارنة بين منهجيات تطوير البرمجيات المدروسة 37

الفصل الثّالث الدّراسة المرجعيّة 41
1.3 منهجية البحث 41
2.3 السياق..... 41
3.3 الدراسات المرجعية 41
1.3.3 الدراسات المرجعية ما قبل DevOps 42
2.3.3 الدراسات المرجعية التي تحدثت عن DevOps..... 43
4.3 تحليل الدراسات المرجعية..... 46
1.4.3 القيمة التي أضافتها الدراسات المرجعية للبحث 47
الفصل الرّابع 50
إطار عمل DevOps المقترح والأدوات المستخدمة 50
مقدمة 50
1.4 منهجية البحث 50
2.4 إطار عمل DevOps المقترح..... 50
1.2.4 مرحلة التطوير 51
2.2.4 مرحلة الاختبار..... 53
3.2.4 مرحلة التكامل 53
4.2.4 مرحلة النشر..... 53
5.2.4 مرحلة المراقبة 53
3.4 التقنيات والأدوات المعتمدة..... 53
1.3.4 الأدوات المختارة في سياق الدراسة الحالية..... 54
خاتمة 58
الفصل الخامس دراسة حالتين: تطوير تطبيقي وب.....60
1.5 منهجية البحث 60
1.1.5 تصميم الدراسة واختيار الحالة Study Design and Case Selection 61

2.1.5 جمع البيانات Data Collection..... 62

3.1.5 تحليل البيانات Data Analysis..... 62

4.1.5 الأسئلة التي سيجاب عنها بعد تحليل دراستي الحالة 64

2.5 وصف دراستي الحالة..... 65

1.2.5 نظام سند الموزع بالتعاون مع UNCHR 65

2.2.5 المكتبة الالكترونية في كلية الطب البشري – جامعة البعث 73

3.5 سياق العمل في كل مرحلة..... 79

1.3.5 دراسة الحالة الأولى: نظام سند الموزع بالتعاون مع UNCHR..... 79

2.3.5 دراسة الحالة الثانية: المكتبة الالكترونية في كلية الطب البشري – جامعة البعث 80

خاتمة 86

الفصل السّادس النتائج والمقترحات المستقبلية 88

مقدمة..... 88

1.6 منهجية البحث 88

2.6 الاستنتاجات المستخلصة من دراستي الحالة 88

1.2.6 المناقشة..... 91

3.6 نتائج البحث 92

4.6 المقترحات المستقبلية 93

خاتمة 94

قائمة المصطلحاتList of Terms 96

قائمة الاختصاراتList of Acronyms 100

المراجع..... 102

الملخّص

مع التطور التكنولوجي الهائل الذي يشهده العالم اليوم، يتزايد الطلب على تطوير برمجيات جديدة قادرة على مواكبة المتطلبات المتغيرة التي تظهر عند مختلف المستخدمين باستمرار والقادرة على العمل على مختلف المنصات التقنية الجديدة، الأمر الذي شكلّ تحدياً كبيراً للباحثين في مجال هندسة البرمجيات بهدف إيجاد أفضل الممارسات والمنهجيات لتحسين عملية التطوير البرمجي، ومن ناحية أخرى زاد من العبء الكبير المنصب على المطورين وفرق العمليات ضمن شركات تطوير البرمجيات.

لذلك استمر الباحثون بتطوير منهجيات تطوير البرمجيات، وتعديل أساليب عملها بناء على النتائج المستخلصة من التجارب التطبيقية البحثية والتجارب العملية للممارسين وشركات التطوير البرمجي.

بناء على ذلك قمنا في هذا البحث بدراسة إحدى أحدث منهجيات تطوير البرمجيات وهي منهجية DevOps، وبناء على هذه الدراسة وضعنا إطار عمل مناسب يدمج ممارسات هذه المنهجية مع أفضل الممارسات المستخلصة من منهجيات تطوير برمجية أخرى أثبتت فعاليتها في عملية التطوير البرمجي.

لاحقاً قمنا باختبار إطار العمل هذا من خلال تطبيقه في عملية تطوير برمجي لمنصة الكترونية تقوم بإدارة المراجع العلمية لإحدى كليات جامعة البعث مقادة بفريق عمل برمجي، وقارنا العملية بعملية تطوير أخرى لتطبيق مشابه تمت من قبل فريق تقانة معلومات تابع لمنظمة أخرى، ويستخدم منهجية تطوير سريعة Agile.

بينت النتائج التي توصلنا إليها بعد عملية الدراسة والمقارنة أن إطار عمل DevOps الذي تبنيناه قام بتحسين عملية التطوير البرمجي من حيث السرعة في العمل وتحسين أداء جميع الفرق من خلال إشراكهم في عملية التطوير منذ البداية وتحسين التواصل فيما بينهم، وأكدت أيضاً أن استخدام الأدوات المناسبة وأتمتة العمليات بأكبر قدر يحسن من عمليات هندسة البرمجيات ويحولها إلى عملية قابلة للتتبع والرصد بشكل أكبر، الأمر الذي يعتبر من أكبر المشاكل التي تواجه هذا المجال.

الكلمات المفتاحيّة: هندسة البرمجيات – منهجيات تطوير البرمجيات – DevOps – منهجيات التطوير السريعة – التطوير – دراسة حالة – الأدوات – الأتمتة.

Abstract

With the tremendous technological development that the world is witnessing today, there is an increasing demand for developing new software that is capable of keeping pace with the changing requirements that appear among various users constantly and that is able to work on the latest various platforms, which posed a great challenge for researchers in the field of software engineering in order to find the best methodologies and practices to improve the software development process, and on the other hand, it increased the great burden placed on developers and operations teams within software development companies.

Therefore, researchers continued to develop new software development methodologies and to modify their working methods based on the results obtained from the applied researches and the practical experiences of practitioners in software development companies.

Accordingly, in this research, we studied one of the latest software development methodologies, which is the DevOps methodology, and based on this study, we developed a framework that integrates the practices of this methodology with the best practices extracted from other software development methodologies that have proven effectiveness in the software development process.

Later, we tested this framework by applying it in a software development process run by a team with the aim of developing an electronic platform that manages the Scientific References for one of the colleges of Al-Baath University, and we compared the process to another development process for a similar application that was carried out by an information technology team of another organization, who used an Agile Development Methodology.

Our findings showed that the adopted DevOps framework has improved the software development process in terms of the speed of work and the performance of all teams by involving them in the development process from the beginning and encouraging communication between them. It also confirmed that automation and the use of the right tools add an extra improvement in tracking and monitoring software engineering processes, and it turns them into a more traceable process, which was one of the biggest problems facing this field.

Keywords: Software Engineering - Software Development Methodologies - DevOps - Agile Development Methodologies - Development - Case Study - Tools - Automation.

قائمة الأشكال والمخططات

الشكل (2-1) مخطط هيكلي يظهر الأسباب الرئيسية لفشل المشاريع البرمجية 17
الشكل (2-2) نموذج الشلال 18
الشكل (2-3) النموذج الحلزوني 20
الشكل (2-4) أسلوب العمل وفقاً لمنهجية البرمجة المتطرفة 22
الشكل (2-5) أسلوب العمل وفقاً ل Scrum 26
الشكل (2-6) أسلوب العمل وفقاً ل Kanban 28
الشكل (2-7) أسلوب العمل وفقاً ل DevOps 30
الشكل (4-1) منهجية DevOps 47
الشكل (4-2) منهجية Scrum 48
الشكل (4-3) لوح Kanban 49
الشكل (4-4) الأدوات المقدمة من Azure DevOps Server 51
الشكل (4-5) واجهة التنصيب ل Azure DevOps Server 53
الشكل (4-6) واجهة إدارة Azure DevOps Server 53
الشكل (4-7) واجهة إدارة طبقة التطبيقات 54
الشكل (4-8) واجهة إدارة المشاريع Azure DevOps Server 54
الشكل (4-9) واجهة المشروع في Azure DevOps Server 55
الشكل (5-1) مخطط تقديم الخدمات في نظام سند الموزع 63
الشكل (5-2) لوحة التحكم في نظام سند الموزع 64
الشكل (5-3) التقارير الإحصائية الخاصة بالمراكز 65
الشكل (5-4) المخطط الخطي لسير العمل ضمن مشروع سند التابع للمنظمة 65
الشكل (5-5) استعراض العائلات وإمكانية إضافة عائلة جديدة 66
الشكل (5-6) استعراض بيانات عائلة 66
الشكل (5-7) إضافة بيانات شخص إلى عائلة 67
الشكل (5-8) واجهة تحويل الإحالات إلى مدراء الحالات 67
الشكل (5-9) واجهة استعراض الحالات وفقاً لمراحلها 68
الشكل (5-10) مخطط يوضح طريقة العمل ضمن المكتبة الالكترونية 72
الشكل (5-11) الصفحة الرئيسية لتطبيق المكتبة الالكترونية 74
الشكل (5-12) المتطلبات الخاصة بتطبيق المكتبة الالكترونية منظمة باستخدام Azure DevOps Server 2019 77

الشكل (5-13) قصص المستخدمين منظمة ضمن Azure Boards 78

الشكل (5-14) اختيار الخدمة المناسبة لعملية نشر إصدار جديد من التطبيق باستخدام Azure Pipelines 78

الشكل (5-15) اختيار الميزات التي ستنشر في الإصدار الحالي 79

الشكل (5-16) نجاح عملية الإصدار 79

الشكل (5-17) ضبط إعدادات نشر الإصدارات الجديدة باستخدام Azure Pipelines 80

الشكل (5-18) مستودع الشيفرة المصدرية Azure Repos 81

الشكل (5-19) المخطط التراكمي لقياس عدد المهام المنجزة خلال ال Sprint..... 81

قائمة الجداول

الجدول (2-1) معدلات المشاريع البرمجية الناجحة والفاشلة والتي تواجه تحديات بالنسبة ل Standish Group 18

الجدول (2-2) مقارنة بين منهجيات التطوير المختلفة 34

الجدول (4-1) يعرض الأدوات الممكن استخدامها في كل مرحلة تطوير باستخدام DevOps 51

الجدول (5-1) وصف للعاملين في كل دراسة حالة والملاحظات 58

الجدول (5-2) وصف للسمات المستخدمة لترميز البيانات المستخلصة 59

الجدول (5-3) وصف ملخص عن سياق دراستي الحالة 59

الجدول (5-4) مقارنة بين أشهر تطبيقات إدارة الكتب الالكترونية 71

الجدول (6-1) المدة الخاصة بكل مرحلة من مراحل التطوير 86

الفصل الأوّل

مقدّمة

مع التطور التكنولوجي الهائل الذي يشهده العالم اليوم، يتزايد الطلب على تطوير برمجيات جديدة قادرة على مواكبة متطلبات المستخدمين المتغيرة باستمرار، وعلى العمل على مختلف المنصات التقنية الجديدة، الأمر الذي زاد من العبء المنصب على المطورين وفرق العمليات ضمن شركات التطوير البرمجية.

لذلك الهدف تم طرح منهجيات وآليات عمل لتنظيم عملية التطوير البرمجي دعيت بمنهجيات تطوير البرمجيات – ونعني بمنهجيات التطوير مجموعة من النشاطات المرتبطة مع بعضها التي تقود إلى إنتاج المنتج البرمجي، والتي تتضمن تطوير البرمجيات من الصفر باستخدام لغة برمجة معينة أو بالتوسعة والتعديل على نظام موجود مسبقاً وذلك بتهيئة إعداداته ودمجها مع المكونات البرمجية الأخرى الخاصة بالنظام [1]. كان أولها النماذج التقليدية والتي ظهرت في سبعينيات القرن الماضي، وأطلق عليها النماذج المقادة بخطة أو التقليدية، أشهرها نموذج الشلال waterfall عام 1970 [1]، واجه هذا النوع من النماذج مجموعة من المشكلات، أهمها الفترة الزمنية الطويلة اللازمة لتسليم المنتج النهائي، الذي أدى غالباً إلى تسليم منتج مختلف عما يريده الزبون نتيجة لتغير متطلباته أو لعدم الفهم الدقيق لها منذ البداية. لذلك ولحل هذه المشكلة طُرِح مفهوم النماذج السريعة Agile القائمة على Agile Manifesto أو إعلان الأجيل في عام 2001 [2] الذي دمج الزبون بعملية التطوير، وظهرت العديد من المقاربات والمنهجيات التي استندت على هذا الإعلان، وساهمت بتحسين معدل إصدار البرمجيات.

استمر العمل على إيجاد منهجيات جديدة بهدف التحسين المستمر continuous improvement لعمليات التطوير البرمجي، فأدى ذلك إلى ظهور منهجيات التطوير الرشيقة Lean، المشتقة من المفاهيم والممارسات المعتمدة في المصانع اليابانية وخصوصاً تلك المطبقة في مصنع Toyota، وكان أولى بوادرها في عام 2001 [3].

واجهت غالبية المنهجيات وأطر العمل السابقة عند تطبيقها في بيئة العمل العديد من التحديات كان أبرزها عدم التوافق بين معدل إصدار البرمجيات العالي ومعدل نشرها، إضافة لظهور العديد من المشاكل عند نشر المنتج البرمجي من قبل فرق العمليات المسؤولة عن إعداد بيئة الإنتاج

Production Environment. لذلك السبب ظهرت منهجية DevOps، والتي كان إحدى أهدافها الرئيسية هو خلق بيئة تعاون بين الفرق وإجراء التحسين المستمر على عملية التطوير، وعرفت بناء على [4] "هي الاتحاد القائم بين الأفراد والعمليات والمنتجات لتوفير التسليم المستمر للقيمة إلى المستخدمين النهائيين. وبالتالي هي ليست منتج تبتاعه وتنصبه، كما أنها ليست الأتمتة والبنية كشيفرة مصدرية فقط، وإنما هي اتباع الأفراد لعملية مفعلة من قبل المنتجات لتأمين القيمة الحقيقية وإيصالها إلى الزبائن".

على الرغم من النجاح الذي أحدثته DevOps ضمن منظمات تطوير البرمجيات، إلا أن المحتوى الأكاديمي الخاص بهذه المنهجية لم يمتلك توضيحاً كافياً معمقاً عن الأسلوب المناسب لتبني هذه المنهجية.

1.1 مشكلة البحث

ظهرت DevOps في عالم هندسة البرمجيات واعتمدت بشكل رسمي منذ عام 2013 [4]، ولكن على الرغم من شهرتها وانتشارها وتبنيها بشكل كبير من قبل منظمات التطوير البرمجي، إلا أننا نجد أن هذه المنهجية محاطة بكثير من الغموض حيث لا يوجد تعريف رسمي لها، ولم توضع العديد من الأبحاث التي تناولت هذه المنهجية بدراسة معمقة لتحديد الأسس الخاصة بعملية تبنيها كمنهجية تطوير برمجية، بالإضافة إلى وجود خلط كبير بينها وبين المنهجيات التي سبقتها، حيث توجد العديد من الآراء التي تقول بأنها عبارة عن إحدى المنهجيات التي تقع تحت مظلة منهجيات الأجيل. الأمر الذي شكل عائقاً كبيراً في تبني هذه المنهجية وانتشارها.

2.1 أهداف البحث وأهميته

نهدف في هذا البحث إلى الوصول إلى تعريف خاص بمنهجية DevOps وآلية واضحة لتبنيها ودراسة التحسين الذي تحدثه في المنظمات التي تتبناها، وسنقوم بتحقيق ذلك من خلال مجموعة من الأهداف الفرعية:

1. دراسة معمقة عن منهجيات تطوير البرمجيات لمعرفة الفروقات والميزات ومجالات تطبيق كل منها، بدءاً من ظهور المنهجيات المقادة بخطة أو التقليدية plan-driven software development methodologies، مروراً بالمنهجيات السريعة Agile Methodologies والمنهجيات ال الرشيقة Lean، وصولاً لمنهجية الDevOps.

2. طرح إطار عمل DevOps مبني على المفاهيم المستخلصة من منهجيات وأطر العمل التي سبقت هذه المنهجية.

3. وضع دراسة حالة مستندة على إطار عمل DevOps لمعرفة ودراسة الأثر الذي أضافه على عملية التطوير البرمجي.

تكمن أهمية هذه الدراسة في كون أن هذه المنهجية هي التوجه الجديد للمنظمات البرمجية، حيث فرضت البيئات الجديدة للتطبيقات على المطورين ومنظمات التطوير البرمجي إيجاد آليات متوافقة لتنظيم العمل، تضمن التسليم السريع للبرمجيات والميزات الجديدة التي تفرضها متطلبات الزبائن المتغيرة، مع المحافظة على أمنها وجودتها.

3.1 مجال البحث

سنقوم ضمن هذا البحث برصد تطور منهجيات تطوير البرمجيات بدءاً من ظهورها، وصولاً إلى منهجية DevOps والتي ستكون محور دراستنا، ومعرفة الأسباب التي أدت إليها، بالإضافة إلى تطبيق المعرفة التي حصلنا عليها في إيجاد إطار عمل يمكن أن تتبناه أي منظمة برمجية لتطوير البرمجيات.

سيوجه إطار العمل المطروح للمنظمات التي تمتلك مجموعة من الفرق التقنية ذات الحجم المتوسط إلى الكبير التي تحتاج إلى العمل مع بعضها لتقديم منتج برمجي إلى الزبائن.

لن نتطرق في دراستنا إلى مجالات ضمان الجودة والأمن التي تعتبر حالياً من المواضيع الرائجة في عالم تطوير البرمجيات، وخصوصاً للباحثين المهتمين بمنهجية DevOps. حيث وجدنا أن التطرق إلى هذه المواضيع ضمن الدراسة الحالية سيقوم بإضافة تشعبات وأبعاد قد تبعد البحث عن هدفه الرئيسي وهو توضيح مفهوم هذه المنهجية الجديدة، وإيجاد تعريف خاص بها ليكون نقطة انطلاق للباحثين أو الممارسين المهتمين في التعمق في المواضيع المرتبطة بهذه المنهجية.

4.1 منهجية البحث المستخدمة

اعتمدنا في دراستنا على منهجية المراجعة البحثية الممنهجة Systematic Literature Review (SLR)، لاستخراج الجوانب النظرية الواجب دراستها في بحثنا، حيث تقوم هذه المنهجية على أساس تجميع المقالات العلمية بناء على استعلامات بحثية تتضمن الكلمات المفتاحية الخاصة بالبحث[5].

في حين اعتمدنا في الجانب التطبيقي للأطروحة على منهجية دراسة الحالة Case Study، والتي عادة ما تستخدم من أجل رصد ظاهرة معينة تحدث ضمن سياقها الأصلي[6]. حيث وجدنا أنها ستكون مناسبة في مراقبة النتائج التي ستحدث نتيجة تطبيق منهجية تطوير برمجيات – في دراستنا هي منهجية DevOps– ضمن منظمة برمجية تمتلك مجموعة من الفرق وتعمل على تطوير مشروع برمجي. واستخدمنا على وجه الخصوص المقاربة الخاصة بدراسة الحالات المتعددة، لأن الهدف من بحثنا هو معرفة الأثر والتحسين الذي سيحدثه تطبيق منهجية DevOps ضمن المنظمة مقارنة بغيرها، فلذلك كان لابد من إيجاد حالة أخرى نستطيع المقارنة بها[6].

لتحليل النتائج المستخلصة من دراستي الحالة قمنا باستخدام تحليل السمات Thematic Synthesis، والذي يقوم على تصنيف البيانات المستخلصة إلى سمات أو مجموعات، ومن ثم تطبيق التحليل المتقاطع Cross-Case Analysis لمعرفة نقاط التشابه والاختلاف بين الحالتين[7] .

5.1 مخطط الأطروحة

بدأنا الأطروحة بفصل المقدمة، والذي عرض النقاط الأساسية التي بنينا عليها هذا البحث من مشكلة وهدف، وحدد مجال الدراسة ومنهجيات البحث التي استخدمت.

قمنا بالحديث عن مفاهيم منهجيات هندسة البرمجيات في الفصل الثاني من الأطروحة، والحاجة التي كانت وراء ظهور كل نوع منها، مع استعراض شرح خاص بمجموعة من المنهجيات التي تندرج تحت مظلة كل نوع، واحتوى أيضاً على شرح معمق عن منهجية DevOps، حيث تحدثنا عن بدايات ظهورها، والأسباب التي دفعت مجتمع هندسة البرمجيات إلى إيجاد المفاهيم التي تنص عليها هذه المنهجية.

الفصل الثالث تَضَمَّنَ الدراسة المرجعية للأطروحة، حيث استعرض مجموعة من أهم الأبحاث التي اعتمدنا عليها لفهم مشكلة البحث، ووضع الأهداف المراد تحقيقها.

شرحنا في الفصل الرابع إطار عمل DevOps المقترح والأدوات التي قررنا استخدامها لتحقيق دراسة الحالة التي سندرس من خلالها أداء إطار عمل DevOps.

قمنا بصياغة دراسة حالتين في الفصل الخامس، تطرقت الأولى إلى تطوير تطبيق وب باستخدام منهجية تطوير سريعة، والأخرى إلى تطوير نظام مكتبة الكترونية تابعة لإحدى كليات جامعة البعث باستخدام إطار عمل DevOps المقترح.

نهاية، قمنا في الفصل السادس بمناقشة النتائج المستخلصة، وطرح مجموعة من المقترحات والرؤى المستقبلية للبحث.

الفصل الثّاني
الدّراسة النظريّة

مقدمة:

تعتبر عمليات تطوير البرمجيات ومحاولات تنظيمها وضبطها من أكثر الأمور الشائكة التي تواجه علم هندسة البرمجيات، لذلك ظهرت العديد من منهجيات التطوير الهادفة إلى ضبط هذه العمليات ووضعها ضمن إطار عمل يمكن رصده وتتبعه.

سنقوم في هذا الفصل بتوضيح المفاهيم الأساسية الخاصة بالأنواع المختلفة لمنهجيات تطوير البرمجيات بدءاً من ظهورها ووصولاً إلى منهجية DevOps، مع التطرق إلى مجموعة من المنهجيات وأطر العمل التي تندرج تحت كل نوع منها، وسننهي الفصل باستخلاص مجموعة من النتائج بناء على مقارنة بين هذه المنهجيات، حيث نهدف من ذلك إلى توضيح الأسباب التي قادت إلى الوصول إلى منهجية DevOps.

منهجية البحث المستخدمة:

اعتمدنا في هذا الفصل على منهجية المراجعة البحثية الممنهجة Systematic Literature Review (SLR)، والتي تقوم على أساس تجميع المقالات العلمية بناء على استعلامات بحثية تتضمن الكلمات المفتاحية الخاصة بالبحث. استطعنا من خلال هذه المنهجية تجميع ما يقارب 100 ورقة بحثية، تم اختزالها وتصنيفها بناء على التصنيفات الأربعة لمنهجيات تطوير البرمجيات، واستخراج أهم ميزات ومساوئ هذه المنهجيات وأسلوب العمل فيها، واستنتجنا مقارنة بناء عليها، كانت أهم أهدافها توضيح سياق استخدام كل منهجية من المنهجيات [5].

1.2 منهجيات تطوير البرمجيات Software Development Methodologies

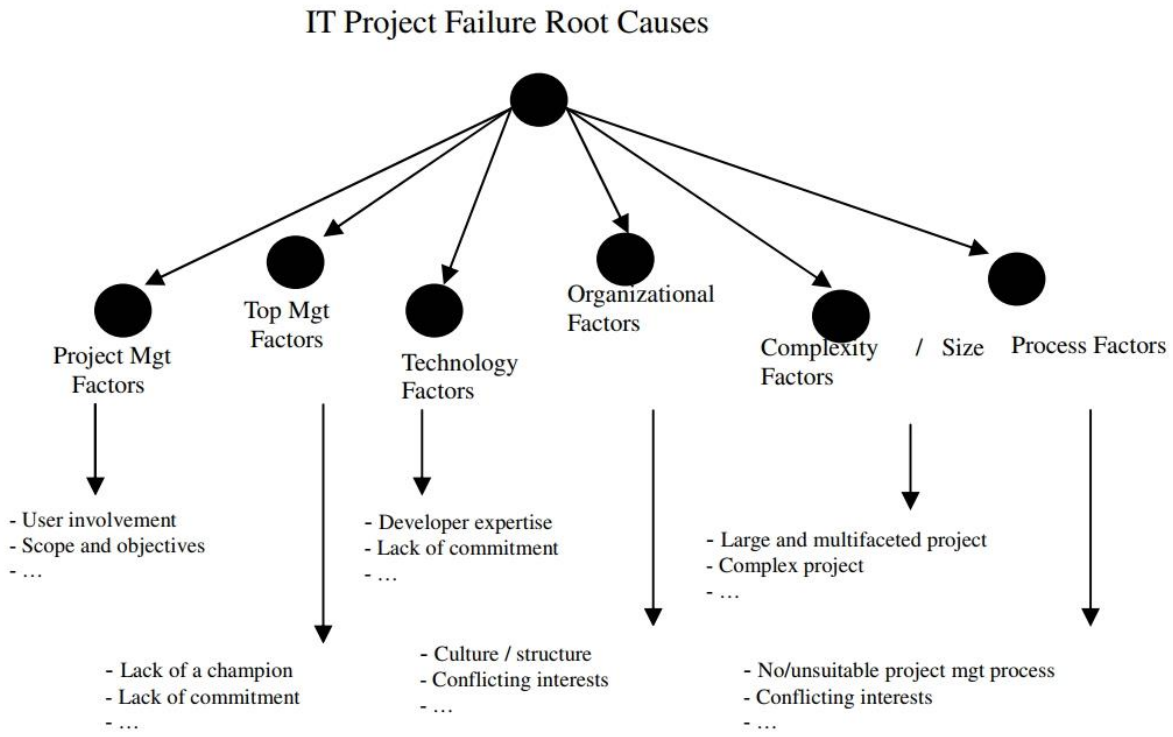
تُعرَّف منهجيات تطوير البرمجيات بأنها مجموعة من النشاطات المرتبطة مع بعضها التي تقود إلى إنتاج المنتج البرمجي، والتي تتضمن تطوير البرمجيات من الصفر باستخدام لغة برمجة معينة

أو بالتوسعة والتعديل على نظام موجود مسبقاً وذلك بتهيئة إعداداته ودمجها مع المكونات البرمجية الأخرى الخاصة بالنظام[1] .

- **أسباب ظهور منهجيات تطوير البرمجيات**
- **فشل المشاريع البرمجية**

تحدثت العديد من الدراسات عن أسباب فشل المشاريع البرمجية، ووجدنا أن غالبية الدراسات قد ركزت على مجموعة من الأسباب الرئيسية وهي[10][9][8] :

1. الإدارة الخاطئة لعملية التطوير، والتي عادة ما تكون نتيجة سوء فهم أسلوب العمل أو اختيار منهجية خاطئة.
2. عدم فهم متطلبات الزبائن، وعدم التزامهم في متابعة عملية التطوير.
3. سوء التواصل بين أفراد الفرق والزبائن.
4. المشكلات الشخصية بين أفراد الفريق الواحد، وبين الفرق المختلفة.
5. اختيار الأدوات والمعدات الغير مناسبة لعملية التطوير، وسوء استخدامها.



الشكل (2-1) مخطط هيكلي يظهر الأسباب الرئيسية لفشل المشاريع البرمجية [8]

وأظهرت تقارير مجموعة Standish Group التي تعنى بمتابعة التقدم الخاص بالمشاريع البرمجية ضمن المنظمات معدلات نجاح وفشل هذه المشاريع خلال مجموعة من السنوات وكانت كالتالي[10]:

السنة	المشاريع الناجحة	المشاريع التي واجهت تحديات	المشاريع الفاشلة
1994	16%	53%	31%
1996	27%	33%	40%
1998	26%	46%	28%
2000	28%	49%	23%
2004	29%	53%	18%

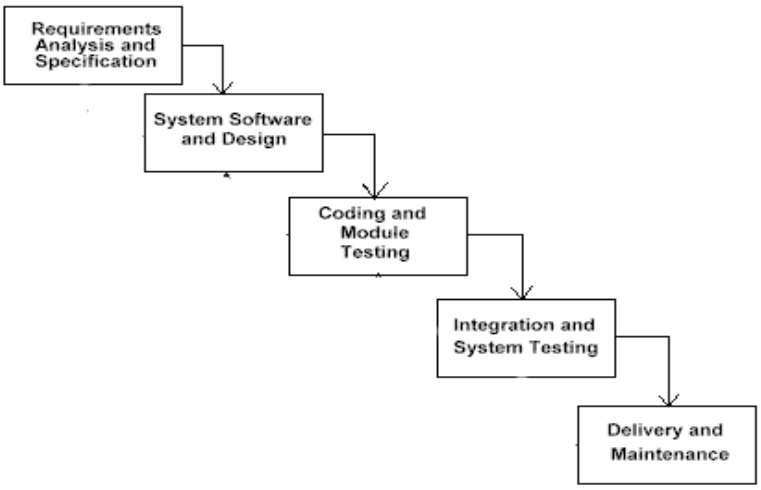
الجدول (2−1) معدلات المشاريع الناجحة والفاشلة والتي تواجه تحديات بالنسبة ل Standish Group

لذلك وبناء على الدراسات السابقة التي أكدت أن أحد أهم العوامل التي تسبب فشل المشاريع البرمجية هو الاختيار الخاطئ لمنهجية التطوير أو سوء عملية الإدارة، لذا كان من الضروري عرض شرح عن منهجيات تطوير البرمجيات، والسياق الخاص لاستخدامها ضمن عملية التطوير البرمجي.

1.1.2 نماذج التطوير التقليدية أو المقادة بخطة Plan-Driven Development Methodologies

هي النماذج التي تقوم على ما يعرف بدورة حياة تطوير البرمجيات SDLC، ويطلق عليها أيضاً النماذج المقادة بخطة وسميت كذلك لأن عملية التطوير تسير وفق خطة تقسم مراحل العمل إلى مجموعة مراحل تبدأ بمرحلة جمع المتطلبات مروراً بالتصميم والتحقيق والاختبار وصولاً إلى مرحلة النشر للمنتج البرمجي، محددة بقيود زمنية ومادية وعقود توقع من قبل الطرف المنجز للمشروع والزبون [1].

1.1.1.2 نموذج الشلال Waterfall Model



الشكل (2-2) نموذج الشلال

يعتبر نموذج الشلال [1] نموذج تطوير تسلسلي، يفرض على المطورين عدم الانتقال من مرحلة إلى أخرى قبل انتهاء المرحلة الحالية بشكل كامل، كما أنه لا يسمح بالعودة إلى مرحلة سابقة، ويكون تسلسل العمل وفقاً لذلك كالتالي:

- بداية، تبدأ مرحلة جمع المتطلبات وتحليلها واستخراج وثيقة المتطلبات Requirements ومن ثم يوضع توصيف النظام System Specification.
- لاحقاً تبدأ مرحلة التصميم، التي تنتهي بتحديد البنية المعمارية الكاملة للنظام.
- تبدأ مرحلة التحقيق انطلاقاً من بنية النظام المحددة في المرحلة السابقة، لتصار جميعها إلى البرمجية المحققة للنظام.
- تجمع مختلف المكونات وتدمج للحصول على البرنامج التنفيذي المطابق لتوصيف النظام، ويتم اختباره.
- نهاية تسلم البرمجية للزبون، ويدخل النظام في مرحلة التشغيل والصيانة.

- **ميزات هذا النموذج**

- لا تبدأ مرحلة التحقيق قبل الحصول على فهم دقيق للمشروع / البرمجية المطروحة من قبل الزبون ووضع تصاميم وتحديد البنية المعمارية المناسبة للمشروع، الأمر الذي يزيد من جودة البرمجية المقدمة، لوجود توثيق دقيق.
- تعتبر كل مرحلة محدودة بفترة زمنية للإنجاز، وينفذ العمل غالباً ضمنها.

- هي مقاربة تسلسلية لذلك عادة ما تكون سهلة التطبيق والإدارة.

- **سيئات هذا النموذج**

- عدم القدرة على تقبل طلبات التغيير من قبل الزبائن.
- تكلفة حدوث تغيير في المتطلبات أو اكتشاف أخطاء ناتجة عن مرحلة سابقة مرتفعة جداً، فمن الممكن أن تؤدي إلى إعادة تطوير المشروع من الصفر أو إيقافه.
- لا يسلم أي منتج للزبون قبل مرور فترة زمنية طويلة نسبياً.

2.1.1.2 النموذج الحلزوني Spiral Model

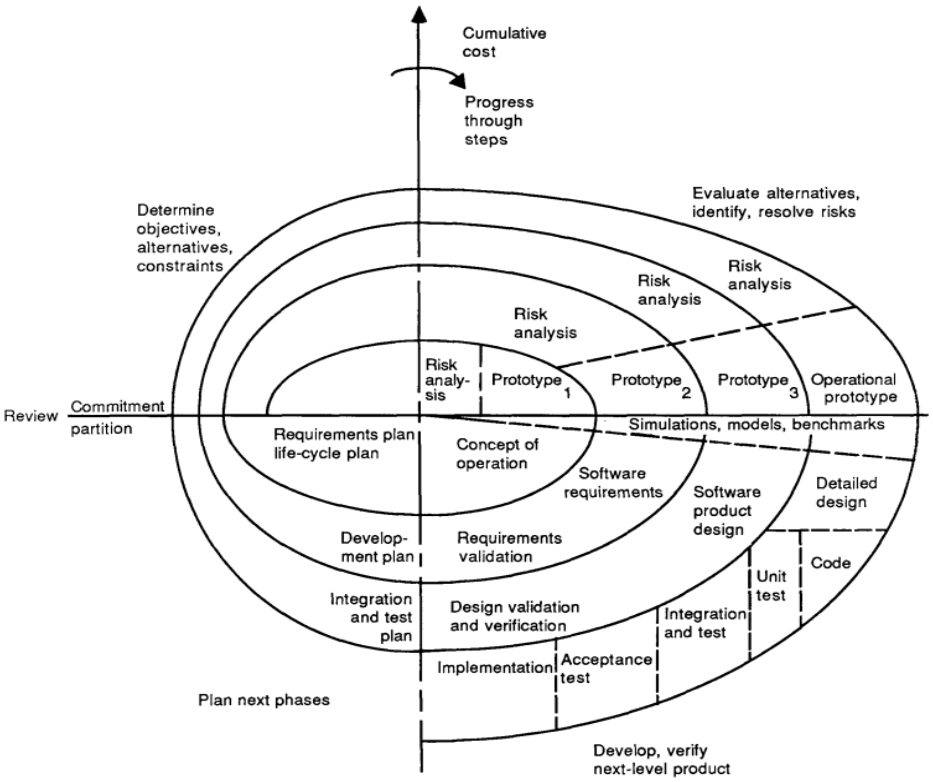
عُرِّف هذا النموذج من قبل باري بوهم Barry Boehm، في مقالته التي نشرت عام 1988[11]، حيث يقوم على ركيزة أساسية هي مرحلة تحليل المخاطر ويأخذ التطوير فيه مساراً يعرف باسم التطوير التكراري Iterative Development، يمتلك كل تكرار أربعة مراحل هي: التخطيط، تحليل المخاطر، الهندسة والتقييم. عادة ما تأخذ هذه التكرارات من ستة أشهر إلى سنتين حتى تكتمل، ويحدد هدف تصميمي لكل تكرار، وينتهي بمراجعة التقدم من قبل الزبون. تبذل الجهود في كل مرحلة من المشروع للوصول إلى الهدف الأخير وهو تسليم المنتج للزبون.

مع نهاية مرحلة تحليل المخاطر، نكون قد حصلنا على نموذج يمكن عرضه على الزبون للتقييم، ويستمر العمل بهذا الشكل حتى الانتهاء من المشروع بشكل كلي.

- **ميزات هذا النموذج**

- يدمج هذا النموذج مرحلة تحليل المخاطر في عملية التطوير منذ البداية.
- نموذج جيد لاستخدامه في المشاريع الحرجة.
- يتم الحصول على نموذج أولي للبرمجية في مراحل متقدمة من التطوير.

- **سيئات هذا النموذج**



الشكل (2-3) النموذج الحلزوني

- يعتبر من النماذج المكلفة، ويعود ذلك بسبب عمليات تحليل المخاطر والتخطيط لتفاديها، والتي أيضاً تحتاج إلى خبرات عالية للتطبيق.
- نجاح المشروع يعتمد بشكل كبير على مرحلة تحليل المخاطر.
- لا يعمل بنفس الجودة للمشاريع الصغيرة.

- **أبرز أدوار العمل التي تسند إلى العاملين وفقا للمنهجيات المقادة بخطة هي:** مدير المشروع، محلل النظم، المصمم، المطور، المختبر، مدير الإصدار.

2.1.2 تطوير البرمجيات السريع Agile Software Development

اعتُمِدَت منهجيات التطوير التقليدية، كأسلوب عمل لتطوير البرمجيات لفترة زمنية طويلة، بافتراضها المنهجيات الأفضل كونها مقادة بخطة واضحة ومتسلسلة. ولكن عند التطبيق كان من الصعب مجاراتها بسبب العديد من العقبات، كان أبرزها هو طول الفترة الزمنية للتنفيذ، وتَغَيُّر متطلبات الزبائن خلالها، إضافة لمشكلة صعوبة الحصول على المتطلبات بدقة وتحقيق برمجية تتوافق مع ما يريده الزبائن بالفعل.

لذلك ولحل هذه العقبات كان لابد من إيجاد آليات جديدة للعمل، فبدأ المبرمجون منذ تسعينيات القرن الماضي بإيجاد صيغ أكثر مرونة عرفت لاحقاً باسم الطرائق السريعة Agile Methods، وفي عام 2001 عُقد مؤتمر شارك فيه مجموعة كبيرة من المبرمجين القادة لحركة التطوير السريع وصدر عنه ما عرف باسم Agile Manifesto أو إعلان الأجيل، والذي اعتبر إعلاناً رسمياً من المطورين للإقرار بمجموعة المبادئ والقيم الأساسية التي تطرحها الأجيل[2].

- **إعلان الأجيل لتطوير البرمجيات أو Agile Manifesto**

وفقاً ل [2] حُددت أربع قيم أساسية للمنهجيات السريعة واثني عشر مبدئاً [12] تهدف إلى توصيل القيمة الحقيقة المرغوبة من قبل الزبون، ولا تعتبر أي منهجية بأنها منهجية سريعة في حال لم تتبعها.

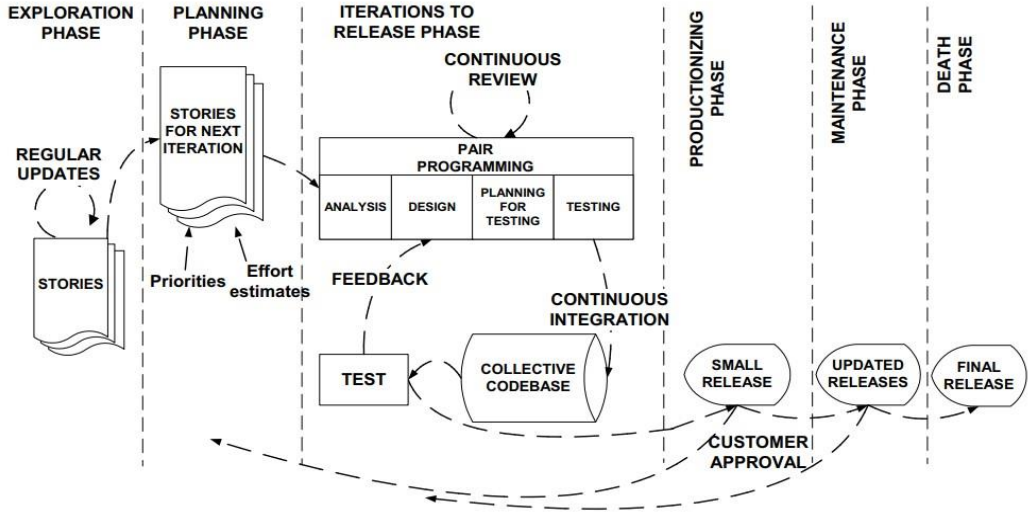
1.2.1.2 البرمجة المتطرفة eXtreme Programming XP

هي المقاربة الأكثر شهرة وانتشاراً في المنهجيات السريعة، حصلت على هذا الاسم من قبل Beck (1999) وسميت كذلك كون المقاربة قد طورت لتدفع الممارسات البرمجية الجيدة، كالتطوير التكراري، إلى مستويات متطرفة Extreme. على سبيل المثال، من الممكن تطوير العديد من الإصدارات الجديدة للنظام من قبل مبرمجين مختلفين، ومكاملتها واختبارها بشكل يومي[13].

هذه المقاربة بالطبع تتبع لمبادئ وقيم إعلان الأجيل، وتعمل وفقاً للتالي:

- تبدأ عملية التطوير بمرحلة الاكتشاف، والتي يتم فيها معرفة متطلبات الزبون وتدوينها كقصص مستخدمين، وتحديد التقنيات اللازمة لتطوير المشروع.
- من الممكن تصميم نموذج أولي يعرض على الزبائن للمراجعة والحصول على تغذية راجعة تفيد في التحقق من المتطلبات وغالباً ما تستمر هذه المرحلة بين عدة أسابيع إلى شهور قليلة.
- لاحقاً، وبعد تحديد المتطلبات ينتقل الفريق إلى وضع خطة عمل وتحديد قائمة المهام والأولويات والمخطط الزمني وفقاً لما يطلبه الزبون. يفضل تقليل الزمن الخاص بمرحلة التخطيط لعدم امتلاكها قيمة حقيقية للزبائن.
- يقسم العمل إلى تكرارات لا تتجاوز مدتها الأربعة أسابيع وفقاً للخطة الموضوعة في المرحلة السابقة. يتضمن كل تكرار خطوات التطوير والاختبار واجتماعات قصيرة لمتابعة سير العمل.

- ينتهي التكرار بمرحلة النشر لمنتج برمجي يحقق قصص المستخدمين المحددة لهذا التكرار، وينفذ اختبار القبول من قبل الزبائن، لمعرفة إن حقق المنتج رغبتهم أم لا، ويتم الحصول على تغذية راجعة للتحسين وإجراء التغيير اللازم.



الشكل (2-4) أسلوب العمل وفقاً لمنهجية البرمجة المتطرفة

تتبع منهجية البرمجة المتطرفة أو XP، إلى مجموعة من المبادئ الإضافية عن تلك الموجودة في إعلان الأجيل، ويمكن تلخيصها كما يلي [13]:

1- التخطيط التزايدي، والذي يعني أنه لا داعي لوضع خطة كاملة مفصلة منذ البدء وإنما يكفي تحديد أقسام المشروع والتكرارات اللازمة للانتهاء من كل قسم، وفي كل مرحلة يتم وضع خطة لها، بدلاً من إضاعة وقت كبير في عمليات التخطيط.
2- الإصدارات الصغيرة الهادفة إلى توصيل القيمة إلى الزبون من خلال إنتاج إصدارات تتضمن الوظائف الأكثر أهمية بالنسبة له بأسرع وقت ممكن، وبشكل تكراري وتزايدي.
3- التصميم البسيط، الهادف إلى إيجاد الحلول الأبسط للمشاكل التصميمية.
4- التطوير المقاد بالاختبار، والذي يعني البدء بعمليات كتابة الاختبارات قبل البدء بعملية التحقيق، للتأكد من أن عملية التطوير تسير وفقاً للمطلوب.
5- البرمجة المزدوجة، وتعني عمل المطورين على شكل أزواج لتطوير وظائف النظام المختلفة، بهدف تحقيق مراجعة للشيفرة المصدرية أثناء التطوير وضمان إمكانية الاستمرار في العمل في حال غياب أحدهم.

6- الملكية المشتركة، وتهدف إلى جعل جميع أفراد الفريق على دراية كاملة بعمل أحدهم الآخر، حتى تصبح الشيفرة المصدرية مسؤولية الفريق بدلاً من فرد واحد.

7- التكامل المستمر، وتهدف إلى تكامل أي وظيفة مطورة إلى الشيفرة المصدرية الأصلية فور جهوزيتها، لتختبر وتنشر.

8- مشاركة الزبون، حيث يصبح الزبون جزءاً من فريق التطوير، للتأكد من التطابق مع متطلباته لتحقيق ما يريده.

- **ميزات هذه المنهجية [17]**

- تسرع عملية تطوير المنتج البرمجي وتسليمه.
- تقلل التكلفة، وذلك بسبب سرعة تكرارات التطوير فيها، إضافة إلى مرونتها في إحداث التغيير مباشرة فور الحصول على رأي الزبون.
- تعزز التواصل بين أفراد الفريق والزبون بهدف تحسين جودة المنتج البرمجي من خلال الحصول على المتطلبات بدقة وفهمها.

- **مساوئ هذه المنهجية**

- تؤدي سرعة التطوير إلى الحصول على بنية أو تصميم سيء للبرمجية، الأمر الذي تحاول المنهجية تفاديه باعتماد مبدأ التصميم البسيط[18] .
- من الصعب تطبيق المنهجية في حال توزع الفرق جغرافياً لضرورة تواجد جميع المبرمجين في موقع واحد بسبب ممارسات البرمجة المزدوجة والملكية المشتركة.
- لا تفرض وجود توثيق متناسب مع عملية التطوير، أو عمليات التعديل الطارئة، لذلك تصبح عمليات تعقب الأخطاء مرهونة لمبدأ الملكية المشتركة للشيفرة المصدرية.

- **الأدوار الأساسية في منهجية التطوير هذه:** المدير، المستشار، المدرب، المتعقب، المختبر، الزبون، المبرمج.

2.2.1.2 منهجية السكرم Scrum Methodology

تعتبر إحدى أشهر منهجيات تطوير البرمجيات السريعة وأكثرها شيوعاً بين أوساط فرق التطوير البرمجية، بدأ الحديث عنها في مقالة نشرت في Harvard Business Review عام 1986 من قبل Hirotaka Takeuchi وIkujiro Nonaka وتحدثت عن عمليات تطوير المنتجات، لاحقاً استخدمها كين شويبر كأسلوب عمل تطوير البرمجيات ضمن شركته في تسعينيات

القرن الماضي [14]، واعتمدت كمنهجية تطوير في عام 2001 عندما قام كلاً من شويبر ومايك بيدل بوصف المنهجية في كتابهم Agile Software Development with Scrum.

- يتم العمل في هذه المنهجية وفقاً للتالي:

- بداية يعقد اجتماع بين أصحاب المصلحة المباشرين في عملية التطوير، وهم الزبون "مالك المنتج" Product Owner، وأعضاء فريق التطوير ومسؤول الفريق أو ما يدعى ب "Scrum Master" أو مسؤول السكرم، ويطرح الزبون متطلباته التي تعرف بقصص الزبون أو user stories، وتجمع ضمن سجل المنتج Product Backlog، وترتب بناء على أهميتها بالنسبة للزبون.
- بناء على سجل المنتج الذي يحصل عليه الفريق، يتم ترتيب اجتماع آخر يطلق عليه اسم اجتماع تخطيط المرحلة Sprint Planning Meeting، ويتم فيه اختيار أهم قصص الزبون من سجل المنتج والتي يمكن تنفيذها ضمن مدة زمنية تتراوح من أسبوع إلى أربعة أسابيع، للعمل عليها في ال sprint التالي وتحديد المخرج منه، لاحقاً يتم مناقشتها مع مالك المنتج ومعرفة تفاصيلها.
- فور الموافقة على مجموعة قصص المستخدمين الخاصة بالsprint الحالي، يباشر المطورون بالعمل، وخلال مدة العمل تعقد اجتماعات يومية تدعى sprint daily meetings، ليناقش الفريق ما قام به في اليوم السابق، وما هي المشاكل التي واجهته، وماذا سيفعل في اليوم التالي.
- ينتهي الفريق من الsprint عند الانتهاء من تطوير جميع قصص المستخدمين المحددة، ويعقد نتيجة لذلك اجتماع يعرف باجتماع مراجعة الsprint ، يدعى فيه جميع أصحاب المصلحة للحصول على تغذية راجعة عن العمل. لاحقاً يعقد اجتماع آخر خاص بالمطورين يدعى sprint retrospective يناقش آلية العمل في الsprint السابق وعما يمكن تحسينه أو الاستمرار بفعله.

- **ميزات هذه المنهجية[1]**

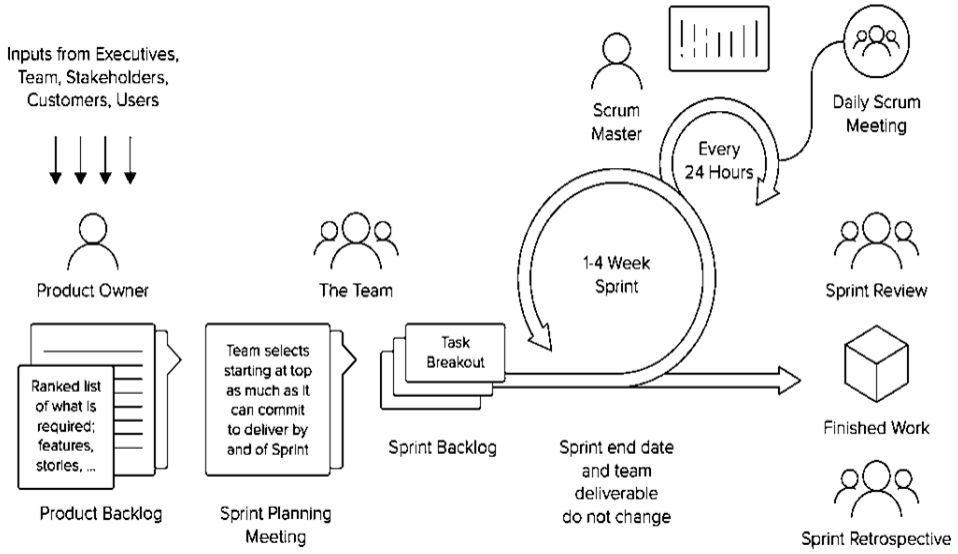
- يفيد تقسيم سجل المنتج إلى أقسام صغيرة تطويره على دفعات يمكن التعامل معها.
- يمكن البدء بتطوير المنتج قبل الحصول على توصيف دقيق لجميع متطلباته وبناء نموذج أولي، لتسريع عملية التطوير والحصول على مراجعة وتقييم الزبون وإجراء التعديلات بما يتوافق معها.

- تعزز التواصل بين أفراد الفريق والزبون، مما يجعل من عملية التطوير عملية مرئية من قبل الجميع، الأمر الذي يقوي الثقة بينهم ويزيد من فرص نجاح المشروع.

- **مساوئ هذه المنهجية [15]**

- غالباً ما تصبح عمليات التعديل والتغيير والإضافة صعبة على المبرمج لعدم فرض المنهجية وجود توثيق للبرمجية.
- تعتبر عمليات التواصل بين أفراد الفريق، والتواصل بين الفريق والزبون عاملاً مهماً وأساسياً لنجاح عملية التطوير وفقاً لمنهجية السكرم، ولكن في حال عدم إدارتها جيداً قد تسبب هدراً في الوقت وأحياناً فشلاً في عملية التطوير.
- من الصعب تطبيق هذه المنهجية في المنظمات الكبيرة الحجم، لمواجهتها للبيروقراطية التي تفرض من قبل إدارة المنظمات، إضافة إلى وجود تباين في مهارات الأفراد في المنظمات الكبيرة، والذي يظهر الأفراد ذوي المهارات الأقل كالحلقة الأضعف.

أدوار العمل في هذه المنهجية [15] : مسؤول السكرم Scrum Master، مالك المنتج Product Owner، فريق السكرم Scrum Team، الزبون Customer، الإدارة Managment.



الشكل (2-5) أسلوب العمل وفقاً ل Scrum

3.1.2 تطوير البرمجيات الرشيق Lean Software Development

بالتزامن مع ظهور إعلان الأجيل المحدد لمبادئ وقيم منهجيات التطوير السريعة، استمر العمل على إيجاد منهجيات تطوير أخرى قادرة على تنظيم العمل وإيصال البرمجيات المطورة

للزبون بأسرع وقت وأفضل جودة ممكنة، الأمر الذي دعا كلاً من الباحثين ماري وتوم بوبينديك [19] إلى إصدار كتاب "Lean Software Development" في عام 2003، الذي تحدث عن منهجيات جديدة تعتمد على مبادئ التصنيع اليابانية، وسعى إلى دمج هذه المبادئ مع منهجيات تطوير الأجيل، وكان هذا الكتاب نتيجة لأبحاث وتطبيقات كثيرة بدأت منذ تسعينيات القرن الماضي. دعيت هذه المنهجيات بمنهجيات التطوير الرشيقة أو Lean Software Development Methodologies.

منذ ظهورها وإلى الآن استمر الخلط بينها وبين منهجيات التطوير السريعة، حيث اعتبر طيف واسع من المطورين والباحثين منهجيات التطوير الرشيقة ما هي إلا إحدى مقاربات المنهجيات السريعة، في حين اعتبرها البعض بأنها مجموعة منهجيات مستقلة عن المنهجيات السريعة كون أن الأخيرة في مبادئها وممارستها وتطبيقها تركز على عملية التطوير وتحسينها، في حين تركز المنهجيات الرشيقة على عملية التطوير بشكل كامل بدءاً من ظهور الفكرة وصولاً إلى الإنتاج وتسليم المنتج [3]، إضافة إلى أنها تتبع لسبعة مبادئ مشتقة من نظام تويوتا للإنتاج Toyota Production System، لا تتعارض مع تلك الخاصة بالسريعة ولكن ليس بالضرورة أن تتواجدان معاً لاعتبار المنهجية بأنها من المنهجيات الرشيقة، هي [3]:

1- **إزالة الهدر**: يركز هذا المبدأ على إزالة أي مسبب لهدر الوقت بدءاً من لحظة الحصول على متطلبات الزبون، وصولاً إلى لحظة تحقيقها، ويمكن أن تكون المسببات هي ميزات إضافية غير مطلوبة، أو ميزات لا تحقق قيمة حقيقية للزبون.
2- **الجودة المضمنة Build Quality In**: تحقق باستخدام الممارسات الجيدة، كالتطوير المقاد بالاختبار، اختبارات القبول المؤتمتة، التكامل المستمر، وتحديد عدد مهام الاختبار الممكن تنفيذها بالتوازي، بهدف اكتشاف الثغرات والأخطاء قبل الوصول لمرحلة النشر.
3- **الحصول على المعرفة Create Knowledge**: يعتمد المطورون على مفهوم التعلم المستمر عند استخدام المنهجيات الرشيقة، حيث يقومون بوضع تصميم مبدئي بناء على متطلبات الزبائن للحصول على التغذية الراجعة للتحقق من صحتها، ويتم العمل وفقاً لحلقة التعلم التي تفيد في الحصول على معلومات أدق عما يريده الزبائن.
4- **تأجيل الالتزام Defer Commitment**: تعتبر عملية التخطيط مهمة جداً لنجاح أي مشروع، ولكن لا يجب اعتبارها عقداً لا يعدل عليه كون أن التغيير عملية محتمة الحدوث،

لذلك يفضل تأخير الالتزام واتخاذ القرارات حتى آخر لحظة ممكنة، لتبنى على أكبر قدر من المعلومات ويقل فيها نسبة التخمين.

5- **التوصيل السريع Deliver Fast**: إيجاد أفضل الآليات للحصول على تقدم سريع في العمل مع المحافظة على جودته.

6- **احترام الأفراد Respect People**: يجب منح الثقة للفرق لاتخاذ القرارات بخصوص أسلوب العمل، دون فرض خطة مقيدة لهم، حيث يكفي تحديد الأهداف والمخطط العام، وترك ما تبقى لهم.

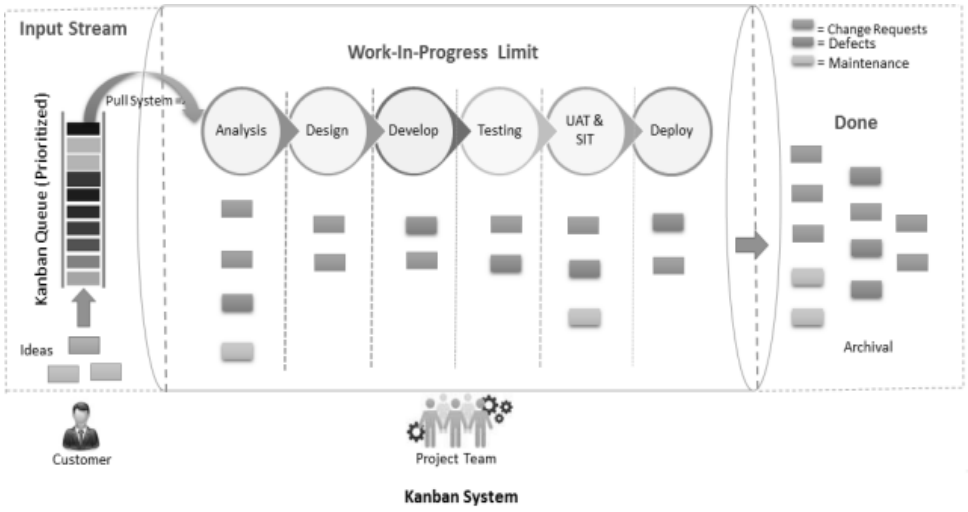
7- **التحسين الكامل Optimize the Whole**: عند البدء بالعمل على مشروع معين يجب تحديد هدف عام وحيد لجميع الفرق المسؤولة عن التطوير، يقسم لاحقاً إلى مجموعة أهداف جزئية منسجمة مع بعضها وتصب في تحقيق الهدف العام عند تجميعها.

1.3.1.2 الكان بان Kanban

تعتبر الكان بان إحدى أشهر المنهجيات الرشيقة، وتعني كلمة Kanban اليابانية لوح الإشارة أو signboard، واستخدم المصطلح في المصانع اليابانية، للإشارة إلى نظام جدولة يحدد متى وأين ومقدار الإنتاج[20] .

بدأت ملامح ظهور هذه المنهجية في عام 2010 من قبل دايفيد أندرسون، الذي دعي من قبل مايكروسوفت للمساعدة في جعل عملية التطوير مرئية قدر الإمكان واختبار ذلك ضمن فريق صغير الحجم بهدف إدارة تدفق العمل، وقد عززت هذه التجربة المساهمة الحقيقية التي يضيفها تبني الكان بان في التطوير [21].

تعرض هذه المنهجية تدفق العمل على شكل بطاقات مهام توضع على لوح مرئي للجميع، مركزة على الحد من العمل قيد الإنجاز Work in Progress WIP في كل مرحلة لتقليل مخاطر حدوث مشكلة عنق الزجاجة في تدفق العمل، إضافة إلى أنها تؤكد على ضرورة قياس زمن التكرار لتحسينه في كل مرة وإزالة الهدر منه.



الشكل (2-6) أسلوب العمل وفقاً ل Kanban

- **أسلوب العمل وفقاً للمنهجية**

- بداية، يجمع سجل المهام Tasks Backlog المعبر عن متطلبات الزبائن، ويقسم إلى مهام تكتب على بطاقات، وتلصق على لوح الكان بان.
- يقسم لوح الكان بان إلى مجموعة أعمدة كل عمود يعبر عن رتل خاص بكل مرحلة من مراحل التطوير، كعمود المهام، والتحقيق، والاختبار، والإصدار، وأخيراً النشر، وتحدد سعة كل رتل – أي عدد المهام الممكن إنجازها بشكل متوازي– بناء على عدد أعضاء الفريق وقدرتهم على التنسيق بينهم لإنجاز المهام.
- يقسم كل عمود إلى عمودين، الأول يعبر عن المهام قيد الإنجاز، والآخر المهام المنجزة، تجتاز كل بطاقة مهمة المراحل السابقة لتصبح ميزة منجزة منشورة للمستخدم.
- تنتقل البطاقة من رتل إلى آخر، في حال تم إنجازها بالكامل في الرتل الحالي، وكان باستطاعة الرتل التالي استيعاب المزيد من البطاقات، وإلا تبقى في رتلها.
- يتم مراجعة اللوح بشكل دائم، لضمان عدم حدوث أية مشاكل كعنق الزجاجة، ولتحديد النقاط التي يمكن تحسينها وتسريعها في خطوات لاحقة.

- **ميزات هذه المنهجية**

- تمنح الفريق الحرية في اختيار أسلوب إدارة المشروع، بهدف تعزيز دوره بدلاً من وجود قواعد كثيرة معيقة لعمله.
- تهدف العملية إلى الحصول على التدفق الأمثلي للعمل، من خلال تقليل الهدر الناتج عن العمليات غير المكتملة وتلك التي لا تضيف قيمة، إلى أقل حد ممكن.

- تفيد عملية عرض المهام على لوح البطاقات في تحسين العمل باستمرار، ومعرفة مواضع الخلل الناتجة عن التغيرات بسهولة أكبر.

- **سيئات هذه المنهجية**

- لا توثق البرمجية المطورة إلا في حال طلب الزبون وجود توثيق، لاعتبارها عملية لا تضيف قيمة، الأمر الذي يؤثر على جودة البرمجية.
- لا توجد أي قواعد تحدد آلية التواصل بين أفراد الفريق والزبائن، ويترك أمر إجراء الاجتماعات عند وجود الحاجة لذلك، الأمر الذي في حال لم يجر بشكل صحيح بإمكانه أن يؤدي إلى سوء في تحقيق البرمجية.

- **أدوار العمل وفقاً لهذه المنهجية:** يمكن للفريق أو المنظمة التي تعتمد منهجية الكان بان إسناد الأدوار التي تتناسب وتساعد في تحقيق مبادئ المنهجيات الرشيقة، وينبع ذلك من فكرة أن الفرق التي تتبع هذه المنهجية هي فرق منظمة ذاتياً.

4.1.2 منهجية DevOps

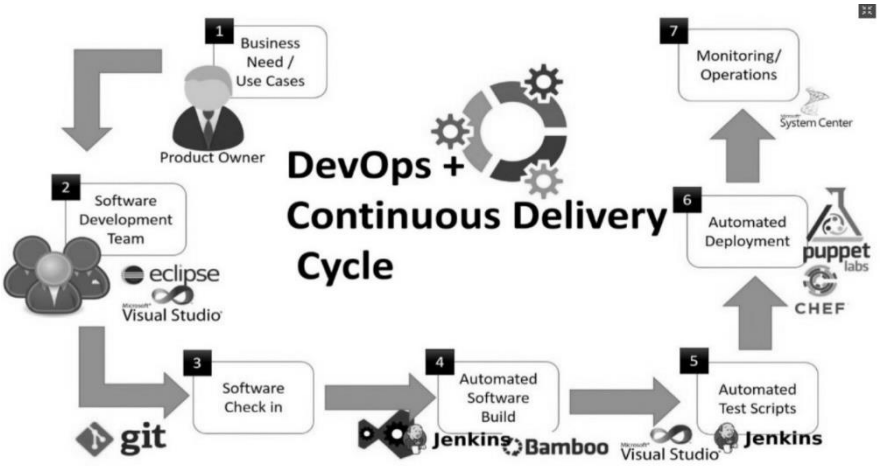
استطاعت كلاً من منهجيات التطوير السريعة Agile والرشيقة Lean حل الكثير من المشكلات التي واجهت مجتمع تطوير البرمجيات في ظل منهجيات التطوير التقليدية، والتي كان أبرزها مشكلة التغيير في متطلبات الزبائن وضرورة تسليم البرمجيات بسرعة.

ركزت المنهجيات السابقة على مفهوم تسليم القيمة الحقيقية للزبون بأسرع وقت ممكن، وبذلت فرق التطوير الجهود في سبيل ذلك، واستطاعت فعلاً إنتاج برمجيات بسرعة عالية مطابقة لمتطلبات الزبائن، وتمتلك المرونة لإجراء التعديلات، ولكن شكل هذا الأمر عبئاً على أطرافٍ أخرى معنية بعملية التطوير، كفرق العمليات، المسؤولة عن إعداد بيئة الإنتاج ونشر البرمجيات الجديدة وتحديثاتها، إضافة إلى عمليات الصيانة ومراقبة الأداء ومدى الاستقرار، واتبعت خططاً واضحة لتحقيق ذلك.

سَبَّب الاختلاف في معدل الإصدارات الناتج عن اتباع فرق التطوير للمنهجيات السريعة أو الرشيقة، وظهور بيئات عمل جديدة كبيئات السحابة والآلية الافتراضية Virtualization

والحاويات Containerization، إلى وضع فرق العمليات في مأزق عنق الزجاجة، فكان لا بد من إحداث تغيير في أسلوب عملها [22]. وكان ذلك من خلال طرح مفهوم الDevOps ، وهو

مصطلح مكون من اختصار لكلمتي Development وتعني التطوير وال Operations وتعني العمليات.



الشكل (2-7) أسلوب العمل وفقاً ل DevOps

جاءت DevOps كتطور طبيعي لمنهجيات تطوير البرمجيات، لتهدم الحواجز الموضوعة بين الفرق المختلفة الموجودة في منظمات التطوير البرمجية، وهي فرق التطوير والجودة والأمن والعمليات، وقامت على مجموعة من الفلسفات الثقافية والمفاهيم، والممارسات والأدوات التي تهدف إلى زيادة السرعة في توصيل التطبيقات للزبائن وهي [24][25]:

1- **التواصل والتعاون:** وتعتبر من أهم المفاهيم التي تقوم عليها المنهجية، حيث تؤكد ضرورة عمل الفرق المختلفة معاً وتوحيد الأهداف والأدوات المستخدمة قدر الإمكان، بهدف إيجاد بيئة عمل تصل ضمنها جميع المعلومات الضرورية إلى جميع الأطراف المعنية بسرعة، باستخدام الأدوات والأساليب التي يتم الاتفاق عليها. وجب الذكر أن هذا المفهوم ليس محدوداً بحدود المنظمة التي تتبنى DevOps وفرقها المختلفة وإنما يشمل أيضاً جميع أصحاب المصلحة الذين هم على علاقة بالمشروع، وإن كانوا خارج المنظمة كالزبائن بالمرتبة الأولى.
2- **التخطيط المستمر:** لا تتعارض ممارسات منهجية DevOps مع عملية التخطيط، ولكنها تفرض ضرورة جعل الخطط الموضوعة مرنة وقابلة للتعديل بما يتوافق مع متطلبات الزبائن، حيث تضع مجموعة نقاط تحقق checkpoints، عند الوصول إليها تطلب التغذية الراجعة من الزبائن وتعدل الخطة بناء عليها.
3- **التكامل المستمر:** ويشير هذا المفهوم إلى ضرورة إجراء عمليات دمج الميزات المطورة الجديدة فور الانتهاء منها، للتحقق من صحتها، ولا يقتصر هذا المفهوم على مرحلة تكامل

الشيفرة المصدرية الجديدة مع النظام الكامل، وإنما تتعدى ذلك لتطرح ضرورة دمج المكونات الجديدة وإيصالها إلى مرحلة النشر، للتأكد من أنها تعمل بشكل صحيح.

4- **النشر المستمر:** وهو المفهوم الأهم في DevOps، حيث تنصح الدراسات والإحصائيات على تهيئة عمليات نشر مؤتمتة للميزات الجديدة المطورة، إضافة إلى تحويل البنية التحتية لبيئة النشر إلى شيفرة مصدرية، تُحَدّث باستمرار وتنشر أيضاً بشكل مؤتمت، وذلك للتخلص من عمليات إعداد البيئات اليدوية التي تأخذ وقتاً طويلاً.

5- **التسليم المستمر:** تعتبر إحدى مقاربات تطوير البرمجيات، والتي تهدف إلى التسليم المستمر للقيمة إلى الزبائن عبر دورات تطوير قصيرة والتأكد من قابلية البرمجية للإصدار في أي وقت [26]. وتعتبر هذه المقاربة إحدى عوامل نجاح هذه المنهجية.

6- **مسارات العمل المؤتمتة Automated pipelines**: ونعني بمسار العمل مراحل دورة حياة البرمجية، وتهدف ال DevOps إلى أتمتة كل الأعمال الممكنة فيه، بهدف زيادة سرعة العمل وتقليل النفقات الناجمة عن الأعمال اليدوية والأخطاء البشرية الممكن حدوثها.

7- **المراقبة المستمرة:** يُراقب أداء التطبيقات المنشورة ومدى توافقها مع البيئة، وتستخلص آراء الزبائن والمستخدمين بأسرع وقت ممكن، لإحداث التعديلات الضرورية والمطلوبة.

8- **ضمان الجودة[27]:** بهدف التأكد من التحسين المستمر لعملية التطوير، كالقيام بالاختبارات المؤتمتة واليدوية التي تتوافق مع سلوك المستخدمين، والتصحيح الفوري للأخطاء عند ظهورها، إضافة إلى دمج هذه الممارسات منذ البداية للمساعدة في الحصول على معلومات أكثر عن المنتج لزيادة القدرة على وضع نماذج اختبار أفضل[28].

9- **الفشل والتعافي بسرعة Fail Fast & Recover Fast [29]:** تطرح منهجية DevOps مفهوم التطوير والتكامل والنشر السريع للبرمجيات بهدف تقليل الزمن المطلوب لاختبار البرمجيات من قبل الزبائن لمعرفة نسبة الأخطاء الكامنة فيها ومدى استقرارها، لتصحيح مسار العمل والتعافي بسرعة من هذه المشاكل.

- **آلية العمل وفقاً لهذه المنهجية**

يمر تطوير التطبيق ضمن مسار عمل وفقاً للمراحل التالية [30]:

- بداية، يحدد مجال وسياق ومتطلبات المشروع، وغالباً ما ينصح استخدام مبادئ وممارسات إحدى المنهجيات السريعة Agile للقيام بهذه الخطوة [36]. تطور البرمجية بما يتوافق مع المنهجية المختارة.

- التثبيت: فور الانتهاء من أي خطوة في التطوير من قبل أي مبرمج، يقوم بإجراء تثبيت Commit للشيفرة المصدرية التي طورها إلى نظام التحكم بالإصدار Version Control System. عندها يُقدَح مخدم التكامل المستمر، الذي يبني الإصدار الجديد بعد الدمج، في حال فشلت العملية يُنَبه المطور ولا يتم استكمال عملية التثبيت.
- الاختبار المؤتمت: عند نجاح عملية التثبيت، ينشر الإصدار الجديد إلى بيئة الاختبار وتنفذ الاختبارات المؤتمتة التي تم بناؤها مسبقاً ضمن بيئة الاختبار المعدة مسبقاً بشكل أوتوماتيكي، في حال فشل الاختبار لا ينتقل الإصدار للمرحلة التالية، وتصحح المشاكل التي ظهرت.
- الاختبار الاستطلاعي Exploratory Testing: ينتقل التطبيق في حال نجاحه بمرحلة الاختبار المؤتمت إلى مرحلة الاختبار الاستطلاعي، بهدف إجراء اختبارات يدوية، على بيئة معدة أوتوماتيكياً أيضاً. وفي حال عدم النجاح بهذه المرحلة لا ينشر التطبيق للمستخدمين.
- الإنتاج Production: ينشر الإصدار الجديد بعد اجتيازه لمرحلة الاختبار الاستطلاعي بنجاح في بيئة الإنتاج النهائية. تراقب البيئة لمتابعة أداء الإصدار المنشور، في حال ظهور أي خطأ أو مشكلة يتم التراجع إلى الإصدار الأسبق مباشرة. تنفذ هذه العملية وفقاً لاستراتيجية النشر.

- **ميزات هذه المنهجية**

- يزيد اتباع منهجية DevOps من سرعة تطوير البرمجيات، كنتيجة للتواصل المنتظم بين أفراد التطوير والعمليات[28] .
- تفعل المنهجية دور ضمان الجودة في التطوير، من خلال دمج الاختبار في كل مرحلة وتقصير زمن التوصيل للمستخدم ليتمكن فريق التطوير من معرفة المشكلات بسرعة وإصلاحها من خلال استخدام أدوات المراقبة المستمرة [29].
- تفيد التغذية الراجعة المستمرة من المستخدمين النهائيين إلى تحسين مسار العمل، وتوصيل برمجيات ذات جودة، إضافة إلى دورها الكبير في اتخاذ القرارات.

- **سيئات هذه المنهجية**

- لا تلائم هذه المنهجية جميع أنواع التطبيقات أو البيئات، كالبيئات التي تنصب فيها البرمجيات وتعمل على أجهزة المستخدمين، أو لتعديل الأنظمة الموروثة، حيث يعتبر من غير المجدي القيام بتحويل أسلوب التطوير فيها إلى DevOps [31].
- في حال طرحت منهجية الDevOps في بيئة عمل لم تتبنى ممارسات Agile، سيكون انتهاجها مكلفاً وصعباً للغاية على فرق التطوير.

- يصعب انتهاجها من قبل فرق العمليات، بسبب انهماكها بجداول الأعمال المزدحمة، فطرح هذه المنهجية من دون وجود تدريب أو استراتيجية صحيحة للتبني قد يسبب زيادة أعباء على فريق العمليات [29].

- **أدوار العمل في المنهجية:** مسؤول النظام System Administrator، فرق التطوير، وفرق العمليات، مهندس DevOps، مهندس الأمن Security Engineer، المختبر، مسؤول الجودة، مسؤول قاعدة البيانات.

2.2 مقارنة بين منهجيات تطوير البرمجيات المدروسة

الجدول (2-2) مقارنة بين منهجيات التطوير المختلفة

من حيث	المنهجيات التقليدية (المقادة بخطة)	المنهجيات السريعة	المنهجيات الرشيقة	DevOps
المنشأ	ظهرت في بداية سبعينيات القرن الماضي.	اعتمدت كمنهجيات تطوير في عام 2001 بعد إطلاق إعلان الأجيل والذي وضع أسس ومبادئ التطوير السريع، وتشكل على أثره ما يعرف باسم حلف الأجيل أو Agile Alliance والذي يشمل مجموعة المطورين الرواد في التطوير السريع.	بدأت الأبحاث لدمج مفاهيم وأسلوب عمل المصانع اليابانية مع مفاهيم وأسلوب عمل تطوير البرمجيات منذ تسعينيات القرن الماضي، وعلى أثرها نشر الباحثان Poppendick كتاب تطوير البرمجيات الرشيق عام 2003.	ظهرت بعد القيام بالعديد من المؤتمرات أولها كان مؤتمر O' Reilly Velocity عام 2009، وصولاً إلى المؤتمر الذي طرح اسم المنهجية لأول مرة وهو DevOpsDays في عام 2010، ووضعت أسس المنهجية فعلياً في عام 2013 في كتاب The Phoenix Project.
طبيعة البرمجيات المطورة	البرمجيات ذات التوصيف والمتطلبات الواضحة، والأنظمة الحرجة التي تتطلب خطة واضحة مفصلة لآلية العمل.	البرمجيات التي يمتلك الزبون فكرة مجردة أولية عنها ولكن لا يستطيع تحديد ما يريد بشكل كامل.	المشاريع البرمجية ذات دورة تطوير قصيرة نسبياً بحجم فريق صغير.	تحقق نجاحاً كبيراً عند استخدامها في التطبيقات المنشورة على السحابة. لا تعتبر مناسبة لاستخدامها في تطوير البرمجيات التي تنصب وتعمل على أجهزة الزبائن.
تدفق العمل	يجري العمل بشكل متسلسل في أغلب المنهجيات التي تتبع لهذا النوع، تبدأ من مرحلة جمع المتطلبات وصولاً إلى مرحلة الاختبار والتسليم، ولا يوجد تداخل بين المراحل، ومن الصعب العودة إلى مرحلة سابقة في حال اكتشاف أي خطأ أو مشكلة فيها.	يخضع تدفق العمل إلى الأسلوب التكراري التزايدي، والذي يهدف إلى زيادة الميزات المطورة مع كل تكرار، وصولاً إلى المنتج النهائي. لا يوجد أي عائق يمنع العودة إلى أي مرحلة سابقة في حال ظهرت مشاكل بسبب خطأ فيها أو طلب للتعديل.	يمكن أن تتبع ما يعرف باسم حلقة التعلم، تبدأ بعملية تخمين لما يريده الزبائن وبناء نموذج أولي، ومن ثم عرضه عليهم لمعرفة آرائهم، ومن ثم الدخول في حلقة جديدة تشبه سابقتها ولكن بمعلومات أكثر ورؤية أعمق لمتطلبات الزبائن.	يمكن اعتماد أي من أسلوبي العمل السريع أو الرشيق في مرحلة التطوير البرمجي. لاحقاً تمر البرمجية المطورة بتكرارات تزايدية، تكون فيها خطوات العمل متسلسلة، تبدأ ببناء التطبيق ومن ثم إجراء التكامل المستمر، والاختبار المستمر، والنشر المستمر، لينتهي التكرار بمرحلة المراقبة المستمرة، لتعود وتدخل البرمجية في تكرار جديد.

الجدول (2-2) مقارنة بين منهجيات التطوير المختلفة

من حيث	المنهجيات التقليدية (المقادة بخطة)	المنهجيات السريعة	المنهجيات الرشيقة	DevOps
التبني في بيئة العمل	غالبية المنهجيات التي تتبع لهذا النمط، هي منهجيات سهلة التبني والتطبيق، لأن خطواتها محددة ومعروفة ولا التباس فيها لأنها تتبع خطة محددة. يعتبر النموذج الحلزوني Spiral Model أصعبها من ناحية التبني، لأنه يتطلب وجود خبرات قادرة على تدريب الكادر على هذه المنهجية ومتابعة العمل لأنها تتعامل مع مشاريع حرجة تتطلب تحليلاً للمخاطر الممكن مواجهتها.	تتطلب تغييراً ثقافياً وفكرياً في بيئة العمل في حال كانت المنهجية المتبعة هي إحدى المنهجيات التقليدية، حتى يتأقلم الأفراد مع مستوى التواصل العالي الذي تتطلبه هذه المنهجيات. كما تتطلب تدريباً على مجموعة الممارسات الجيدة التي تقترحها، كالبرمجة المزدوجة، والتطوير المقاد بالاختبار وغيرها.	لا تعتبر عملية تبني هذا النمط من المنهجيات صعباً لأنها تركز على تحسين أسلوب العمل وإزالة الهدر، أكثر من تركيزها على ممارسات معينة. إضافة إلى أنها تقترح تعزيز دور الفريق وإعطاءه الثقة في تنظيم نفسه واتخاذ القرارات.	تحث الكثير من الدراسات على وجوب اعتماد نمط التطوير السريع والرشيق وفهم قيمه ومبادئه قبل تبني منهجية الDevOps، لأنه في حال العكس ستواجه المنظمة صعوبة كبيرة في تبني أسلوب العمل وفقاً لها.
الهدف	تحديد القيود الزمنية والكلفة للمشاريع البرمجية وتنظيم عمل الفرق البرمجية وفقاً لخطة توضع بناء على المتطلبات والقيود التي يضعها الزبون، حتى تسهل عملية	حل مشكلة التأخر في التسليم، أو تسليم برمجيات لا تتوافق مع متطلبات الزبائن وذلك بإشراكهم في العمل بشكل دائم.	تحسين عملية تطوير البرمجيات بشكل كامل وتسليم القيمة الحقيقية للزبائن بأعلى جودة وبأسرع وقت.	سد الفجوة بين فرق التطوير والعمليات في المنظمة بهدف إيصال البرمجيات ذات القيمة الحقيقية للزبائن، بسرعة وبجودة عالية.
التواصل	لا تضع أي قيود أو تطرح ضرورة التواصل والتعاون بين الفرق المختلفة كفرق التطوير والاختبار وضمان الجودة والعمليات، حيث تعتمد بشكل أساسي على توصيف المتطلبات الذي يتم الحصول عليه بالمرحلة الأولى.	إحدى المبادئ الأساسية التي تقوم عليها هذه المنهجيات، حيث تعتبر عاملاً أساسياً لنجاح التطوير وفقاً لها، وتتطلب استمراريته في كل مراحل العمل بين فرق التطوير والزبون.	تعتمد على التواصل مع الزبائن للحصول على معرفة أعمق عن البرمجية المطورة، وتضع مفهوم حلقة التعلم، حيث يطرح المطورون نموذجاً أولياً يراجعه الزبائن ويعدل مسار العمل وفقاً لآرائهم.	ظهرت المنهجية لسد الفجوة في التواصل والتعاون بين فرق التطوير والعمليات، دون أن تقلل من أهمية التواصل مع الزبائن بشكل مستمر لتحسين عملية التطوير.

خاتمة:

استطعنا في هذا الفصل الحصول على فهم معمق لمنهجيات تطوير البرمجيات واستنتجنا الفروقات ومجال استخدام كل منهجية، بدءاً من ظهورها وصولاً إلى منهجية DevOps والتي هي محور دراستنا، حيث كان من الضروري رصد التغيرات والأسباب التي قادت إلى ظهور كل منهجية لنستطيع معرفة العقبات الواجب تجنبها والميزات التي ستحسن من عملية التطوير البرمجي عندما نقوم بطرح إطار عمل خاص بمنهجية DevOps مستند على كل هذه المفاهيم.

الفصل الثّالث

الدّراسة المرجعيّة

مقدمة:

سنقوم في هذا الفصل بعرض مجموعة من الدراسات المرجعية التي تطرقت إلى آليات تبني منهجيات تطوير البرمجيات ضمن المنظمات البرمجية التي استندنا عليها ضمن هذا البحث والقيام بتحليلها مع التعمق والتركيز في الدراسات التي تناولت الحديث عن DevOps وأسلوب تبنيها، لرصد الأثر الذي أضافته هذه المنهجية، ولمعرفة التوجه الذي سيقوم عليه بحثنا وما الذي سنقوم بإضافته إلى مفاهيم الDevOps .

1.3 منهجية البحث

اعتمدنا في هذا الفصل على منهجية البحث المستخدمة في الفصل الثاني والتي كانت منهجية المراجعة البحثية الممنهجة (SLR)، والتي تقوم على أساس تجميع المقالات العلمية بناء على استعلامات بحثية تتضمن الكلمات المفتاحية الخاصة بالبحث[5].

2.3 السياق

قمنا في عملية استخراج الأبحاث والدراسات السابقة بالاعتماد على استخدام أنواع منهجيات تطوير البرمجيات ككلمات مفتاحية ضمن البحث، واستطعنا أن نقوم بتصنيف الأبحاث التي حصلنا عليها بناء على هذه الكلمات المفتاحية، وكان الهدف من ذلك فهم الأبحاث السابقة وكيفية تطرقها إلى دراسة منهجيات تطوير البرمجيات ضمن الجانبين الأكاديمي والتطبيقي، ومعرفة القصور التي وجدته هذه الأبحاث في كل نوع من المنهجيات بهدف إيجاد الحل له ومعرفة مدى إمكانية تجنبه أو إيجاد بدائل أفضل منه ضمن منهجية DevOps والاعتماد على ما تم دراسته للوصول إلى إطار عمل كامل لهذه المنهجية يمكن أن تتبناه المنظمات البرمجية في عمليات إدارة وتطوير مشاريعها، وأنهينا عملية الدراسة المرجعية هذه بتحليل عميق للمشاهدات والنتائج التي استطعنا التوصل إليها.

3.3 الدراسات المرجعية

ظهرت العديد من المنهجيات التي تناولت عملية تطوير البرمجيات، وتشابهت هذه المنهجيات في العديد من النقاط كالاتفاق على تسلسل تطوير البرمجيات بدءاً من الحصول على متطلبات الزبون وصولاً للتسليم، واختلفت بنقاط أخرى كآلية التعامل مع كل مرحلة من مراحل التطوير البرمجي. أدى ذلك إلى حدوث خلط وأخطاء في اختيار المنهجية المناسبة نتيجة لعدم الفهم الدقيق للمنهجيات، الأمر الذي بدوره أثر سلباً على فهم المنهجيات الجديدة كمنهجية DevOps، لأنها لم تضع أساساً واضحاً لها أو للمنهجيات المشتقة منها. شكلت هذه المشكلات عائقا كبيراً أمام المنظمات لتبني منهجية DevOps ومقارباتها، لذلك قمنا بدراسة مرجعية مستخلصة من مجموعة كبيرة من المقالات والأبحاث، وكان معيار الاختيار والتمايز بينها بالنسبة لدراستنا هو استخدام الدراسات التي طرحت مقارنات بين المنهجيات، وقامت بتوضيح الفروقات بينها، بالإضافة إلى ذكر دراسات حالة – ارتكزت بشكل أساسي على دراسات حالة لDevOps ومقارباتها– لاختبار المنهجيات ودراسة أدائها.

1.3.3 الدراسات المرجعية ما قبل DevOps

تحدث كلاً من Kalpana Sureshchandra وJagadish Shrinivasavadhani [32] عن الحاجة الملحة لتبني المنهجيات السريعة والرشيقة بدلاً من المنهجيات التقليدية واستندا في ذلك على دراسة حالة طُوِّر فيها مشروع برمجي على عدة مراحل، في المرحلة الأولى انتهجوا منهجية الشلال، ورصدوا التحديات والعقبات كالتأخير في التسليم، والأخطاء التي ظهرت عند إجراء اختبار القبول من قبل المستخدمين. لاحقاً قاموا بتدريب فريق العمل على مفاهيم المنهجيات السريعة، ودعوهم يختاروا ما يناسبهم من ممارسات بهدف حل العقبات التي ظهرت في مرحلة الشلال. استنتجا أنه من الواجب وجود تدريب للفريق حتى يكتسبوا مهارة التطوير السريع إضافة لضرورة مراقبتهم حتى بعد بدء العمل وفق المنهجية لضمان صحة سير عملهم. أخيراً نصح الباحثان على انتهاج المنهجيات السريعة بشكل متدرج بهدف التأكد من اتقان الممارسات السريعة كالتطوير المقاد بالاختبار والبرمجة المزدوجة.

وجد كلٌ من Pekka Abrahamsson وآخرون [33] أن المنهجيات التقليدية المقادة بخطة لا تنفذ بدقة في السياق العملي للتطوير لأنه من الصعب اتباع الخطة الموضوعة فيها، ولذلك ظهرت المنهجيات السريعة التي تتوافق بشكل أكبر مع أسلوب عمل الفرق، وطرحوا بعضاً من المعايير التي يجب تواجدها في المنهجية حتى تعتبر سريعة، كالتزايدية، والتعاون بين أصحاب المصلحة، إضافة إلى صفة البساطة والتوافقية. شملت الدراسة شرحاً لمجموعة من المنهجيات السريعة وقارنت بينها وعرضت مجال تطبيق كل منها وأسلوب عملها إضافة إلى الأدوار الخاصة بكل منهجية، لكن لاحظنا أن الدراسة لم تقم بتفضيل منهجية عن أخرى وإنما حاولت فقط إبراز نقاط القوة الخاصة بكل واحدة.

لم تتوقف الدراسات والمقارنات بين المنهجيات المختلفة بل على العكس ازدادت واتسعت لتشمل المقاربات الجديدة فطرح Laurie Williams ورقة بحثية [17]، استهلها بشرح عن المنهجيات السريعة والمنهجيات الرشيقة، وأجرى مقارنة بسيطة بينهم معتمداً على ما طرحه Poppendiecks بإمكانية استخدام المنهجيات الرشيقة لتفسير المنهجيات السريعة ولماذا الأخيرة هي أكثر قابلية للتطبيق في عمليات التطوير من غيرها. شملت المقارنة المفاهيم التي تتوافق فيها المنهجيات كمفهوم إزالة الهدر والتوصيل السريع والمستمر، وتحفيز الأفراد، والبساطة وغيرها، ولكن لاحظنا أن الدراسة لم تتطرق إلى مقارنة ما هو أعمق من ذلك كالفوارق الأساسية بين المنهجيتين ومجال تطبيق كل منها.

نشر كلٌ من Oisin Cawley وآخرون [34]، دراسة عن استخدام المنهجيات السريعة في تطوير التطبيقات الحرجة المضمنة، ووجدوا أن عملية تبني هذه المنهجيات في هذا النوع من التطبيقات غير مطروق بشكل كاف وأرجعوا السبب في ذلك إلى كون أن الشركات غالباً ما تحجم عن نشر ممارساتها الداخلية للعامة، إضافة إلى أنه لا يمكن اعتبار ممارسات المنهجيات السريعة متوافقة بالكامل مع نمط هذه التطبيقات، لذلك وجب دمج ممارسات المنهجيات الرشيقة Lean معها، واعتبروا ممارسات المنهجيات السريعة ما هي إلا دعامة لفلسفة المنهجيات الرشيقة في التطوير، كما أشاروا إلى ضرورة إجراء أبحاث أكبر عن هذه المنهجيات تستعرض السياسات ودورة حياة المنتج الواجب اتباعها. فنرى أن دراستهم كانت محاولة لإلقاء الضوء على مجال جديد يمكن تطبيق المنهجيات السريعة فيه، بالإضافة إلى أنها كانت بذرة لطرح دراسات أخرى تقوم على دمج المنهجيات السريعة والرشيقة.

نشرت الدراسة السابقة في عام 2010، واستمر كل من Oisin Cawley وزملائه الباحثين في التعمق بهذا الموضوع، فنشروا في عام 2012 دراسة أخرى [35]، اعتمدت على تقارير من ثلاثين تجربة تتعلق بتطبيق المقاربات الرشيقة في تطوير البرمجيات السريع. ونتج عنها أن ممارسات ومفاهيم ومبادئ المنهجيات الرشيقة يمكن تطبيقها في العمليات السريعة، بهدف التحسين المستمر لهذه العمليات. إضافة إلى أنها قد وجدت أنه لا يوجد منهجية تناسب كل المنظمات وكل أنواع التطبيقات، فمن الواجب وضع القواعد المتناسبة مع سياق التطوير، أهداف المشروع، والقيود، مع الأخذ بالاعتبار الجوانب المختلفة للمنهجيات السريعة والرشيقة قبل الخوض في التبني الكامل لهذه المنهجيات من قبل المنظمة، وهو إحدى التحديات الأبرز في هذا المجال. كما طرحت الدراسة إحدى العقبات التي تواجه الباحثين، وهي عدم وجود تعريف موحد معتمد لهذه المنهجيات، واقترحت ضرورة إجراء أبحاث تطبيقية أعمق في هذا السياق للحصول على فهم عميق عن التشابه والاختلاف بين نوعي المنهجيات.

2.3.3 الدراسات المرجعية التي تحدثت عن DevOps

مع ظهور منهجية DevOps، قامت Lucy Ellen Lwakatare وآخرون [36]، بإجراء دراسة شملت مقارنة بين منهجية DevOps ومفاهيم المنهجيات السريعة والرشيقة، إضافة إلى مفهوم النشر المستمر، من أربع جوانب وهي الأصل، التبني، التحقيق، الأهداف.

خلصت الدراسة إلى أن منهجية الDevOps نشأت من مفهوم النشر المستمر، كتطور ناتج عن عمليات التطوير البرمجية السريعة، وهي عملية مقادة بمبادئ المنهجيات الرشيقة. وخلصت إلى ضرورة اتباع مبادئ وممارسات وقيم المنهجيات السريعة، واستخدام مبادئ وممارسات المنهجيات الرشيقة للحصول على تبنٍ ناجح لDevOps . كما أظهرت الدراسة أن تطبيق DevOps يؤدي إلى الإصدار السريع والمنتظم للبرمجيات وبجودة عالية. لاحظنا أن الدراسة لم تدعم نتائجها بأية دراسة حالة وإنما اكتفى الباحثون بوضع خلاصة أبحاثهم المستندة على تجارب الآخرين.

لم تتوقف الأبحاث عند تلك النقطة بل ظهرت دراسة أخرى وضعها Cheng Wang وChangling Liu [37] عن إطار عمل لتبني منهجية DevOps ضمن أسلوب عمل المنهجيات السريعة لتحسين عمل المنهجيات السريعة وحل مجموعة من التحديات الشائعة ضمن مجتمع تطوير البرمجيات، واستندوا في ذلك على مجموعة كبيرة من الدراسات والأبحاث بالإضافة إلى إجراء مقابلات واستبيانات مع باحثين وممارسين اختصاصين في المجال، وخلصوا إلى توصيف إطار عمل خاص يقوم على دمج منهجية Scrum مع منهجية DevOps، بالإضافة إلى استعراض مجموعة من التحديات الأكثر شيوعاً وبعض الحلول الممكنة لها، لكن الدراسة لم تتطرق إلى أية مقارنة توضح الفرق بين أسلوب عمل المنهجيات السريعة والDevOps وإنما قامت بافتراض أن الDevOps هي عنصر مساعد في عملية التطوير السريع، واستندت بشكل كامل على أبحاث واستبيانات ومقابلات دون إجراء تطبيق فعلي لإطار العمل الناتج.

قام كل من هيمن وآخرون "Hemon et. al." [38]، بدراسة نظرية تحدثت عن خطوات الانتقال من منهجية Agile ضمن المنظمات التي تنتهجها إلى DevOps، وقاموا بدعمها من خلال دراسة حالة، وركزوا فيها على ضرورة فهم دور الأتمتة في عملية التطوير، ودعم العمل بالمهارات اللينة Soft والتقنية Hard اللازمة لكل دور مسند في العملية بهدف تعزيز التعاون ومستوى التواصل اللازم لتحسين بيئة التطوير، والذي يختلف بناء على طبيعة المهمة المسندة.

نقدت الدراسة القصور في الدراسات السابقة، كونها لم تجد إلا دراسة واحدة تطرقت إلى وضع نموذج واضح لإدارة عملية الانتقال إلى DevOps، ولكن وضعت هذه الدراسة لأهداف تجارية بحتة. وقام الباحثون بذكر قصور دراستهم أيضاً بأنها لم تجرب نظريتها إلا ضمن شركة واحدة ولم تشرك

جهة خارجية تقوم بعملية التحقق من صحة النتائج validation، لذلك نصحت بإجراء المزيد من الدراسات للتوجه بذكاء نحو DevOps.

في حين ذكر غاندوماني وموساي "GANDOMANI & Mousaei" [39]، ضمن دراسة مرجعية ركزت على توضيح مفاهيم المنهجيات السريعة والرشيقة والDevOps وممارساتها وأساليب عملها، وأكدت الدراسة أن DevOps قد اعتمدت واستفادت من النوعين الآخرين. لاحظنا أن هذه الدراسة اقتصرت على الجانب النظري دون أن تدعمه بدراسة حالة عملية، أو الخوض في بناء نموذج كامل واضح لإطار عمل خاص ب DevOps كسابقتها، ولكنها عززت بعضاً من المفاهيم النظرية الضرورية لدراستنا.

تناولت دراسة غوڤيل وآخرون "Govil et. al." [40]، دراسة حالة لتطبيق وب طوّر وفقاً لدمج إطار عمل ال Scrum مع ثقافة DevOps، حيث استخدمت السكرم بهدف تنظيم العمل والتنسيق بين الأفراد، والDevOps لتسريع وتنظيم عملية التسليم وضبط جودة التطبيق، وأكدت الدراسة على الممارسات الضرورية الواجب تواجدها أثناء العمل وفقاً لثقافة DevOps، ولكنها لم تطرح منهجية البحث التي اعتمدت عليها لتأكيد نتائج الدراسة المستخلصة، ولكنها أضافت لبحثنا معلومات قيمة عن تفاصيل والخطوات المستخدمة لتطوير التطبيق وفقاً لدمج المنهجيتين التي قمنا بوضعهما تحت الدراسة.

في حين ناقش ماروكيان وآخرون "Maroukian et. al." [41]، بعضاً من المفاهيم النظرية والممارسات لكل من منهجيات التطوير DevOps و Agileو Lean، ضمن مجموعة منظمات كبيرة الحجم ومختلفة التقانات والثقافات، وتطرقوا إلى الجوانب الإدارية في عملية تطوير البرمجيات، والأدوار الضرورية لإدارة المشاريع البرمجية، واعتمدوا في ذلك على استخدام مقاربات البحث ضمن البيانات الكيفية Qualitative Data، كأسلوب المقابلات مع العاملين في المنظمات. خلصت الدراسة إلى مجموعة من النتائج أهمها الممارسات والمبادئ الأهم في عملية التطوير كعمليات المراقبة واعتماد أساليب Scrum وKanban والتوصيل المستمر Continuous Delivery، وأكدت على ضرورة وضع الآلية المناسبة لدمج جميع الأطراف في عملية التطوير. لم تحتوي الدراسة على أي تطبيق حقيقي أو دراسة حالة، لتأكيد أو لرفض النتائج المستخلصة، الأمر الذي نصحت الباحثين بالقيام به بالاعتماد على نتائج هذا البحث للتحقق منها.

تحدث سالتز "Saltz" [42]، عن وضع إطار عمل جديد لدمج DevOps ضمن بيئات التطوير، بالاعتماد على Scrum و Kanban وأطلق عليه اسم SKI أو Structured Kanban Iteration،

حيث انتقل بممارسات ومبادئ هاتين المنهجيتين إلى مستوى أعلى، من خلال الاستفادة من أفضل ممارساتهما، والتعديل بما يتناسب مع العمل ومتطلبات DevOps. قدمت الدراسة مثالاً تطبيقياً لإطار العمل الجديد، ولكن لم تتحقق من صحة الدراسة، وحددت مجموعة من المقاييس الممكن استخدامها لتحقيق ذلك، ونصحت الدراسات اللاحقة باختبار إطار العمل والتحقق منه. كانت ميزة هذه الدراسة أن إطار العمل المطروح سيساعد الشركات التي تعمل وفقاً لScrum، بالانتقال إلى نموذج مدعوم بDevOps بسلاسة، وقد أغنت هذه الدراسة تجربتنا حيث قامت بالإضاءة على أهم النقاط في دمج هذه المنهجيات.

4.3 تحليل الدراسات المرجعية

بناء على ما تم الاطلاع عليه من دراسات سابقة لاحظنا أن التسلسل الذي ظهرت فيه منهجيات تطوير البرمجيات ما كان إلا تطوراً طبيعياً ناتجاً عن التطور التكنولوجي المستمر لأساليب وتقنيات العمل ولبيئات نشر المنتجات البرمجية، وكنتيجة لذلك نجد أنه لا يكفي للحصول على منتج برمجي جيد التركيز فقط على وضع خطة عمل مدروسة ومضبوطة زمنياً ومادياً –كما حددت المنهجيات التقليدية–، أو التركيز على أسلوب تنظيم فريق التطوير وعلاقته مع الزبون وتقليل أي شكل من أشكال الهدر في أي مرحلة –الأمر الذي تناولته المنهجيات السريعة والرشيقة–، كما لم تكن محاولة دمج جميع أصحاب المصلحة وكل من له علاقة مباشرة بالمنتج البرمجي المطروح في عملية التطوير، كفرق الاختبار وضمان الجودة والعمليات –أسلوب العمل وفقاً لمنهجية DevOps– الحل الأمثل لتطوير جميع المنتجات البرمجية، وإنما استطعنا التأكيد أن العوامل التي تلعب الدور الأكبر في تفضيل منهجية على أخرى هي خصوصية المشروع البرمجي المطروح وطبيعة بيئة النشر النهائية له وطريقة تنظيم فرق المنظمة البرمجية وأساليب عملها.

من خلال الدراسات السابقة استطعنا التوصل إلى ما يلي:

- استطاعت منهجيات التطوير المقادة بخطة من وضع الإطار الأساسي لتنظيم عملية تطوير البرمجيات ونجحت في تطوير مشاريع برمجية ذات توصيف دقيق وواضح، إلا أنها لم تستطع أن تلقَ ذلك النجاح دوماً، وكانت أبرز الأسباب هي تغير متطلبات الزبائن خلال الفترة الزمنية المنقضية بين مرحلة إعداد الخطة والتسليم، مما أدى في كثير من الأحيان إلى فشل المشروع.
- في حين تمكنت المنهجيات السريعة والرشيقة من تلافي مشكلة الفترات الزمنية الطويلة التي ينتظر فيها الزبون ريثما يتم تسليمه ميزة أو أي قيمة حقيقة، وتمكنت من تسليمه منتج متوافق

بنسبة كبيرة مع متطلباته من خلال دمجه ضمن فريق العمل وإشراكه في اتخاذ القرارات وعملية متابعة تطوير الميزات، واعتبرت حلاً للعديد من المشاكل التي واجهت مطوري البرمجيات، ولكن سرعان ما بدأت بمواجهة تحديات جديدة نتيجة التطور التكنولوجي السريع الذي فرض وجود بيئات تطوير وأساليب عمل جديدة، بالإضافة إلى وجود تنظيمات مختلفة للفرق الموجودة ضمن مؤسسة التطوير الواحدة.

- حاولت DevOps معالجة المشكلات التي واجهت المنهجيات السابقة عن طريق طرح أساليب عمل جديدة واقتراحات لدمج جميع أصحاب المصلحة بدءاً من فرق التطوير مروراً بالزبائن وصولاً إلى فرق العمليات المسؤولة عن إعداد البنى التحتية لعمل التطبيقات في عملية التطوير بهدف الحصول على المنتج البرمجي بأسرع وأجود طريقة ممكنة، حيث استطاعت من إثبات نفسها كمنهجية قادرة على إنجاح عملية تطوير البرمجيات في البيئات السحابية، والتي تبنى من قبل فرق مقسمة تنظيمياً ومفرقة جغرافياً، وذلك من خلال أساليب عملها المؤتمتة.

- ظهرت العديد من الدراسات التي دمجت عدة منهجيات مع بعضها ك Scrumban، والتي كانت دمجاً لمنهجية Scrum مع Kanban، وSKI التي كانت دمجاً لScrum و Kanban مع DevOps، وهدف هذا النوع من الدراسات إلى إيجاد آليات للحصول على أفضل الميزات الخاصة بكل منهجية، وتلافي المشكلات التي قد تؤدي إلى تدني أدائها.

- لاحظنا أيضاً أنه ظهرت العديد من الدراسات التي تناولت دراسات حالة عن منهجية DevOps، وتنوعت هذه الدراسات في أسلوب طرحها لهذه المنهجية حيث عالجتها بعض الدراسات من الجانب الأكاديمي وتحدثت عن خطوات العمل وفقاً للمنهجية والنتائج المحسنة، وأخرى عالجتها من الجانب التطبيقي المهني، حيث عرضت النتائج التي يجلبها التبني الناجح لDevOps في المنظمات، بالإضافة الى المهارات اللازمة لتطبيقها.

1.4.3 القيمة التي أضافتها الدراسات المرجعية للبحث

استطعنا من خلال الدراسات المرجعية تحديد النقاط الأساسية التي سننطلق منها في بحثنا، وهي:

- ضبابية أسلوب عمل DevOps، وعدم وجود تعريف واضح رسمي معتمد لهذه المنهجية، الأمر الذي عادة ما يسبب قلقاً لدى المنظمات لتبني هذه المنهجية.

- تناولت العديد من الدراسات الحديثة أمثلة عن تطبيقات لمنهجية DevOps، ولكن لحداثة هذه المنهجية نصحت غالبية الدراسات إلى ضرورة وجود دراسات تطبيقية أكثر عن هذه المنهجية، لوضع أدائها وأدواتها وأساليب عملها تحت الدراسة والمراقبة.
- يفضل دمج أفضل الممارسات المستخلصة من منهجية DevOps، مع أفضل الممارسات الخاصة بالمنهجيات الأخرى، للحصول على منهجية تطوير برمجيات ذات أداء عال يؤدي إلى أفضل النتائج.

خاتمة:

استعرضنا في هذا الفصل مجموعة منتقاة من الدراسات المرجعية التي استند عليها بحثنا، واشتملت على دراسات تحدثت عن منهجيات تطوير البرمجيات، بدءاً من ظهورها وصولاً إلى منهجية DevOps.

ساهمت هذه الدراسات بتوضيح النقاط الرئيسية التي سننطلق منها، وأضاءت على مجموعة من العقبات الواجب تجنبها، والميزات الواجب تعزيزها ضمن بحثنا.

الفصل الرّابع

إطار عمل DevOps المقترح والأدوات المستخدمة

مقدمة

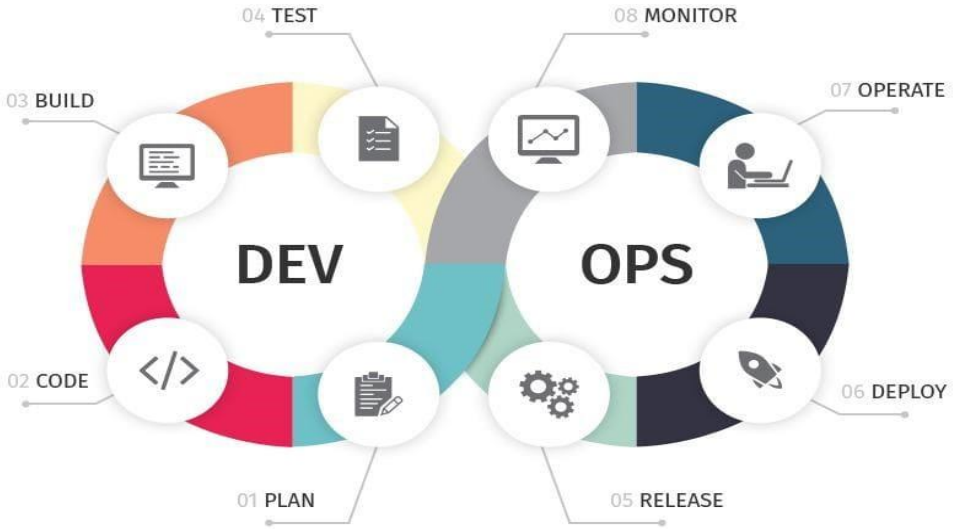
سنقوم في هذا الفصل بشرح إطار عمل DevOps المقترح وما هي مراحل عمله، بالإضافة إلى ذكر الأدوات الممكن استخدامها لدعم تطبيق إطار العمل هذا في بيئة العمل.

1.4 منهجية البحث

قمنا باستخدام منهجية بحث SLR [5]، بهدف وضع إطار عمل DevOps المقترح. في حين قمنا بالبحث ضمن المواقع التقنية للشركات الرائدة في هذا المجال، من أجل معرفة الأدوات المستخدمة لتطبيق DevOps ومن ثم قمنا بالمقارنة.

2.4 إطار عمل DevOps المقترح

بناء على ما تم دراسته في الفصل الثاني والثالث من هذه الأطروحة، قمنا بوضع إطار عمل يعتمد على منهجية DevOps في تطوير البرمجيات، وسيتم العمل وفقاً لما يلي:



الشكل (4-1) منهجية DevOps

الإطار العام للعمل سيكون وفقاً لمنهجية السكرم، حيث سيتم ضبط مرحلة جمع المتطلبات وفقاً للاجتماعات المحددة ضمن هذه المنهجية ومن ثم ستقسم المتطلبات وتنظم وفقاً لما يعرف باسم قصص المستخدمين User Stories وستجمع في سجل المنتج Product Backlog.

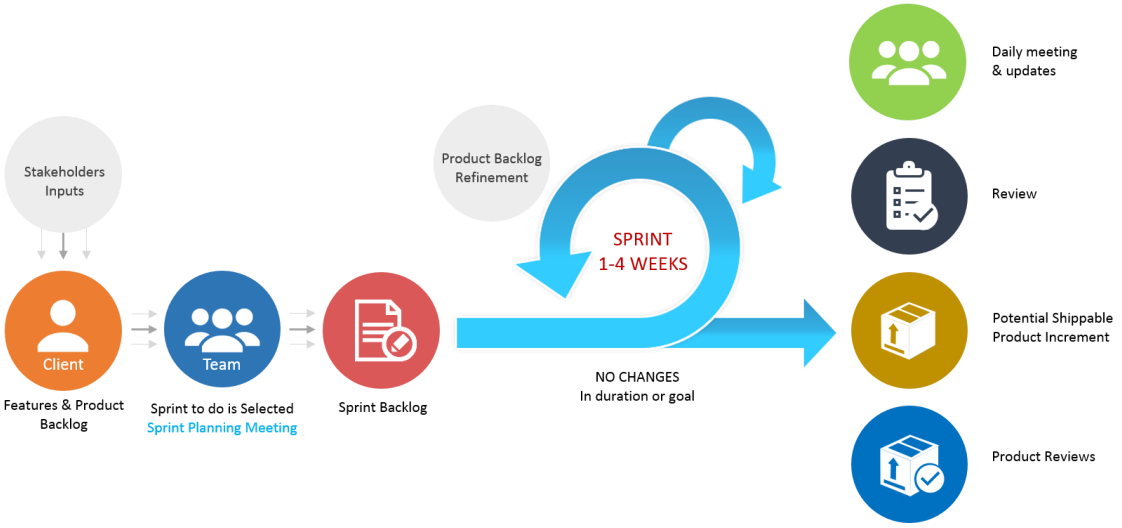
فور الانتهاء من جمع المتطلبات والشروع في عملية التطوير سيتم دمج مفاهيم وأسلوب منهجية Kanban في تنظيم المراحل المتبقية من تحليل وتصميم وتطوير وصولاً إلى النشر، وفقاً للوح الKanban. ستضيف منهجية الDevOps إلى هذا اللوح عملية المراقبة المستمرة للمنتج المنشور وذلك بهدف معرفة مدى رضا الزبائن عن المنتج، "وهو ما يمكن تشبيهه باجتماع مراجعة الSprint الخاص بـScrum، ولكنه سيحول إلى عملية مستمرة تشمل الزبائن لأن المنتج سيكون في مرحلة النشر ويتم الحصول على التغذية الراجعة منهم".

العملية بشكل كامل ستكون محكومة بإحدى أهم المفاهيم التي تتبناها الDevOps وهي مسارات العمل المؤتمتة Automated Pipelines.

1.2.4 مرحلة التطوير

سيتم اعتماد منهجية الScrum لإدارة عملية التطوير، واستخدام منهجية Kanban في كل Sprint بهدف تحسين سير العملية بشكل كامل قدر الإمكان، حيث سيتم إدارة سجل المنتج وقصص المستخدمين باستخدام لوح الكان بان. وفي كل Sprint سيتم العمل على تحسين هذه المرحلة من خلال قيم الكان بان كإزالة الهدر والحد من العمل قيد الإنجاز.

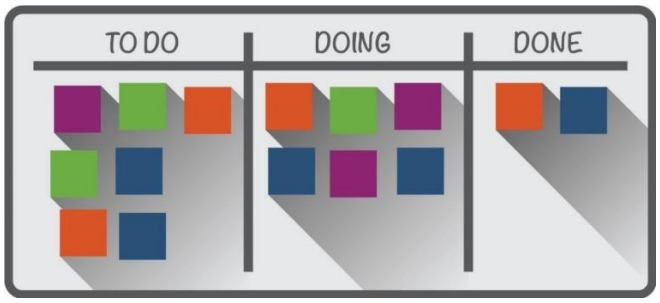
- **Scrum**



الشكل (4-2) منهجية Scrum

بناء على ما تم جمعه من خطوط عريضة لأهم الميزات المطلوبة، سيتم إعداد سجل المنتج Product Backlog، وترتيبه حسب الأولوية، ويحدد في كل Sprint أهم الميزات "قصص المستخدمين" للتطوير.

- **Kanban**



الشكل (4-3) لوح Kanban

عند تحديد سجل المنتج، واختيار أهم قصص المستخدمين للSprint القادم، سيتم تحديد مراحل العمل في Sprint وفقاً للوح ال Kanban، المقسم كالتالي:

التحليل			التصميم			التطوير			الاختبار			النشر		
(#WIP)			(#WIP)			(#WIP)			(#WIP)			(#WIP)		
To Do	In Progress	Done	To Do	In Progress	Done	To Do	In Progress	Done	To Do	In Progress	Done	To Do	In Progress	Done

ولكن سيتم تعديل اللوح ليشمل إحدى مراحل العمل في DevOps وهي مرحلة المراقبة المستمرة، ليصبح اللوح كالتالي:

التحليل			التصميم			التطوير			الاختبار			النشر			المراقبة		
(#WIP)			(#WIP)			(#WIP)			(#WIP)			(#WIP)			(#WIP)		
To Do	In Progress	Done	To Do	In Progress	Done	To Do	In Progress	Done	To Do	In Progress	Done	To Do	In Progress	Done	To Do	In Progress	Done

حيث يتم تحديد العمل قيد الإنجاز (work in progress) #WIP بما يتناسب مع حجم فريق العمل والمهام.

2.2.4 مرحلة الاختبار

سيتم اعتماد أهم مبادئ الDevOps وهي استخدام الاختبارات المؤتمتة، حيث سيتم كتابتها بالتوازي مع مرحلة التطوير ليتم استخدامها في كل Sprint، بهدف اختبار قصص المستخدمين المطورة.

3.2.4 مرحلة التكامل

سيتم دمج العناصر المطورة حديثاً والمختبرة مع الشيفرة البرمجية الأساسية، وذلك من خلال إحدى الأدوات المستخدمة لتحقيق هذا الهدف، ومن ثم سيتم إعادة تنفيذ الاختبارات المؤتمتة على النظام.

4.2.4 مرحلة النشر

سيتم نشر التطبيق بعد تطوير الميزات الجديدة الخاصة بالSprint الحالي باستخدام أدوات النشر المستمر Continuous Deployment فور جهوزيتها دون الانتظار لانتهاء الSprint، ودمجها مع المنتج الأصلي للمستخدمين من خلال أدوات التكامل المستمر Continuous Integration، ليتم بعدها الانتقال إلى مرحلة المراقبة.

5.2.4 مرحلة المراقبة

سيخضع التطبيق للمراقبة "يفترض تطبيقها من قبل فريق العمليات بالتعاون مع فريق الاختبار"، بهدف تسجيل المشكلات الظاهرة في أثناء الاستخدام، بهدف تحسين الميزات المطورة وزيادة جودتها.

تنتهي المراحل عند المراقبة، ليتم تكرارها من جديد خلال ال Sprint التالي.

3.4 التقنيات والأدوات المعتمدة

طورت العديد من الأدوات التي تساعد وتدعم عمليات التطوير باستخدام مفاهيم DevOps، بعضها كان مفتوح المصدر ومجاني والبعض الآخر كان مدفوعاً. في الجدول التالي سنقوم بذكر مجموعة من الأدوات التي من الممكن استخدامها لتحقيق مفاهيم DevOps في كل مرحلة من مراحل التطوير:

المرحلة	الأدوات
جمع المتطلبات	Trello – Asana – Jira – **Azure Boards**
مستودع الشيفرة المصدرية	Github – Gitlab – Bitbucket – **Azure Repos**

الاختبار	Junit – Selenium – **Azure Test Plans**
التكامل والنشر	Jenkins – Docker – Puppet – Chef – **Azure Pipelines**
المراقبة	Splunk – Nagios – **Azure Monitor / Azure Insights for Applications**

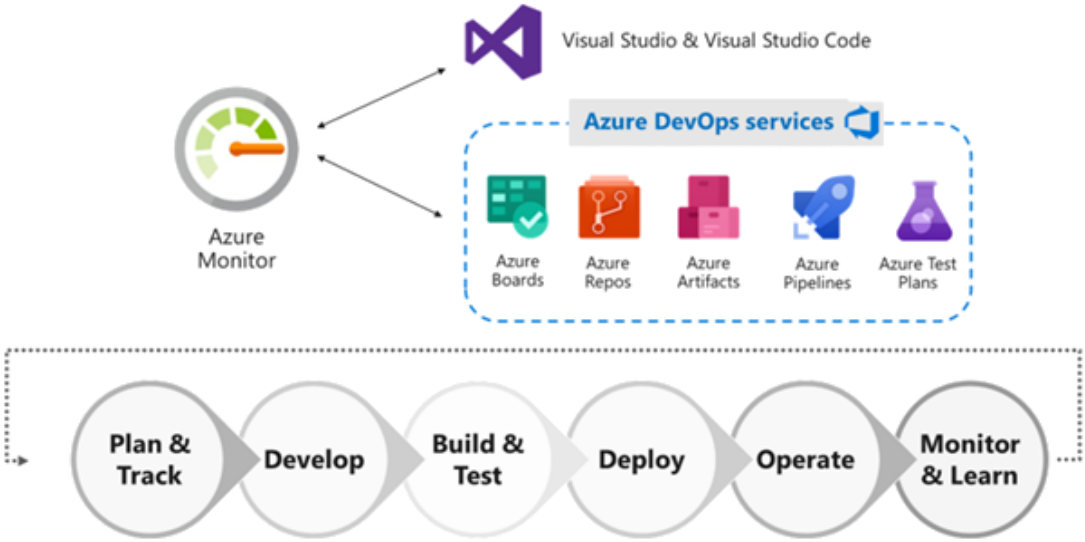
الجدول (4–1) جدول يعرض الأدوات الممكن استخدامها في كل مرحلة تطوير باستخدام DevOps

1.3.4 الأدوات المختارة في سياق الدراسة الحالية

1.1.3.4 مخدم Azure DevOps Server 2019

يعتبر هذا المخدم هو تطوير لمنتج مايكروسوفت الأسبق Team Foundation Server (TFS)، وهي النسخة التي توفر للمنظمات إمكانية تنصيب جميع ميزاتها على بيئتهم وضمن مخدماتهم، حيث يدعم هذا المخدم أسلوب العمل وفقاً لمنهجية DevOps بالكامل، بالإضافة إلى إمكانية تطبيق أسلوب المنهجيات السريعة باستخدامه أيضاً [47].

تقدم هذه الخدمة مجموعة الأدوات الضرورية لإدارة عملية التطوير باستخدام منهجية DevOps، حيث تدمج ما يلي:



الشكل (4–4) الأدوات المقدمة من Azure DevOps Server

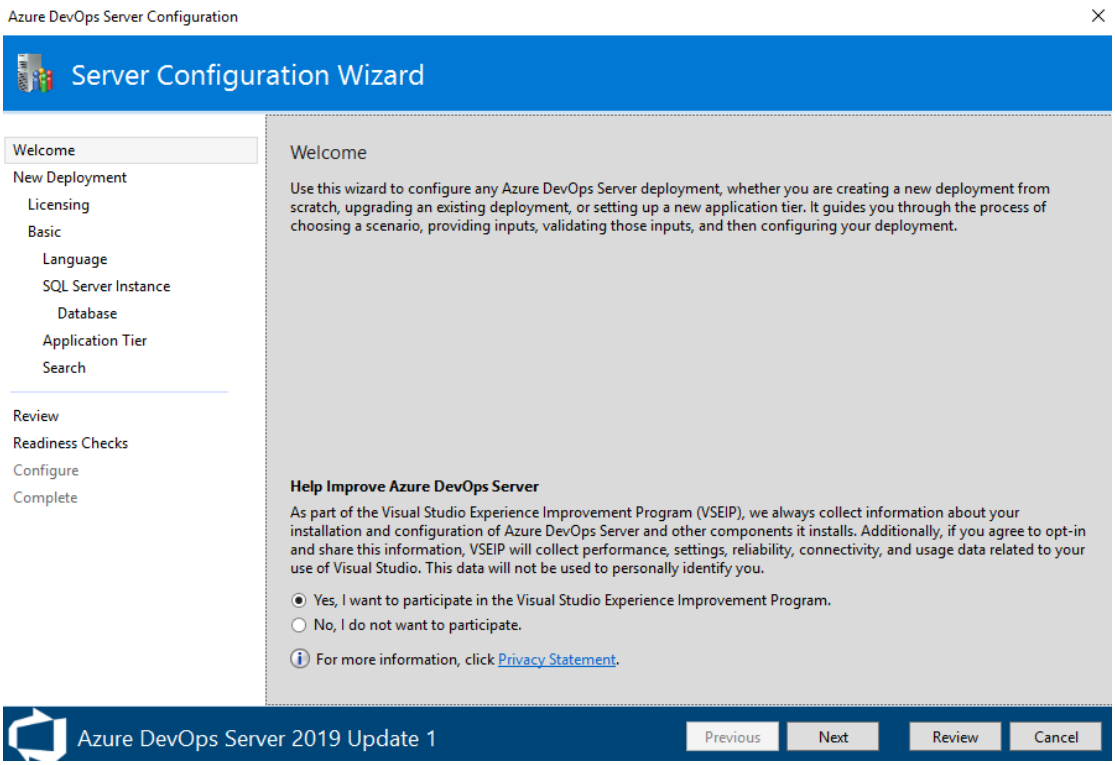
1. **ألواح Azure – Azure Boards:** حيث يقدم مخدم Azure DevOps الأدوات الضرورية للتطوير السريع المعتمد على مفاهيم Agile والتي تتيح إمكانية إدارة الخطط، والتعقب، ومناقشة العمل بين الفرق.
2. **مسارات عمل Azure - Azure Pipelines:** تساعد في بناء مسارات عمل لبناء واختبار ونشر التطبيق باستخدام آليات التكامل المستمر والنشر المستمر CI/CD، المتوافقة مع أي لغة، منصة، أو سحابة، ويمكن مكاملتها مع مستودعات الشيفرة المصدرية كال GitHub أو أي مزود خدمة Git آخر.
3. **مستودعات Azure – Azure Repos:** تتيح إمكانية إضافة مستودعات "مشاريع برمجية" غير محدودة، بخيارات متقدمة لإدارة الملفات.
4. **خطط اختبار Azure – Azure Test Plans:** تقدم الأدوات اللازمة لإدارة عمليات الاختبار المختلفة.
5. **Azure Artifacts:** يتيح إمكانية إنشاء، استضافة، مشاركة الحزم مع الفرق، ووضع حزم جديدة لمسارات عمل CI/CD .
6. **Azure Insights Monitor:** لمعرفة أداء التطبيق، ومراقبة معدل الطلبات والاستجابة، ومعدل الفشل، الاستثناءات، جلسات المستخدمين وغيرها.
7. **سوق الإضافات Extensions Marketplace:** يمكن دمج العمل مع Slack، وسحابة SonarCloud، و1000 تطبيق وخدمة أخرى.

2.1.3.4 لماذا اخترنا Azure DevOps Server؟

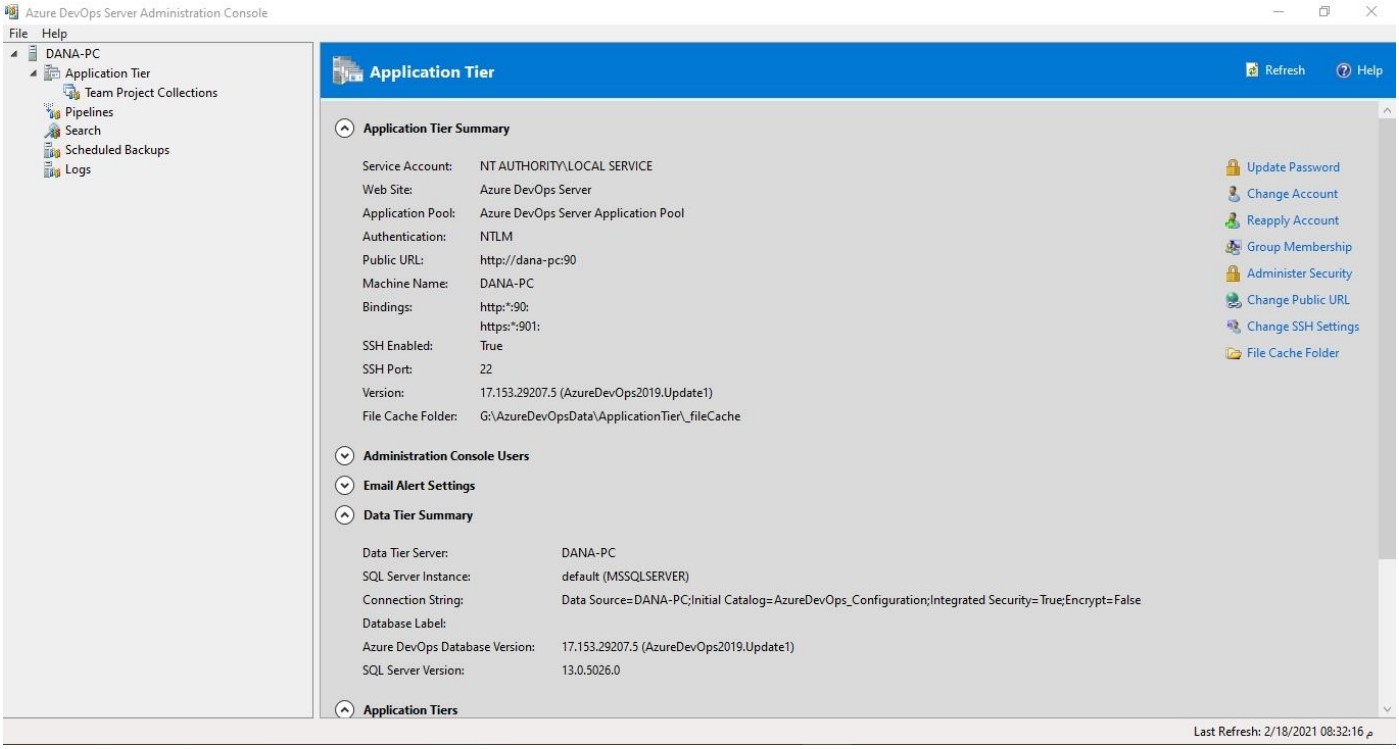
اخترنا في دراستنا استخدام Azure DevOps Server 2019 لإدارة عملية التطوير بشكل كامل كونه يقدم مجموعة متكاملة وموجهة لأسلوب التطوير باستخدام منهجية DevOps مع إمكانية استخدامه لتطوير تطبيقات موجهة لبيئات عمل مختلفة وليس فقط تلك البيئات التي تدعمها Microsoft.

في حين وجدنا أنه لو استخدمنا تطبيقات أو أدوات أخرى، كنا سنحتاج وقتاً أكبر لعملية التعلم عليها، وإيجاد أساليب لدمجها مع بعضها لتحقيق أفضل نتيجة مرجوة منها، الأمر الذي سيشكل عبئاً إضافيا على عملية التطوير البرمجي من وجهة نظرنا.

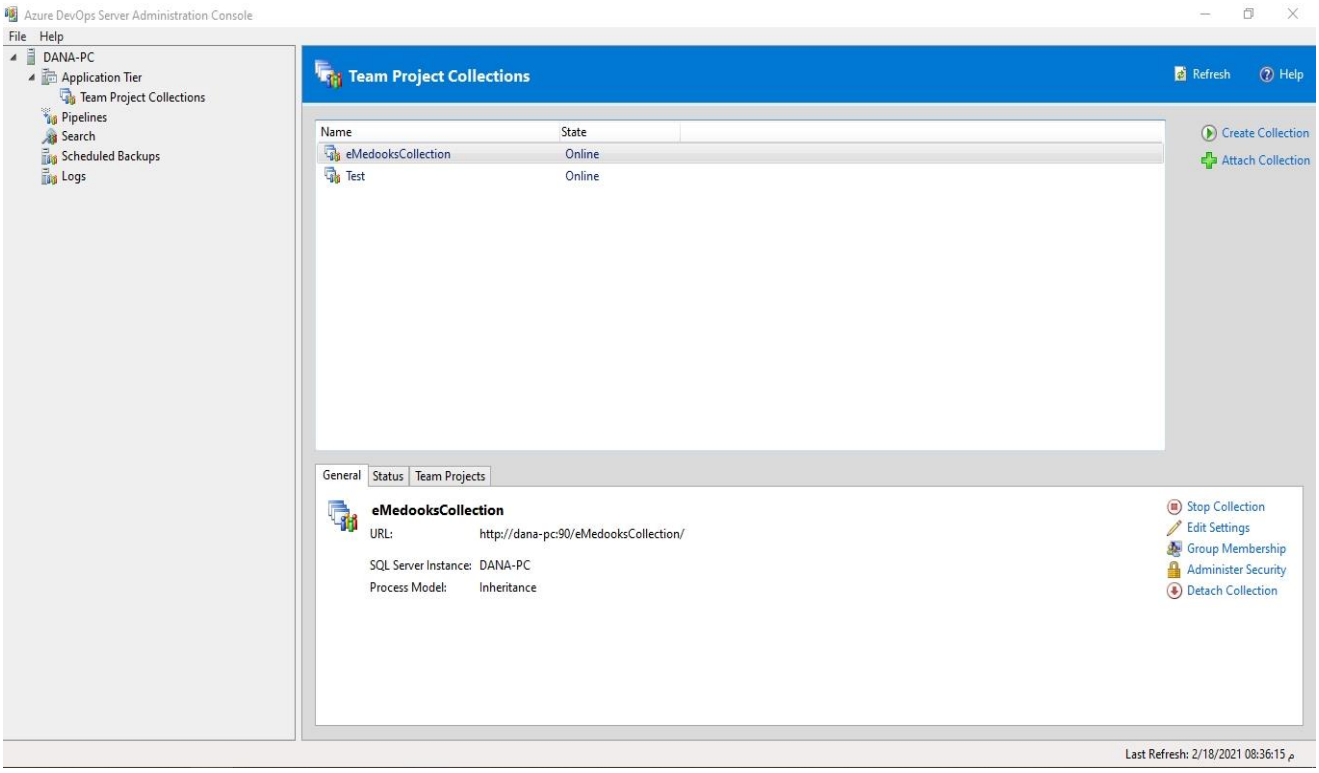
3.1.3.4 تنصيب واستخدام Azure DevOps Server



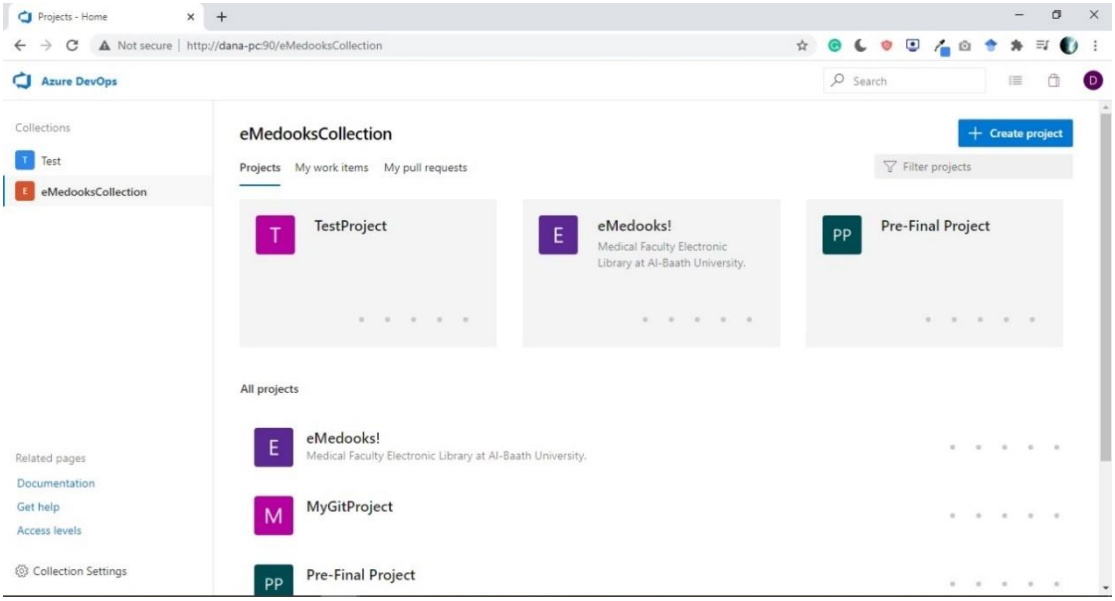
الشكل (4-5) واجهة التنصيب ل Azure DevOps Server



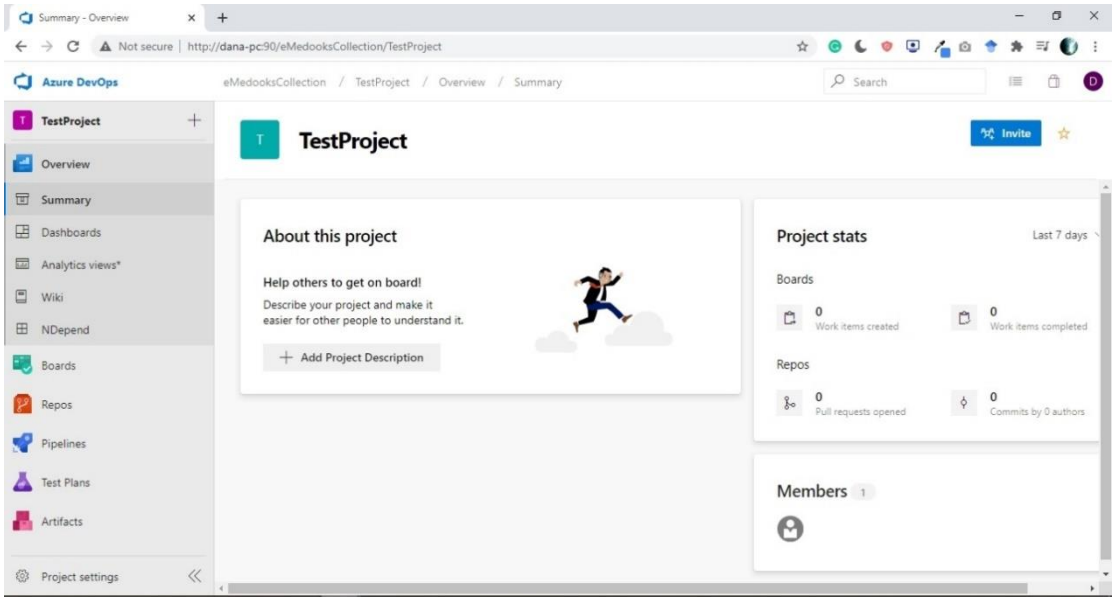
الشكل (4-6) واجهة إدارة Azure DevOps Server



الشكل (4-7) واجهة إدارة طبقة التطبيقات



الشكل (4-8) واجهة إدارة المشاريع Azure DevOps Server



الشكل (4-8) واجهة المشروع في Azure DevOps Server

خاتمة

قمنا في هذا الفصل بشرح إطار عمل DevOps المقترح والمستخلص من الدراسات المرجعية التي قمنا بتحليلها في الفصل السابق، كما قمنا بشرح الأدوات التي اعتمدنا عليها لوضع إطار العمل المقترح تحت الاختبار والدراسة.

الفصل الخامس

دراسة حالتين: تطوير تطبيقي وب

مقدمة

سنقوم في هذا الفصل باستعراض دراسة حالتين قمنا بوضعهما لدراسة الأثر الذي يحدثه تبني DevOps مقارنة بالمنهجيات السريعة. طبقنا إطار عمل DevOps المقترح في الفصل الرابع في إحدى دراستي الحالة واستخدمناه لتطوير تطبيق وب موجه للاستخدام من قبل جهة حكومية في حمص سوريا، وفي الأخرى قمنا باستخدام منهجية تطوير سريعة Agile، ورصدنا الفرق والتحسين الذي حصلنا عليه في الحالتين.

1.5 منهجية البحث

قمنا باستخدام منهجية دراسة الحالة Case Study، والتي عادة ما تستخدم من أجل رصد ظاهرة معينة تحدث ضمن سياقها الأصلي[47]. حيث وجدنا أنها ستكون مناسبة في مراقبة النتائج الذي ستحدث نتيجة تطبيق منهجية تطوير برمجيات، ضمن منظمة برمجية تمتلك مجموعة من الفرق وتعمل على تطوير مشروع برمجي. واستخدمنا على وجه الخصوص المقاربة الخاصة بدراسة الحالات المتعددة، لأن الهدف من بحثنا هو معرفة الأثر والتحسين الذي سيحدثه تطبيق منهجية DevOps ضمن المنظمة مقارنة بغيرها، فلذلك كان لابد من إيجاد حالة أخرى نستطيع المقارنة بها واخترناها بحيث تكون مشابهة من حيث الجهة الطالبة للمشروع وحجم المشروع وعدد أعضاء الفريق، وأن تكون خاضعة تقريبا لنفس الشروط، ولكن تطبق منهجية عمل مختلفة.

من الواجب ذكر أن هذا النوع من منهجيات البحث هو الأنسب للاستخدام في سياق عملنا، ويعود ذلك إلى خصوصية مفاهيم هندسة البرمجيات التي تفرض صعوبات في عملية جمع بيانات كمّية لاستخدامها في القياس والمقارنة وفقاً لأسس إحصائية، ولذلك توجهنا إلى إحدى المقاربات البحثية التي تمكننا من التعامل مع البيانات النوعية/الوصفية Qualitative الممكن استخلاصها من تجارب تطوير البرمجيات.

1.1.5 تصميم الدراسة واختيار الحالة Study Design and Case Selection

بنيت دراستي الحالة وطورتا في فترتين زمنيتين مختلفتين، حيث أجريت الأولى في الفترة ما بين شهر تشرين الثاني 2017 إلى نيسان 2018 وتبنينا أسلوب المنهجيات السريعة لتحقيقها، والثانية منذ شهر تشرين الأول 2019 حتى أيار 2020 واعتمدنا إطار عمل DevOps المقترح.

قمنا باختيار منهجية البحث هذه لأنها ستعطينا القدرة على فهم DevOps بشكل أعمق ودراسة خطوات الانتقال لها بشكل يتناسب مع طبيعة العمل في البيئة المدروسة، انطلاقاً من دراسة الحالة التي استخدمت فيها منهجية سريعة Agile، وتحديد الأثر الذي أحدثته DevOps ضمن المنظمة البرمجية وذلك ما قمنا به في دراسة الحالة الثانية.

- **اختيار الحالة:** جرت الدراسة الكاملة في سوريا، محافظة حمص، وتميزت هذه الدراسة بخصوصية كبيرة، لأن الممارسين للعمل البرمجي فيها يعانون إلى الآن من نقص في الفهم والمعرفة الخاصة بمنهجيات تطوير البرمجيات، ولذلك كان من الصعب إيجاد منظمات قادرة على مواكبة متطلبات دراسة الحالة أو العمل وفقاً لضوابطها، لذلك وبعد البحث تم اختيار منظمتين تمتلكان قسم مختص بتقانة المعلومات يهتم بالتعامل مع أحدث التقنيات وتطوير آليات العمل.

ركزنا في اختيارنا أيضاً إيجاد تطبيقين متشابهين من ناحية النوع، والحجم، والهيكل التنظيمي للجهة الطالبة وعدد أفراد فريق العمل وخبرتهم في المجال، والهدف من ذلك كان التركيز على محور الدراسة والذي يمكن اعتباره العامل المتغير فيها، وهو منهجية تطوير البرمجيات.

أخذت دراسة الحالة الأولى حيزاً ضمن منظمة خيرية غير ربحية وهي جمعية البر والخدمات الاجتماعية، مكتب تقانة المعلومات – قسم الحلول التقنية والبرمجية، والذي خاض تجربة العمل وفقاً لمنهجية تطوير سريعة Agile لتطوير مشروع مشترك مع المفوضية السامية للأمم المتحدة لشؤون اللاجئين UNHCR.

وطورت الدراسة الثانية أيضاً في سوريا – حمص – جامعة البعث كلية الطب البشري، حيث تم استخدام إطار عمل DevOps، والذي استفدنا من التجربة في الحالة الأولى في صياغته، بالإضافة إلى الاعتماد على الأبحاث والدراسات السابقة، وكان الهدف بناء نظام إدارة مكتبة الكترونية خاصة بالكلية.

كان الهدف من كلا الدراستين الحصول على أفضل البرمجيات وأكثرها أمناً بأسرع الوسائل وأكثرها فاعلية، ومعرفة أي المنهجيات ستحقق هذا الهدف وفقاً لهذه الشروط.

الجدول (5-1)

وصف للعاملين في كل حالة والملاحظات.

الحالة	العاملين	الملاحظات
الحالة الأولى "نظام سند الموزع"	4 مطورون، مدير المشروع منسق "من طرف الزبون"، تقنيان	كان الباحث القائم على هذه الدراسة هو مدير المشروع والمشرف على عملية التطوير، مما أثرى الدراسة وزاد مدى دقة المعلومات المستخرجة.
الحالة الثانية "المكتبة الالكترونية"	3 مطورون، مسؤول عمليات الأتمتة "التكامل والنشر"، مراقب، داعم تقني، مدير المشروع.	وضع الباحث القائم على هذه الدراسة إطار العمل الخاص بتطوير هذا المشروع.

2.1.5 جمع البيانات Data Collection

قمنا باستخدام مجموعة من الأساليب الخاصة بجمع البيانات النوعية Qualitative وهي المقابلات مع أصحاب المصلحة وكافة العاملين على كلا المشروعين، بالإضافة إلى إجراء ما يعرف بالمقابلة الجماعية الموجهة Focused Group Discussion، وكانت هذه المقابلة مدمجة ضمن الاجتماعات التي تنص عليها منهجية التطوير السريع، وكان النقاش مفتوحاً ضمن هذه الاجتماعات ومع وجود أسئلة محددة أساسية ذات نهايات مفتوحة بالإضافة لأسئلة ال (نعم / لا) وذلك للحصول على أكبر قدر من المعلومات لإثراء البحث.

جرت الاجتماعات بشكل دوري ومنتظم وعادة ما كانت تحدث وتسجل في موقع العمل ليتم لاحقاً كتابتها وأرشفتها. لم تجري أي اجتماعات عبر وسائل الاتصالات الفيديوية، ولكن بعض البيانات استخرجت أيضاً من خلال اتصالات هاتفية.

3.1.5 تحليل البيانات Data Analysis

قمنا بتحليل البيانات والوثائق التي حصلنا عليها مستندين على الأهداف والأسئلة الموضوعة لهذه الدراسة، والتي تركزت في اتجاهين هما معرفة العقبات التي واجهت آلية التطوير منذ بدايتها حتى نهايتها بالإضافة إلى التأثير والاختلاف الذي أضافه إطار عمل DevOps على هذه العملية، وكان العنصر الأساسي في عملية البحث هو الباحث الواضع لهذه الدراسة الذي كان جزءاً من عمليتي التطوير وهو الذي قام بمعاينة واستخراج الأنماط والسمات للبيانات ومراجعتها وتدقيقها من قبل مشرف البحث الأكاديمي والرجوع إلى ممثلي الزبائن للتحقق من مطابقة النتائج مع الواقع، معتمدين على التدوين اليدوي والبرامج المكتبية الالكترونية لتسجيل وتحليل النتائج، كما قمنا بتصنيفها بناء على كل مرحلة من مراحل التطوير ويوضح الجدول (5-2) هذه التصنيفات أو كما تسمى السمات [7] ،

وتعاملنا مع بيانات كل حالة على حدا "انظر الجدول (5-3)" وانتهينا باستخلاص النتائج منهما باستخدام أسلوب التحليل التقاطعي Cross-Case Analysis.

الجدول (5-2)

وصف للسمات المستخدمة لترميز البيانات المستخلصة

السمة / السمة الجزئية	الوصف
السياق / هيكلية المنظمة	البيانات الخاصة بآلية تقسيم المنظمة من أدوار ومسؤوليات وفرق وأصحاب المصلحة، والزبائن.
السياق / طبيعة المشروع	البيانات التي توضح متطلبات المشروع من وظائف أساسية ووظائف تحسينية، بالإضافة إلى طبيعة واجهات المستخدم المطلوبة.
المنهجية / جمع المتطلبات التحليل، التصميم، التحقيق، الاختبار، الصيانة والمراقبة	البيانات المستخلصة من هذه المراحل
تحديات العمل	العقبات التي واجهتها الفرق في أثناء عملية التطوير
الأدوات	جميع الأدوات المستخدمة من قبل الفرق والهدف من استخدامها وفي أي مرحلة استخدمت
التأثير / إيجابي أم سلبي	الأثر الذي أحدثه استخدام منهجية تطوير مقارنة بغيرها وماهي نتائجه.

الجدول (5-3)

وصف ملخص عن سياق دراستي الحالة

جانب الدراسة	دراسة الحالة الأولى "نظام سند الموزع"	دراسة الحالة الثانية "المكتبة الالكترونية"
وصف عن المؤسسة وحجمها	جمعية خيرية غير ربحية، 1000+ موظف/ة	كلية أكاديمية تابعة لجامعة البعث السورية، 2500+ موظف / مدرس
المشروع	تطبيق وب يخدم المراكز المجتمعية لتسجيل الخدمات المقدمة والمستفيدين منها	تطبيق وب عبارة عن منصة الكترونية تفاعلية للمراجع الأكاديمية المفيدة التي تخدم الطلاب
الزبون	المراكز المجتمعية المتعاقدة مع UNHCR	كليات جامعة البعث – حمص – سوريا
المستخدمون	موظفو المراكز المجتمعية كل حسب منصبه الوظيفي	طلاب وأساتذة جامعة البعث
حجم فريق العمل	4 مطورون، مدير المشروع، مسؤول IT من طرف الزبون، تقنيي صيانة وبنية.	3 مطورين، 1 مختبر، مراقب، تقني، مدير المشروع.
المدة للإصدار	في نهاية كل دورة sprint، ما يقارب شهر.	عند الانتهاء من تطوير أي ميزة من قبل أي مطور

معدل النشر	عند الانتهاء من ال sprint.	تقريباً بمعدل يومي للاختبار، وأسبوعي للنشر
أسلوب التواصل	الاجتماعات والاتصالات الهاتفية	العمل المشترك من خلال متابعة العمل على Azure board وإرسال الملاحظات عليه والاجتماعات والاتصالات الهاتفية.
عدد ساعات العمل	بمعدل 6 ساعات في 6 أيام العمل	بمعدل 6 إلى 7 ساعات في 5 أيام عمل
منهجية العمل	منهجية تطوير سريعة تعتمد على مبادئ إعلان الأجيل.	إطار عمل مبني على مفاهيم الDevOps، ودمج مع منهجية سريعة ومنهجية رشيقة لإدارة العمل بدءاً من عملية جمع المتطلبات والتحليل، فكان مزيجاً من Scrum لتنظيم المتطلبات وإدارة مدة الإصدارات، و Kanban لمراقبة العمل.
الأدوات	▪بيئة التطوير Visual Studio 2015 Community ▪نظام إدارة الشيفرة المصدرية Github ▪نظام إدارة قواعد البيانات Sql Server 2014 Express Edition ▪نظام Windows Server 2012 ▪مخدم IIS server 8.x	▪Azure DevOps Server 2019 ▪نظام إدارة الشيفرة المصدرية Azure Git. ▪نظام إدارة المتطلبات Azure Boards. ▪نظام إدارة المكتبات والمحفوظات Azure Artifacts. ▪نظام إدارة الاختبارات Azure Testlabs. ▪نظام إدارة الإصدار والنشر Azure Pipelines. ▪نظام إدارة قواعد البيانات Sql Server 2016 Enterprise Edition. ▪ Visual Studio 2019 Enterprise Edition. ▪مخدم IIS Server 8.x ▪نظام Windows Server 2012.

4.1.5 الأسئلة التي سيجاب عنها بعد تحليل دراستي الحالة

بنينا دراستي الحالة على مجموعة من الأسئلة وهي:

1. ما هي الصعوبات التي واجهت سير العمل في الحالتين؟ وماهي الحلول؟
2. ما هو الاختلاف الذي طرأ على أسلوب العمل من ناحية التعاون وتنظيم الفرق؟
3. ما هي التغيرات التي طرأت على مخرجات كل مرحلة من مراحل عملية التطوير؟

2.5 وصف دراستي الحالة

1.2.5 نظام سند الموزع بالتعاون مع UNHCR[1]

1.1.2.5 وصف النظام

تطبيق وب، طور بهدف متابعة حركة الأفراد والعائلات المستفيدة من الخدمات المقدمة من قبل مراكز سند الخدمية التابعة لجمعية البر والخدمات الاجتماعية بالتعاون مع منظمة UNHCR، ومتابعة المراحل التي تمر بها الخدمة المقدمة، حيث يعالج أي طلب لخدمة ما من قبل منسقي المراكز قبل تقديمها إلى المستفيدين. كما يوفر النظام إمكانية تصدير تقارير إحصائية خاصة بكل خدمة مقدمة، بالإضافة لإحصائيات عن المستفيدين ضمن المراكز.

تجمع البيانات من قبل **مدراء المراكز أو مسؤولي الاستقبال** في المراكز الخدمية وفقاً للصيغة والمعلومات المحددة من قبل منظمة UNHCR، ويتم إدخالها إلى النظام، الذي يقوم بالكشف عن البيانات المكررة، ونعني بذلك بيانات المستفيدين الذي سبق وحصلوا على خدمة أو سجلوا في مركز خدمي آخر.

تحول البيانات المدخلة إلى **مدراء الحالات**، وهم المسؤولون عن دراسة ومتابعة طلب التقدم لأي خدمة، وتقسم الخدمات المقدمة ضمن المراكز المجتمعية إلى خدمات تعليمية، حماية الطفل، خدمات قانونية، طبية وغيرها.

يعتمد مدراء الحالة في الكثير من الأحيان في عملية دراسة الطلبات المقدمة على **فرق متطوعي الوصول**، والذين يقومون بالذهاب إلى منازل المستفيدين للتأكد من حالتهم، ومطابقة البيانات التي قاموا بتقديمها مسبقاً مع الوضع الحقيقي لحالتهم.

تصدر تقارير أسبوعية وشهرية وسنوية من قبل **مسؤولي التقارير** في كل مركز خدمي، وتعطى بيانات إحصائية رقمية حصراً إلى منظمة UNHCR، وذلك بهدف معرفة إن كانت المراكز المجتمعية تقوم بإعطاء المساعدات والخدمات لجميع الأفراد بأسلوب عادل، مع المحافظة على سرية البيانات المأخوذة منهم.

2.1.2.5 الهدف وراء تطوير النظام

- **الوضع ما قبل تطوير النظام**

[1] مقتبس من وثيقة مشروع تحسين إدارة البيانات في المراكز المجتمعية المقدمة من قبل منظمة UNHCR.

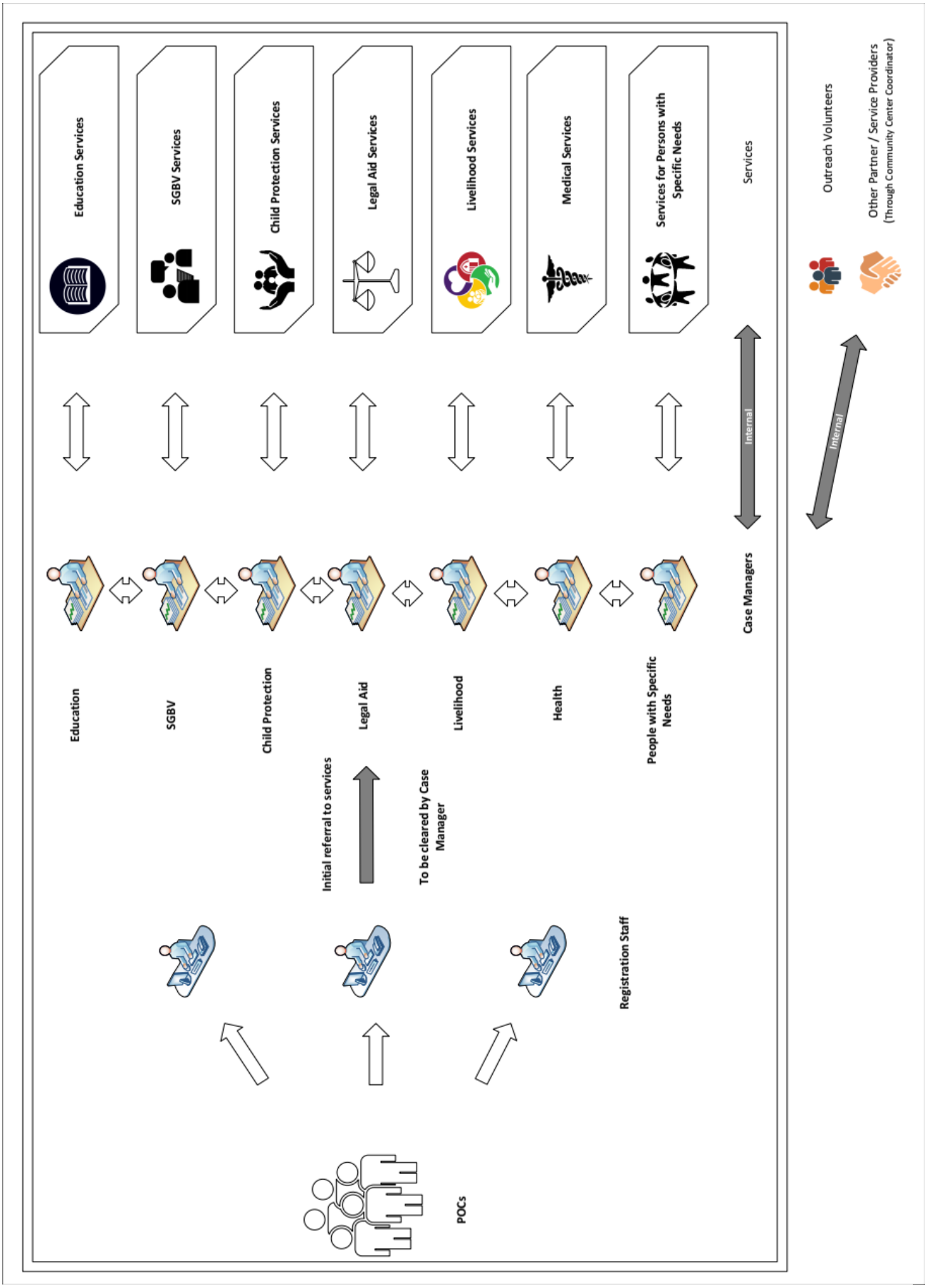
تقوم منظمة UNHCR بالتعاون مع العديد من المراكز الخدمية الموجودة في سوريا، بهدف توصيل مختلف الخدمات (صحية، تعليمية، قانونية ... وغيرها)، إلى الأفراد والعائلات المحتاجة، ولكن خلال السنوات التي تعاونت بها المنظمة مع هذه المراكز، واجهت مجموعة من المشكلات في عملية الحصول على معلومات دقيقة عن الخدمات والمساعدات المقدمة وآلية توزيعها على المستفيدين، ويعود ذلك إلى اعتماد كل شريك من شركاء المنظمة أسلوباً مختلفاً في عملية تسجيل البيانات الخاصة بالخدمات، والمستفيدين منها.

بالإضافة إلى أنه أصبح من الضروري إيجاد طريقة تضمن عملية توزع الخدمات بشكل عادل على مختلف الفئات المجتمعية، من خلال كشف التكرار في البيانات المدخلة، ومقارنة البيانات بين المراكز التابعة لشركاء مختلفين.

- **الهدف من النظام المطور**

وضع نظام معتمد قادر على جمع البيانات الخاصة بالمستفيدين من الخدمات التي تقدمها منظمة UNHCR عبر المراكز المجتمعية المختلفة، وكشف البيانات المكررة، والمقارنة بين مجموعة الخدمات المقدمة، والالتزام بآلية موحدة لعملية جمع البيانات بين مختلف شركاء المنظمة.

الهدف من ذلك التأكد من توزيع الخدمات والمساعدات بشكل عادل على مختلف الفئات المجتمعية، ورصد التأثير الذي تحدثه هذه المساعدات على المجتمع، ووضع الآلية المناسبة لتوصيل هذا الأثر إلى الجهات المانحة للمساعدات، لتعزيز الثقة في المنظمة وشركاؤها.



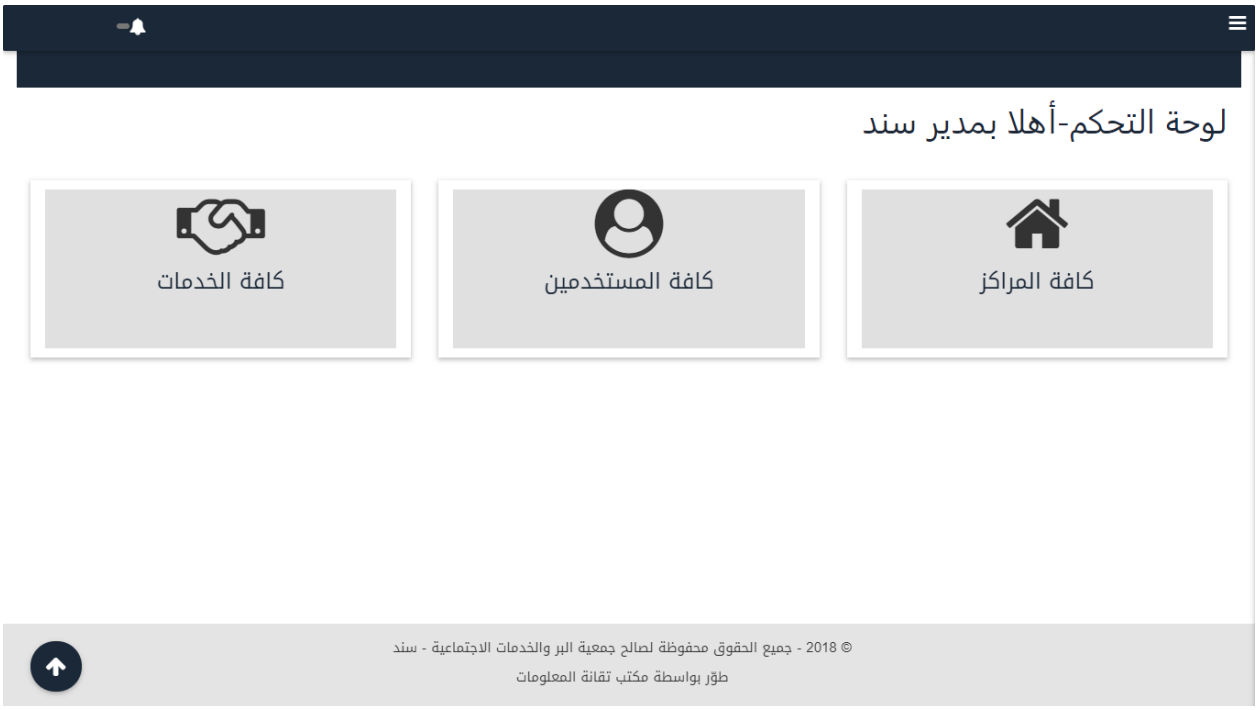
الشكل (5-1) مخطط تقديم الخدمات في نظام سند الموزع

3.1.2.5 الخدمات التي يوفرها النظام للمستخدمين وفقاً للصلاحيات

الصلاحيات التي يوفرها النظام (كل صلاحية مرفقة مع أسماء المناصب التي سوف سيمتلكونها ضمن النظام):

- **مدير النظام Admin**

1. إضافة مركز جديد.

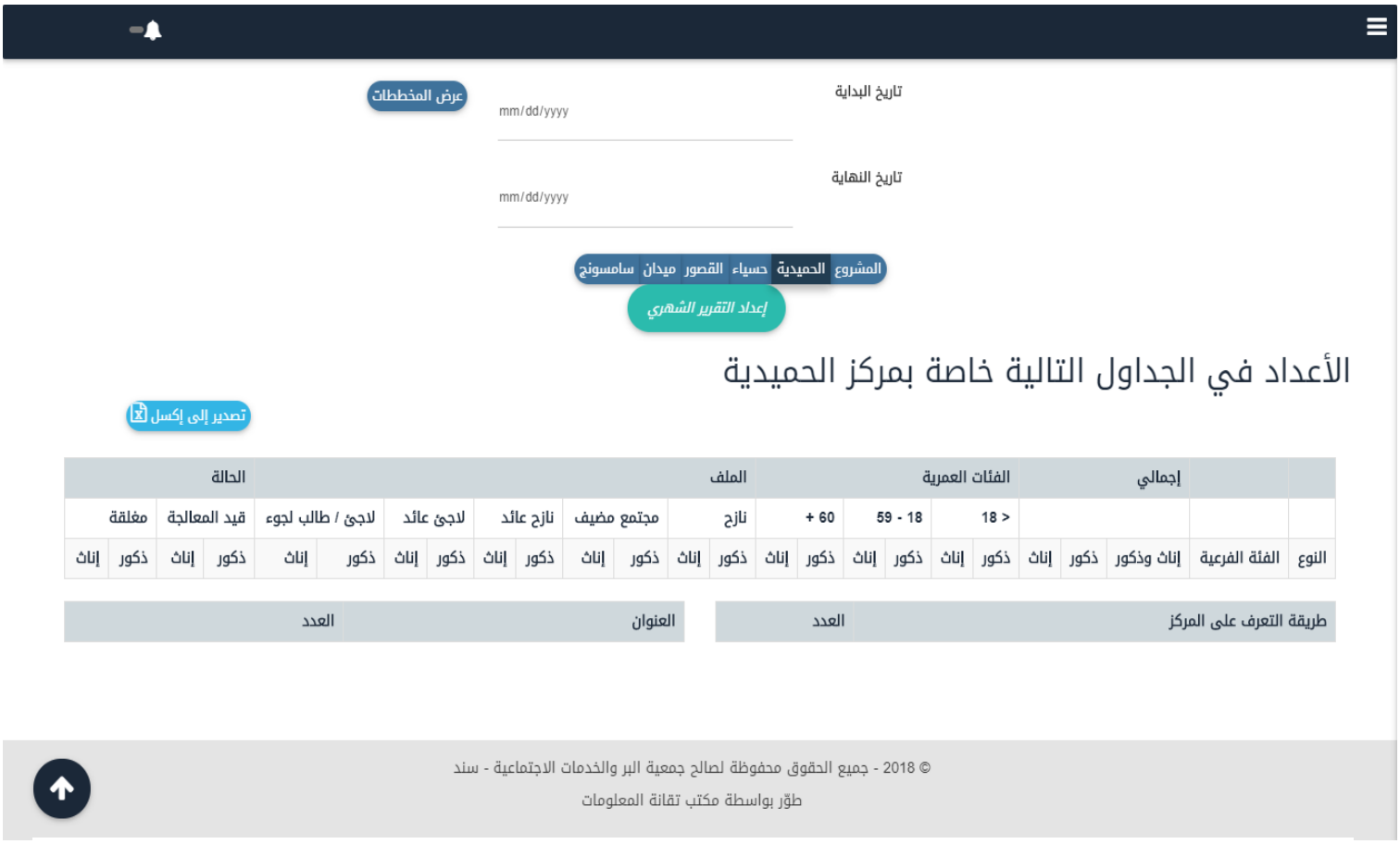


الشكل (5-2) لوحة التحكم في نظام سند الموزع

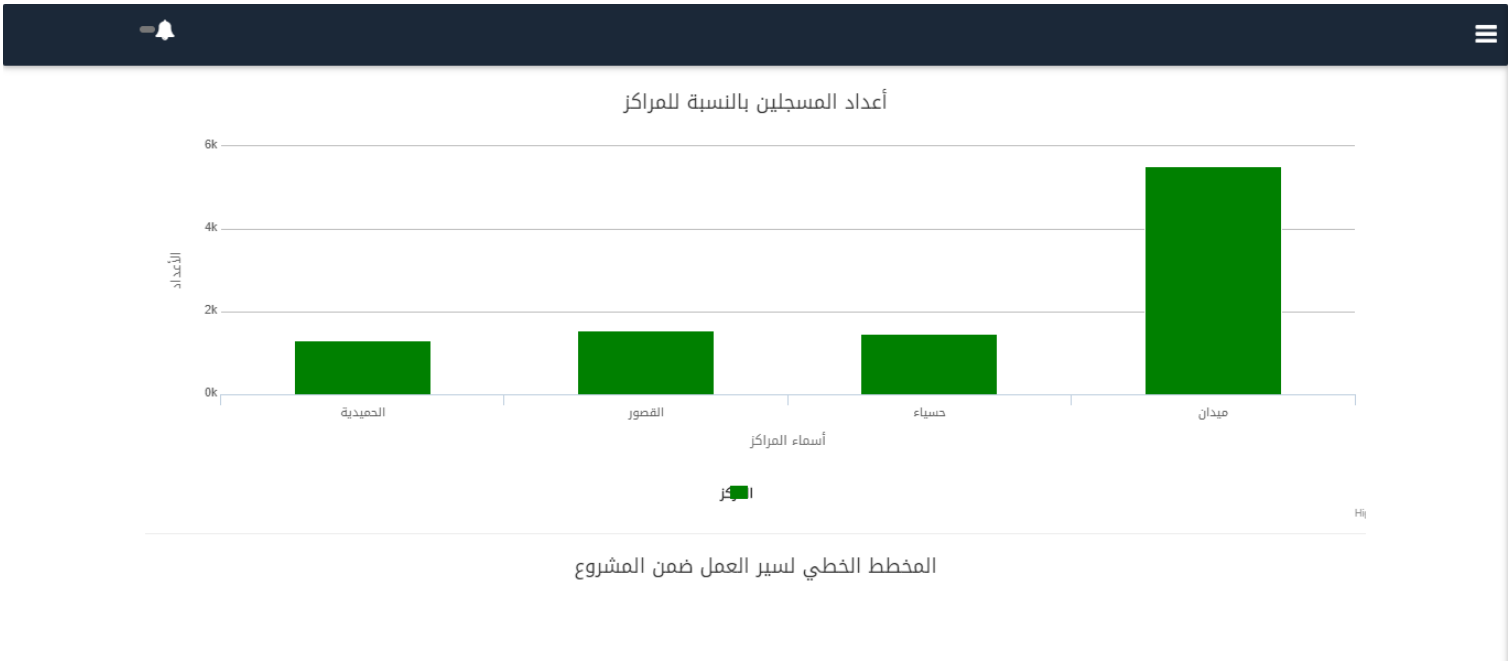
2. إضافة خدمة جديدة ضمن كل قسم.
3. إدارة الإحالات الخارجية المحولة من قبل الأقسام.
4. إضافة مستخدمي النظام (المنسقون – مدراء الحالة – قادة القطاع – مدراء المراكز – مسؤولو الاستقبال – الباحثون الاجتماعيون).
5. استعراض كافة التقارير من النظام.

- **معد/ معدة التقارير Reporter**

استعراض كافة التقارير من البرنامج.



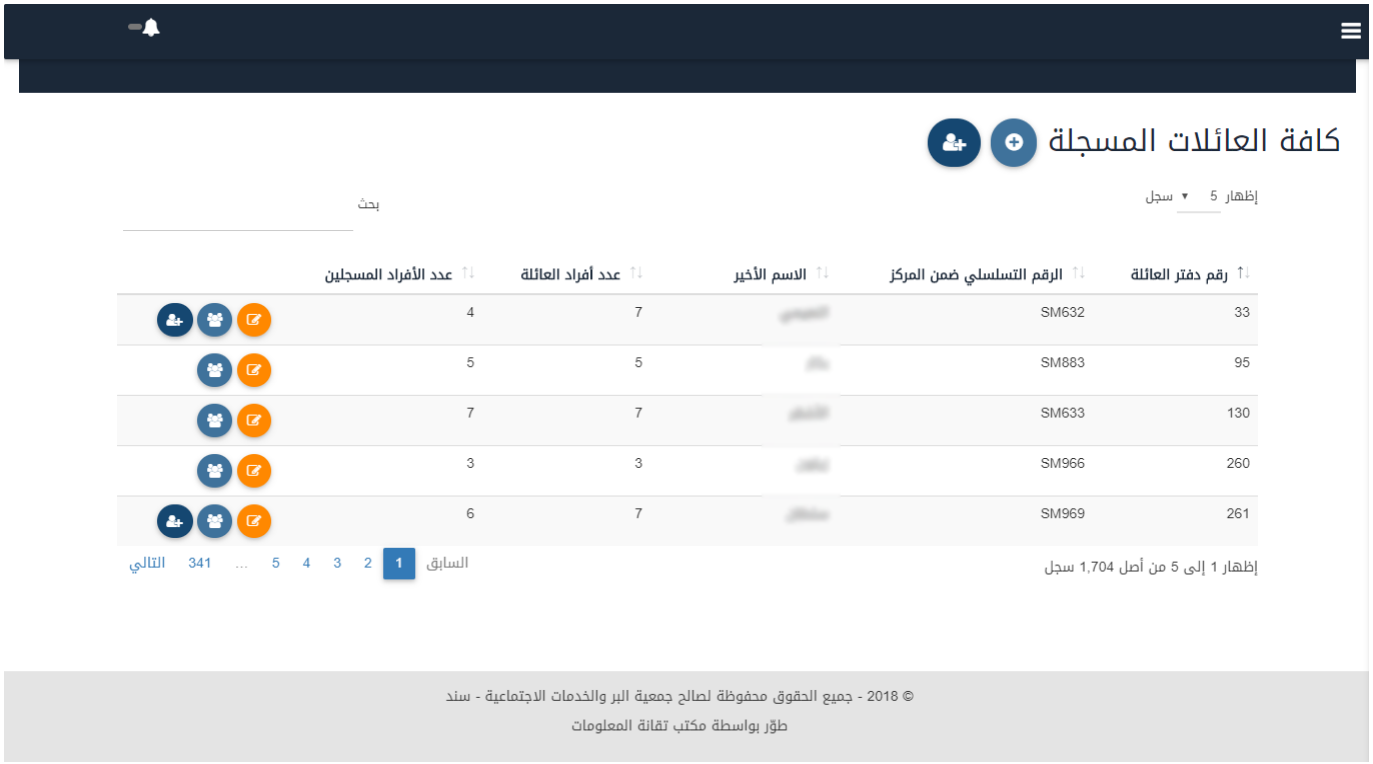
الشكل (5-3) التقارير الإحصائية الخاصة بالمراكز



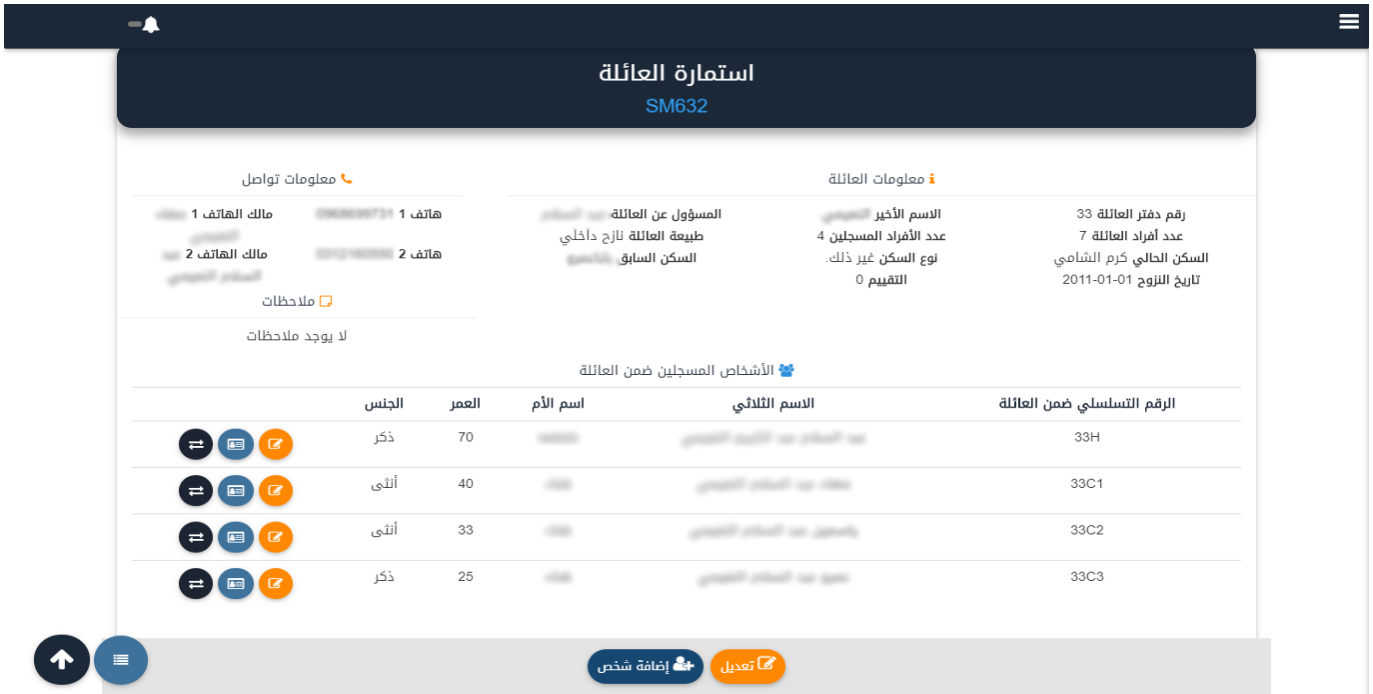
الشكل (5-4) المخطط الخطي لسير العمل ضمن مشروع سند التابع للمنظمة

- **موظفي الاستقبال Registration Staff (مدير المركز – مسؤول الاستقبال)**

1. إضافة بيانات عائلة جديدة، واستعراض بيانات عائلة ما.

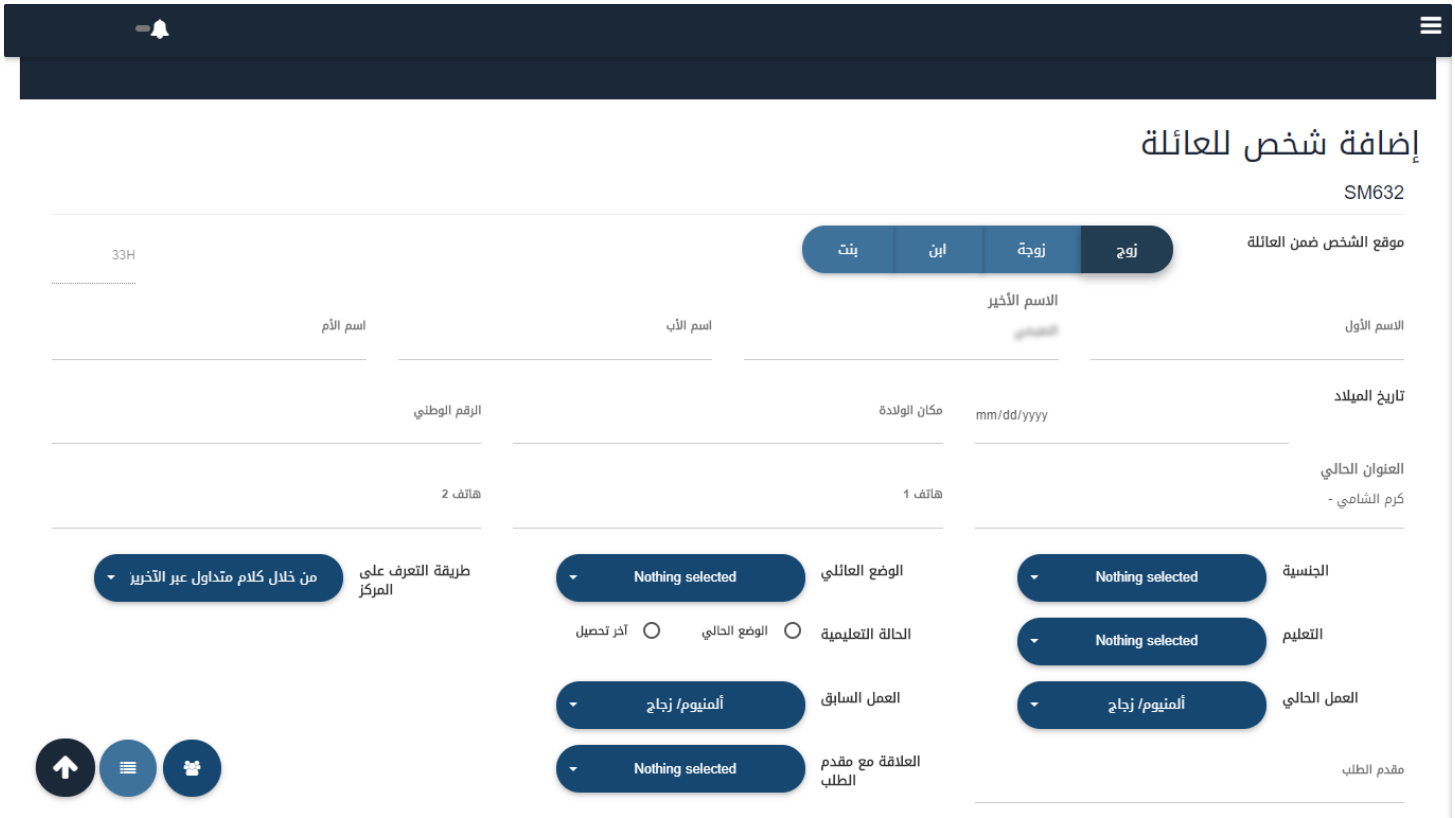


الشكل (5-5) استعراض العائلات وإمكانية إضافة عائلة جديدة

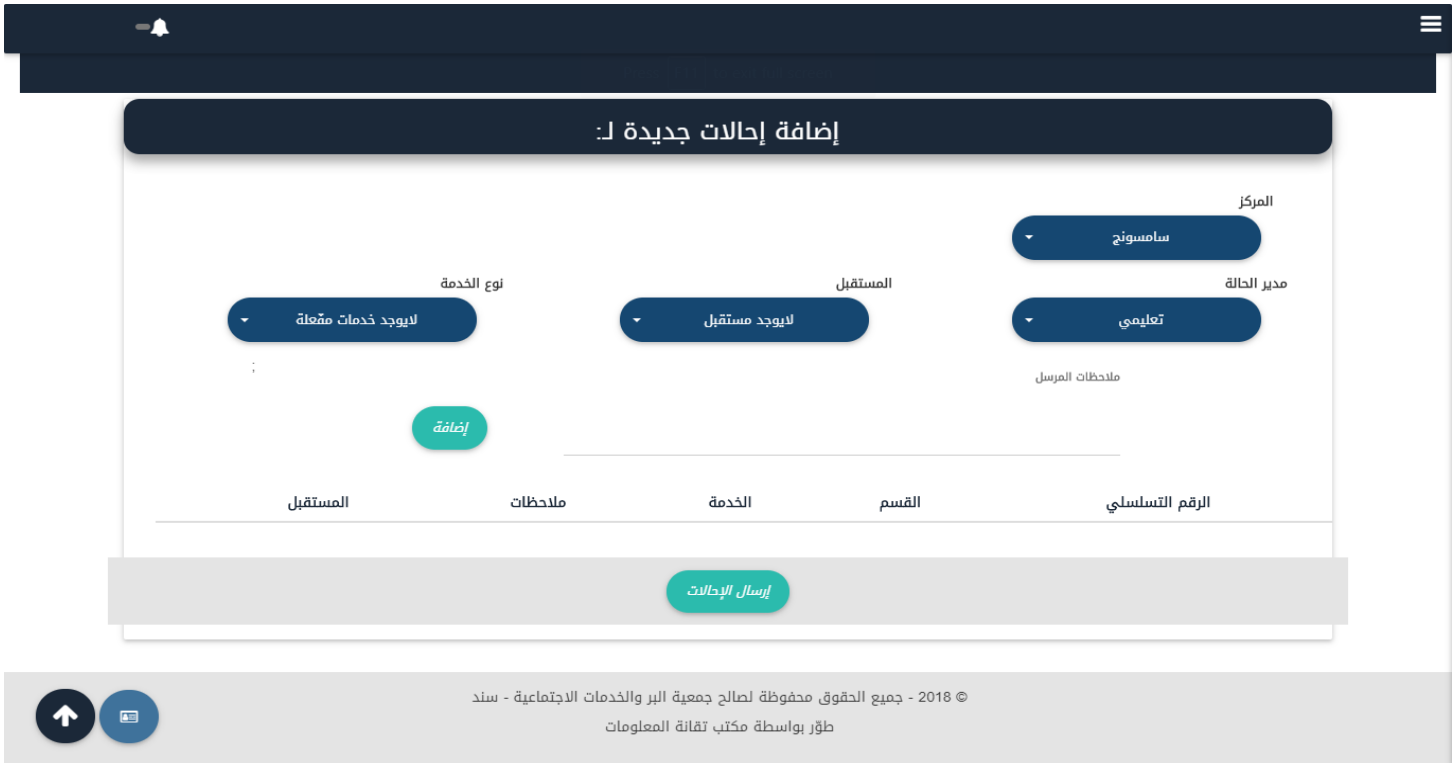


الشكل (5-6) استعراض بيانات عائلة

2. إضافة بيانات شخص جديد.



الشكل (5-7) إضافة بيانات شخص إلى عائلة



الشكل (5-8) واجهة تحويل الإحالات إلى مدراء الحالات

3. تعديل بيانات عائلة موجودة.

4. تعديل بيانات شخص موجود.

5. إحالة شخص إلى خدمة أو عدة خدمات حسب التقييم الأولي له، وذلك ضمن المركز الذي سجّل فيه الشخص بيانات الأساسية.

- **منسقي الأقسام Coordinator**

1. إدارة الخدمات ضمن القسم الخاص به (تفعيل أو إلغاء تفعيل).

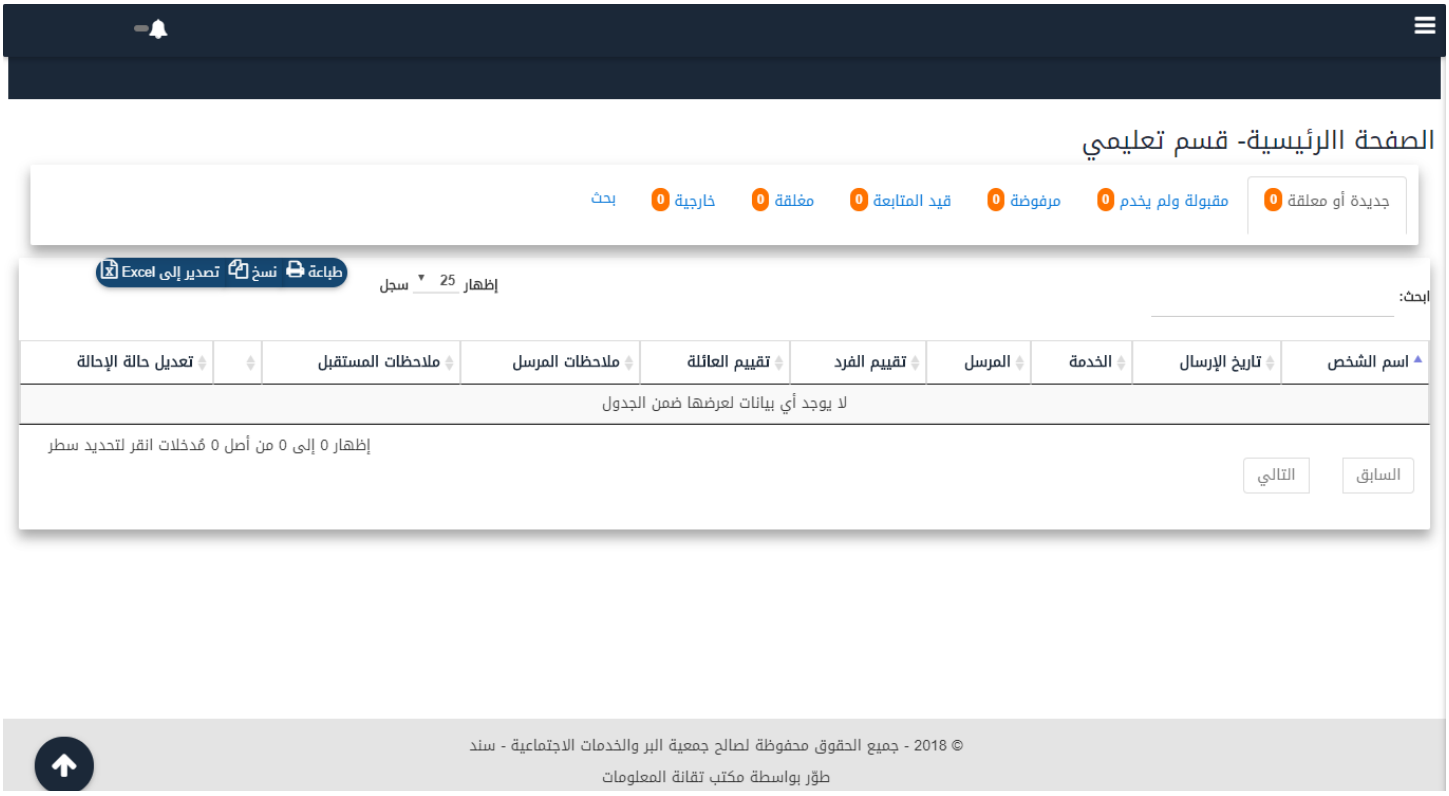
2. استعراض التقارير ضمن القسم الخاص به.

- **مدير حالة Case Manager**

1. إدارة الحالة المحولة لهم من قبل مسؤول الاستقبال أو من قبل مدراء الحالة الآخرين

2. إحالة حالة إلى قسم آخر.

3. استعرض الإحالات التي قام بإدارتها فقط حسب حالتها (حالة جديدة أو معلّقة – مقبولة ولم تخدّم – مرفوضة – قيد المتابعة – مغلقة –تمت إحالتها لفريق متطوعي الوصول).



الشكل (5–9) واجهة استعراض الحالات، وفقاً لمراحلها

2.2.5 المكتبة الالكترونية في كلية الطب البشري – جامعة البعث

تسعى كلية الطب البشري في جامعة البعث إلى زيادة موسوعتها العلمية وذلك من خلال محاولة تأمين أكبر عدد من المراجع العلمية الطبية المفيدة لطلاب الكلية سواء بشكلها الورقي أو الالكتروني التي عادة ما يكون من الصعب على الطالب تأمينها، إما لعدم امتلاكه الخبرة الكافية على اختيار الكتب التي تمتلك المعلومات الدقيقة المعتمدة، أو لصعوبة الحصول عليها نتيجة للواقع الجغرافي أو المادي.

لذلك قامت الكلية بتجهيز مكتبتها بمجموعة غنية من الموسوعات العلمية الطبية، إضافة إلى إعدادها لمكتبة الكترونية يتم رفع أحدث وأهم المراجع العلمية الطبية العالمية عليها وتحديث محتواها بشكل دوري.

استخدم في سبيل تحقيقها، تطبيق مكتبي Alfa Book Manager بنسخة مجانية، منصب على مخدم ذو نظام Windows Server 2012، متصل بمجموعة من الأجهزة الحاسوبية التي يستخدمها الطلاب للوصول إلى واجهة الوب المرتبطة بالتطبيق المكتبي لتحميل أو قراءة الكتب التي يحتاجونها.

استطاع التطبيق تقديم الخدمة الضرورية للطلاب بكفاءة، ولكن بسبب عدم القدرة على الحصول على النسخة المدفوعة كاملة الميزات للتطبيق كإمكانية إضافة والاستماع إلى الكتب الصوتية، وإضافة مكتبة خاصة بالفيديوهات، كان لابد من التفكير بحل آخر قابل للتطوير والتوسعة لمواكبة الاحتياجات المتزايدة تباعاً، فنتجت فكرة تطوير تطبيق وب للمكتبة الالكترونية يمتلك الميزات الضرورية اللازمة لتوفير احتياجات الطلاب، قابل للتوسعة عند الطلب.

1.2.2.5 الهدف وراء تطوير النظام

- **الوضع ما قبل تطوير النظام**

استطاع تطبيق Alfa Book Manager أن يلبي احتياجات الكلية من ناحية تنظيمه للكتب الالكترونية وتصنيفها، حيث يعتبر من أهم وأشهر تطبيقات إدارة المكتبات الالكترونية الموجودة حالياً والمستخدمة على نطاق واسع.

ولكن يجد مسؤولو النظام مجموعة من الصعوبات في أثناء تعاملهم معه وتلخص بالتالي:

- التعامل مع واجهة المستخدم، خصوصاً أنها باللغة الإنكليزية.
- آليات البحث ليست فعالة بشكل كاف.

- صعوبة في تصنيف الكتب، وإعطاءهم دلالات تعريفية Tags.
- صعوبة في إضافة الكتب.
- التطبيق مدفوع، لذلك لا يمكن الحصول على جميع الميزات التي يقدمها.

بالمقابل لا يمكن العمل على إجراء التعديلات على التطبيق لحل هذه المشكلات أو إضافة ميزات جديدة يحتاجها المستخدمون.

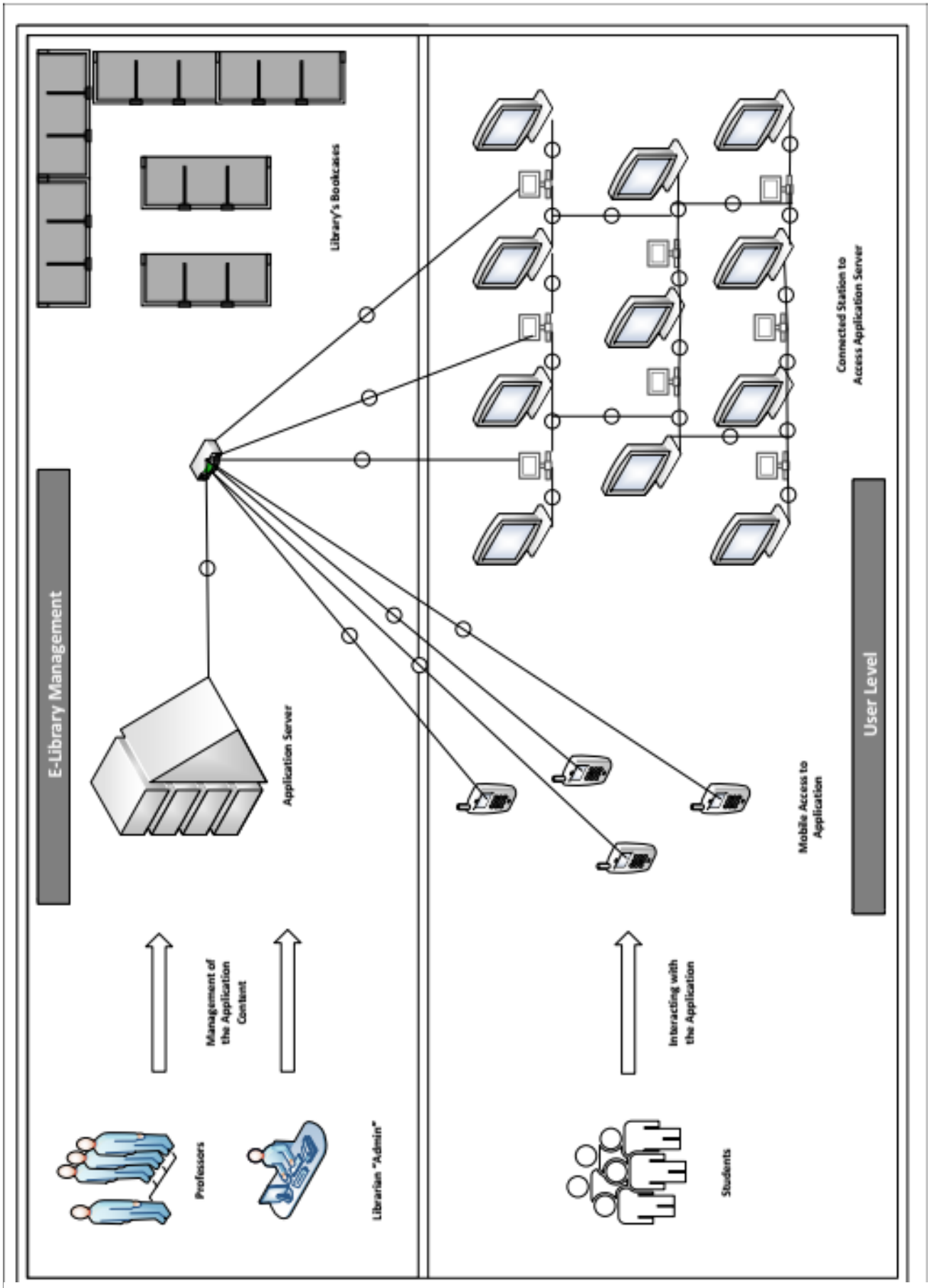
- **تطبيقات إدارة المكتبات الالكترونية**

قمنا بإجراء مقارنة تتعلق بميزات ومحدوديات أربع تطبيقات إدارة مكتبات الكترونية، تم اختيارها بهدف استعراض الميزات المختلفة التي يمتلكها هذا النوع من التطبيقات ومدى توافقها لاستخدامها في مؤسسة حكومية، أو في حال عدم توافقها، واستخلاص أهم الميزات التي سيحتاجها أي نظام سيبنى لهذا الهدف، وبنيت هذه المقارنة بناء على التجريب المباشر للتطبيقات الأربع واستعراض آراء المستخدمين المنشورة على مجموعة من المواقع التقنية، وهي[49] [48]:

	Alfa Book Manager	Calibre	LibraryThing	GoodReads
نوع التطبيق	تطبيق شبكي، جزء المخدم هو تطبيق مكتبي، ويوفر واجهة وب خاصة للمستخدمين	تطبيق شبكي، جزء المخدم هو تطبيق مكتبي، ويوفر واجهة وب خاصة للمستخدمين	موقع وب (منصة تواصل اجتماعي لاستعراض الكتب)	منصة تواصل اجتماعي لاستعراض الكتب، يمتلك نظام ناصح للمستخدمين
مدفوع	مدفوع مع وجود مجموعة من خطط الدفع، إضافة لفترة تجريبية	مجاني بالكامل	يمكن الحصول على حساب مجاني أو مدفوع على الموقع	يمكن الحصول على حساب مجاني على الموقع
مفتوح المصدر	–	مفتوح المصدر	–	–
ميزات التطبيق	1- إضافة مكتبة. 2- إضافة الكتب وبياناتها التعريفية مع إمكانية تعديلها. 3- قراءة وتحميل الكتب من خلال واجهة وب للمستخدمين. 4- إدارة الكتب الصوتية.	1- إضافة مكتبة. 2- إضافة الكتب وبياناتها التعريفية مع إمكانية تعديلها. 3- قراءة وتحميل الكتب من خلال واجهة وب للمستخدمين.	1- إضافة مكتبة. 2- إضافة معلومات عن الكتب الخاصة بالمستخدم. 3- استعراض المكتبات والكتب	1- إضافة مجموعة الكتب المقروءة أو التي يرغب المستخدم بقراءتها. 2- تحميل الكتب بناء على روابط

	5- تحويل الصيغ. 6- يقدم مجموعة من السمات لتغيير أسلوب العرض.	4- إدارة الكتب الصوتية. 5- تحويل الصيغ.	الخاصة بكل مستخدم. 4- وضع تقييم للكتب. 5- التفاعل مع المستخدمين الآخرين.	لمواقع التحميل ك amazon. 3- التفاعل مع المستخدمين الآخرين. 4- وضع تقييمات للكتب.
محدودياته	يستخدم على منصة Windows فقط، كما لوحظ ظهور مجموعة من الأخطاء عند التعامل مع بعض الميزات كتعديل اسم الكاتب أو إضافة الصور [49]	يعتبر أداء التطبيق سيئاً إذا ما قورن بغيره من التطبيقات ك Alfa Book Manager على سبيل المثال حيث يوجد بطئ كبير في إنجاز المهام [50] إذا ما قورن بتطبيقات أخرى.	منصة لعرض الكتب فقط والتفاعل بين المستخدمين، ولا يتيح إمكانية قراءتها أو تحميلها.	منصة تواصل اجتماعي مختصة بالكتب وتقييماتها، عادة ما ينتقد بسبب عدم قوة نظام النصح الخاص بالمنصة.
يصلح لمؤسسة حكومية ولماذا؟	يصلح ولكن توجد مجموعة من الأسباب التي تقف عائقاً في سبيل تبنيه بشكل كامل وهي: 1- مشكلة شراء التطبيق وضرورة وجود حساب paypal أو ما يعادلها. 2- عدم وجود إمكانية تعديله وتخصيصه بما يتناسب مع احتياجات المؤسسة حتى في حال شرائه.	يصلح، حيث يتيح هذا التطبيق إمكانية التعديل بما يتوافق مع احتياجات الكلية، ولكنها عملية ستتطلب وقتاً طويلاً وجهداً كبيراً.	لا يصلح، كونه عبارة عن منصة لعرض الكتب والتفاعل بين المستخدمين، والهدف الرئيسي من المكتبة الالكترونية هو توفير إمكانية تحميل الكتب للمستخدمين. فعملية إدارة المحتوى ليست خاصة بالكلية.	لا يصلح كونه منصة تواصل اجتماعي مستقلة قائمة بذاتها. ولا يمكن تخصيص عملية إدارة المحتوى.

الجدول (5-4) مقارنة بين أشهر تطبيقات إدارة الكتب الالكترونية



الشكل (5-10) مخطط يوضح طريقة العمل ضمن المكتبة الالكترونية

- **الهدف من النظام المطور**

تطوير تطبيق إدارة مكتبة الكترونية، قادر على تلبية احتياجات الطلاب في كلية الطب البشري، لتأمين الوصول السريع والمناسب إلى المراجع العلمية التي يحتاجونها، من خلال تطوير واجهة مستخدم بسيطة وسهلة الاستخدام، وتخصيصه بشكل متوافق مع المؤسسة التي سيتم تنصيب النظام فيها، وهي شعبة المكتبة الالكترونية في كلية الطب البشري.

سيتم تطويره ليكون عبارة عن منصة الكترونية تفاعلية مرتبطة بشبكة داخلية ضمن كلية الطب البشري، تتيح للطلاب الوصول إلى المراجع والكتب الطبية من خلال رابط يُدخل إلى المستعرض المنصب على أجهزتهم المحمولة أو الحواسيب المكتبية الموجودة في الكلية. تهدف هذه المنصة إلى دمج أهم الميزات الموجودة حالياً في تطبيقات إدارة الكتب الالكترونية ومنصات التواصل الخاصة بالكتب بشكل عام، لزيادة تفاعل الطلاب مع المكتبة وتحويلهم إلى مساهمين حقيقين فيها من خلال إتاحة إمكانية إضافة كتب، ووضع تقييمات ومراجعات.

سيشمل التطبيق المقترح مجموعة الميزات الموجودة في التطبيق الحالي الموجود Alfa Book Manager، وتشمل:

1- إضافة مكتبة وتعديلها.
2- إضافة تصنيفات للكتب.
3- إضافة الكتب وبياناتها التعريفية مع إمكانية تعديلها.
4- قراءة وتحميل الكتب من خلال واجهة وب للمستخدمين.
5- إضافة الكتّاب.

إضافة إلى مجموعة من الميزات الجديدة المطلوبة، وهي:

1- إضافة مكتبات صوتية ومرئية.
2- إمكانية ربط الكتب بأكثر من تصنيف.
3- توفير آليات بحث متنوعة (بناء على الكاتب، ISBN، .....).

سيتم تطوير المكتبة الالكترونية لتصبح منصة تفاعلية لطلاب كلية الطب البشري، حيث سيمتلك كل طالب حساب مستخدم يمكنه من القيام بما يلي:

1- استعراض الكتب الموجودة وتحميلها أو قراءتها.
2- إتاحة إمكانية رفع كتب أو مراجع، ولا تتاح للجميع إلا بعد موافقة مسؤول النظام.

3- إضافة تقييمات ومراجعات بالمراجع الموجودة.

4- التفاعل مع المستخدمين الآخرين.

الشكل (5-11) الصفحة الرئيسية لتطبيق المكتبة الالكترونية

3.5 سياق العمل في كل مرحلة

سنقوم باستعراض تفاصيل كل حالة على حدا في هذا القسم من النواحي التالية: المنهجية، أسلوب العمل وتتبع عمل الفريق، طبيعة المشروع، الجهة الطالبة للمشروع.

1.3.5 دراسة الحالة الأولى: نظام سند الموزع بالتعاون مع UNHCR

وضع نظام سند الموزع الذي قمنا بشرحه في الفقرات السابقة تحت البحث، لموائمته مع هدف هذه الدراسة وهو تطبيق وب تم تطويره بناء على معايير وبنود وشروط موضوعة من قبل المفوضية السامية للأمم المتحدة لشؤون اللاجئين UNHCR. حيث يهدف إلى تتبع عملية تقديم الخدمات للمستفيدين المسجلين ضمن هذه المنظمة عبر مراكزها المجتمعية، بحيث يمكن التحقق من بياناتهم والتأكد من عدم استفادتهم من خدمات مسبقة ضمن المراكز. وأجريت هذه الدراسة ضمن مكتب تقانة المعلومات في جمعية البر[2]، المكون من قسمين، الأول قسم الدعم التقني والثاني قسم الحلول التقنية والبرمجية المختص بتطوير البرمجيات، ويعمل هذان القسمان على تلبية وتخديم جميع أقسام ونشاطات الجمعية المتعلقة بمتطلباتهم التقنية كالصيانة وتطوير البرامج بناء على قيود وتعليمات وقوانين المنظمات الداعمة للجمعية.

- **عمل على هذا المشروع:** مدير المشروع الذي قام بالتنسيق بين المبرمجين ومسؤول التقانة من UNHCR، 4 مبرمجون، وتقنيان من قسم الدعم التقني اللذان قاما بإعداد المخدم الخاص بنشر التطبيق وتهيئته. بالإضافة إلى أن كل مبرمج لعب أكثر من دور في عملية التطوير كالتحقيق والاختبار. وقام مدير المشروع بالمشاركة ببعض أعمال البرمجة والاختبار اليدوي قبل النشر للمستخدمين، وهو فعلياً الباحث الواضع لهذه الدراسة.

- **أسلوب العمل:** اتُبع أسلوب المنهجيات السريعة Agile. وعقدت مجموعة من الاجتماعات بين فريق العمل ومسؤول تقانة UNHCR للمناقشة والوصول إلى فهم أعمق للمتطلبات بناء على توصيف UNHCR وتحديد قصص المستخدمين. جرت الاجتماعات على عدة مستويات بناء على المناصب الوظيفية للمستخدمين. في كل دورة للتطوير sprint لمعرفة المتطلبات الأكثر أهمية لكل دورة.

سجلت البيانات ورقياً ونوقشت لاحقاً ضمن الفريق، وحدد موعد بدء وانتهاء كل Sprint، وعند الانتهاء كانت تناقش نتائجه وتراجع مع مدير المشروع ومنسق UNHCR.

[2] جمعية خيرية غير ربحية تقع في سوريا حمص

عند الانتهاء من إضافة أو تطوير أي ميزة يقوم المبرمج المسؤول عنها بعملية دمج العمل مع المشروع باستخدام github ومعالجة التضاربات الناتجة يدوياً، لاحقاً يقوم المبرمجون الآخرون بعملية اختبار يدوية أيضاً. وعند وجود أخطاء يتم اخبار المبرمج بذلك باتصال هاتفي أو عند الاجتماع التالي. لم يتم نشر أي ميزة إلا للمستخدمين الخبراء القادرين على التعامل مع الأخطاء.

بعد الانتهاء من التطوير وضع النظام تحت الاختبار بيتا وتم تدريب المستخدمين عليه، وسجلت مراجعتهم بشكل يدوي أيضاً لإجراء التحسينات بناء عليها. استغرق العمل ما يقارب ستة أشهر بين بدء أول اجتماع وعملية تسليم المنتج الأولي للزبائن، وكان متوسط عدد ساعات العمل اليومي 6 ساعات.

- **الأدوات:** استخدمت الأدوات التالية الضرورية لعملية التطوير:
- أدوات فريق التطوير وهي:
  - بيئة التطوير Visual Studio 2015 Community Edition .
  - لغة البرمجة Asp.net MVC 5.
  - نظام إدارة قواعد البيانات Microsoft Sql Server 2014 Express Edition.
  - نظام إدارة الشيفرة المصدرية Github.

أدوات فريق الدعم التقني وهي:

- تهيئة المخدم بتنصيب Windows Server 2012.
- مخدم IIS Server 8.x.

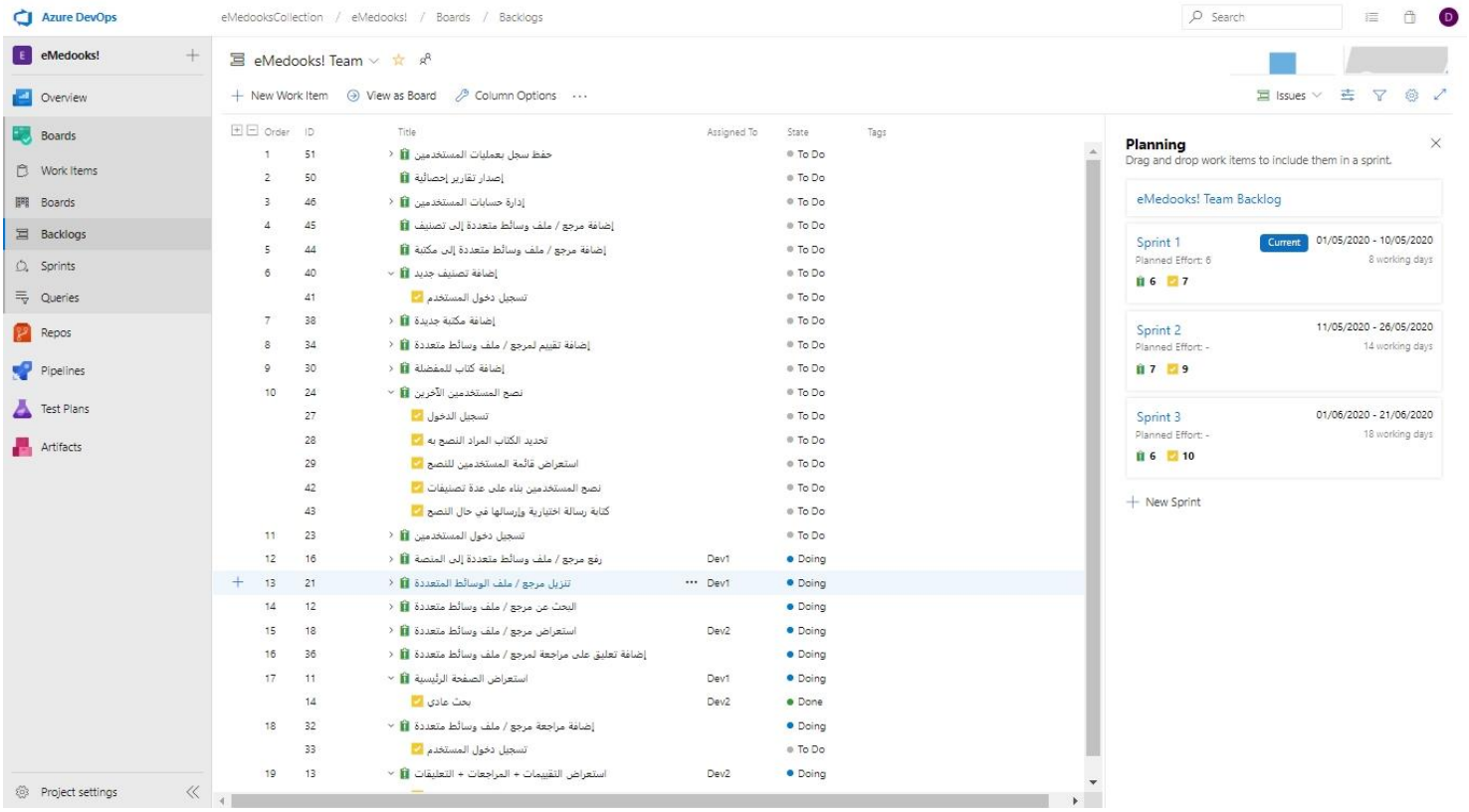
2.3.5 دراسة الحالة الثانية: المكتبة الالكترونية في كلية الطب البشري – جامعة البعث

تم دراسة الحالة الثانية بالتعاون مع شعبة المعلوماتية الخاصة بكلية الطب البشري في جامعة البعث، وهي مسؤولة عن متابعة جميع التجهيزات الحاسوبية وأداء الشبكة في الكلية، بالإضافة إلى تنصيب البرمجيات الضرورية للمستخدمين.

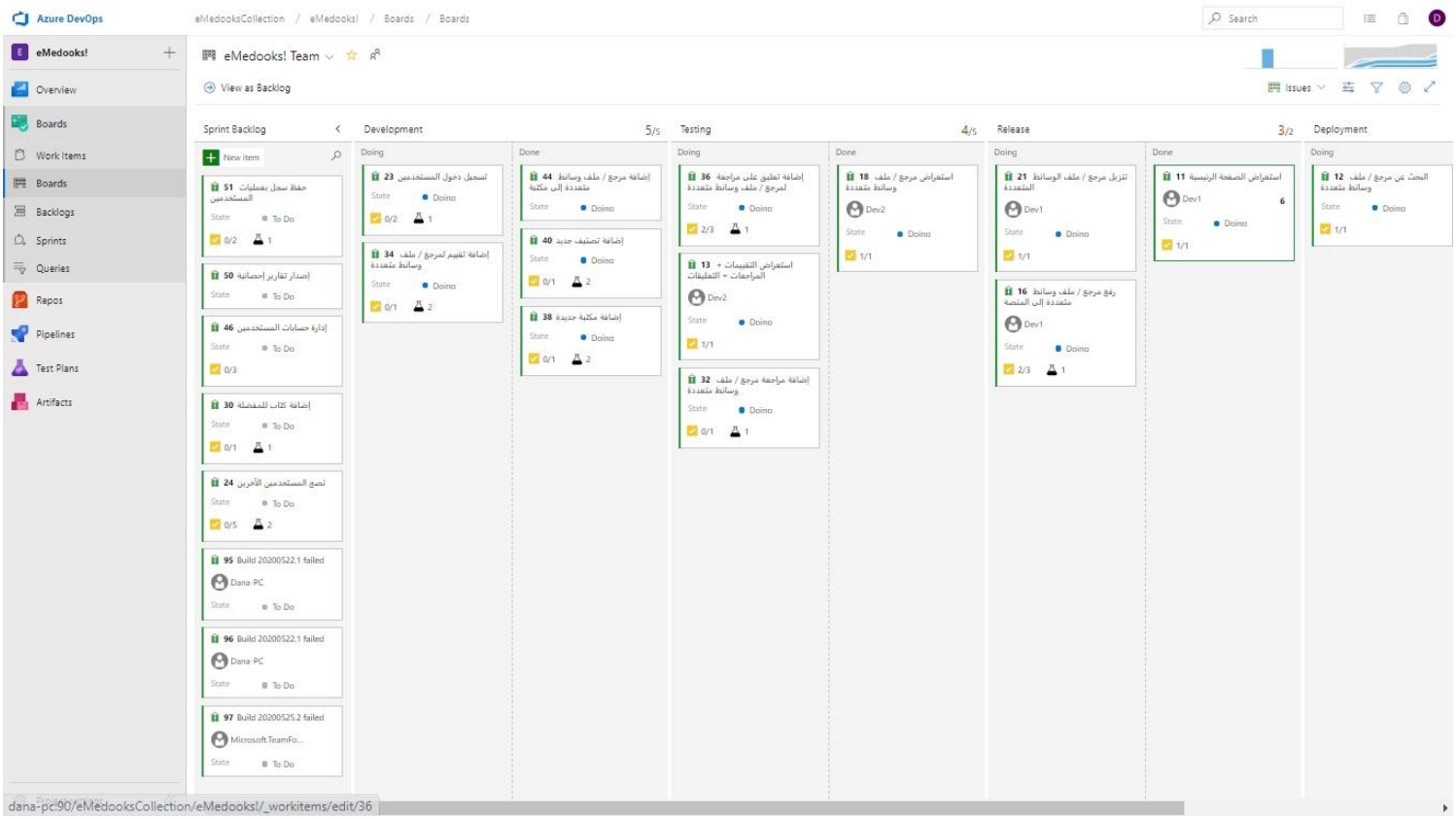
تم وضع دراسة خاصة بالتطبيق بالتعاون بين مهندسي الشعبة، ولعب باحث هذه الدراسة دور مدير المشروع. التطبيق المطلوب كان عبارة عن تطبيق وب بني بهدف جعله منصة تفاعلية لإدارة مكتبة الكترونية مرتبطة بشبكة داخلية ضمن جامعة البعث، قادرة على تلبية احتياجات الطلاب في

كلية الطب البشري، لتأمين الوصول السريع إلى المراجع العلمية التي يحتاجونها، من خلال تطوير واجهة مستخدم بسيطة وسهلة الاستخدام، وتخصيصه بشكل متوافق مع شعبة المكتبة الالكترونية في كلية الطب البشري، وبني بشكل قابل للتوسعة ليشمل كافة كليات ومعاهد الجامعة لاحقاً.

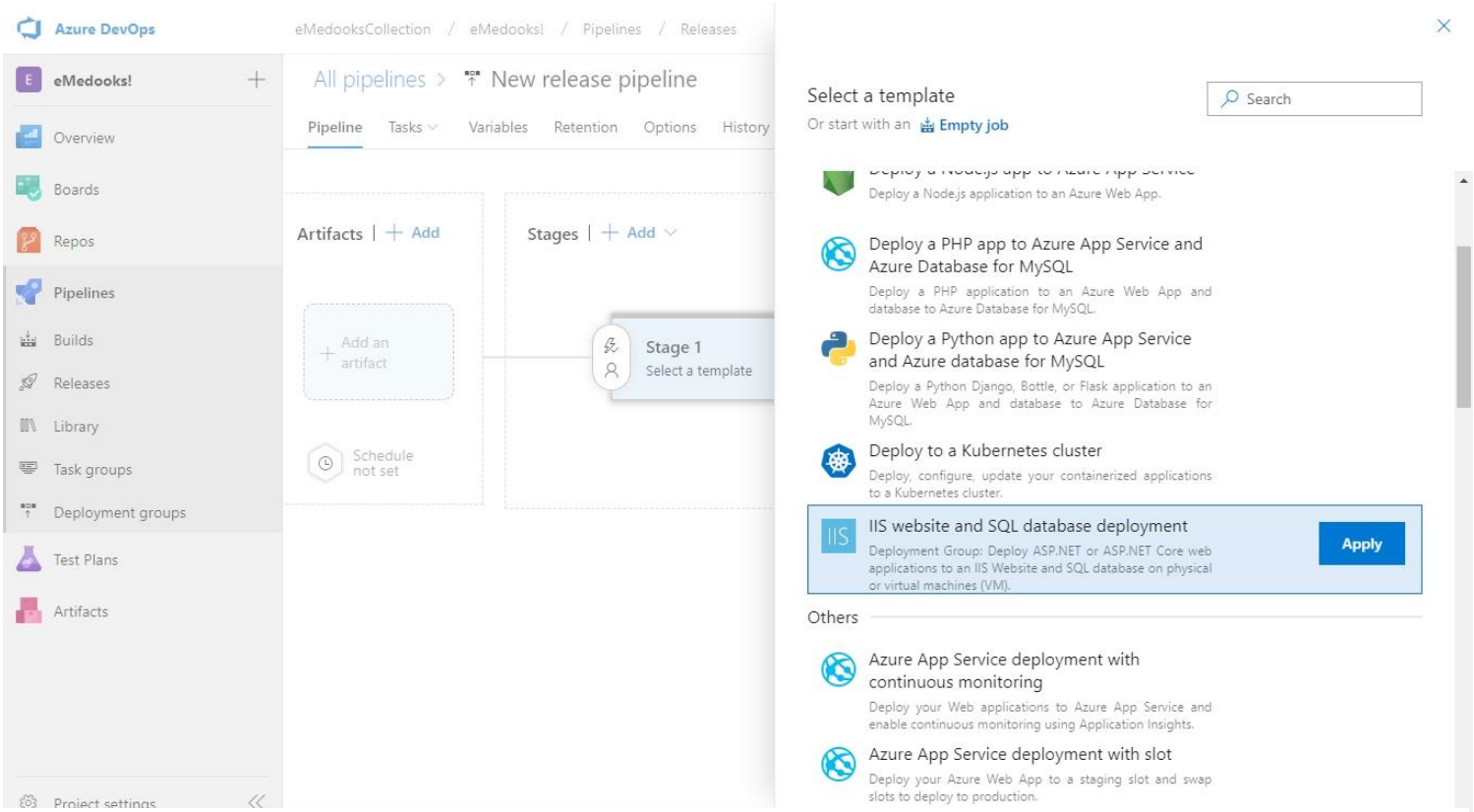
- **عمل على هذا المشروع:** شعبة المعلوماتية، وبسبب نقص في الكادر تم تقسيم العمل بشكل تسلسلي بدلاً من متوازي بحيث تم تقسيمه على أساس وجود 3 مبرمجين، مختبر، مراقب، داعم تقني.
- **أسلوب العمل:** طورت المنصة وفقاً لدمج عدة منهجيات وهي Scrum مع Kanban مع DevOps، وعمل فيها وفقاً للطريقة التالية: ضبطت مرحلة جمع المتطلبات وتنظيم قصص المستخدمين وفقاً لمنهجية السكرم Scrum، وأيضاً سارت الاجتماعات وفق لأسلوبها في تحديد أوقاتها وضرورتها وخصوصيتها وأية المواضيع تناقش. دونت قصص المستخدمين ضمن لوح الKanban الالكتروني المقدم من قبل Azure Boards، حيث استفدنا من ذلك في تنظيم المراحل المتبقية من تحليل وتصميم وتطوير وصولاً إلى النشر ومتابعة ما تم إنجازه وما هو قيد الإنجاز من خلال نقل قصص المستخدمين وبطاقات العمل على اللوح وتحديد عدد المهام الممكن إنجازها في كل مرحلة Work In Progress WIP، وأضفنا بناء على منهجية الDevOps عملية المراقبة المستمرة للمنتج المنشور وذلك بهدف معرفة مدى رضا الزبائن عن المنتج، وهو ما اعتبرناه كاجتماع مراجعة الSprint الخاص بScrum.



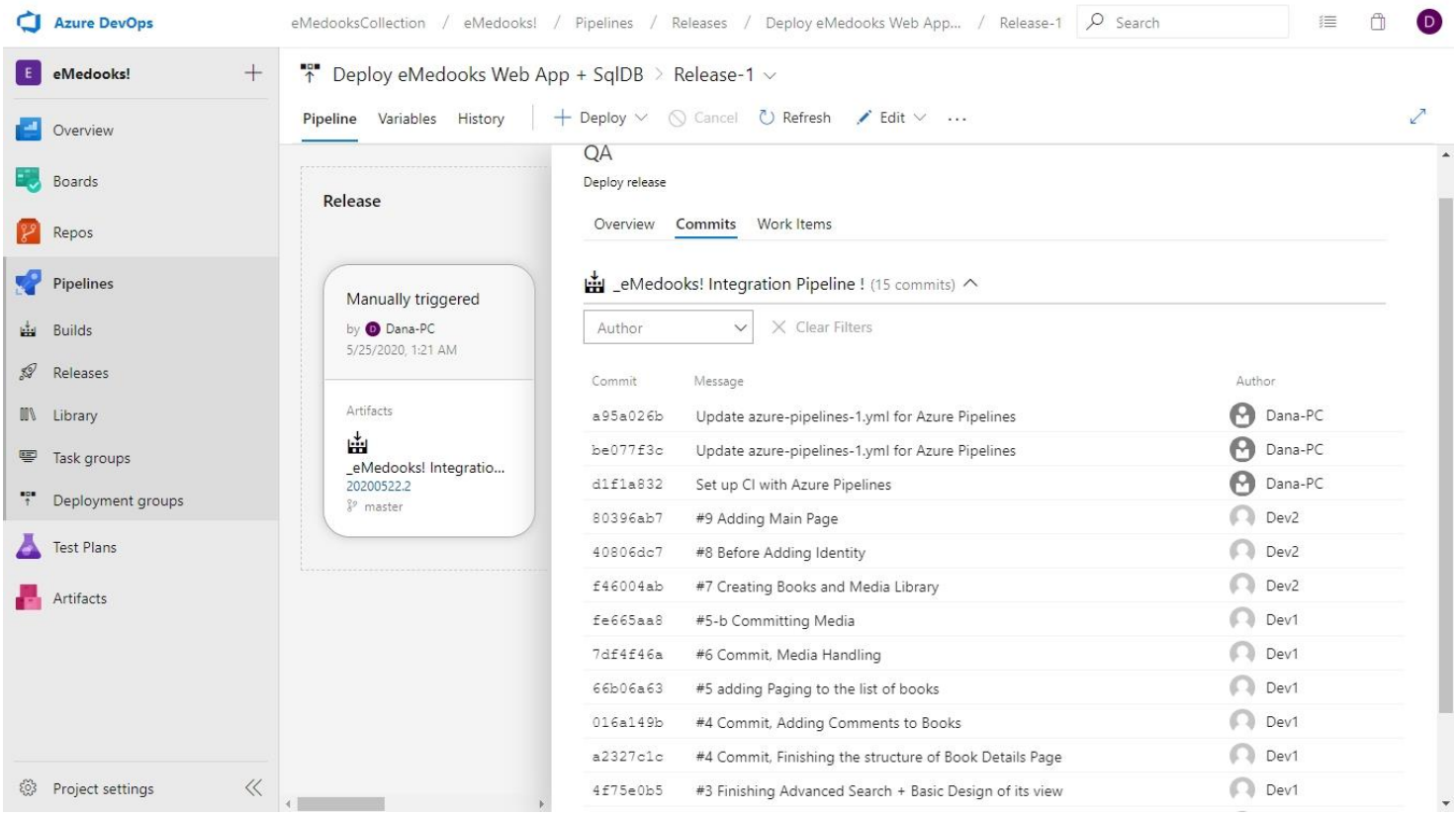
الشكل (5-12) المتطلبات الخاصة بتطبيق المكتبة الالكترونية، منظمة باستخدام Azure DevOps Server 2019



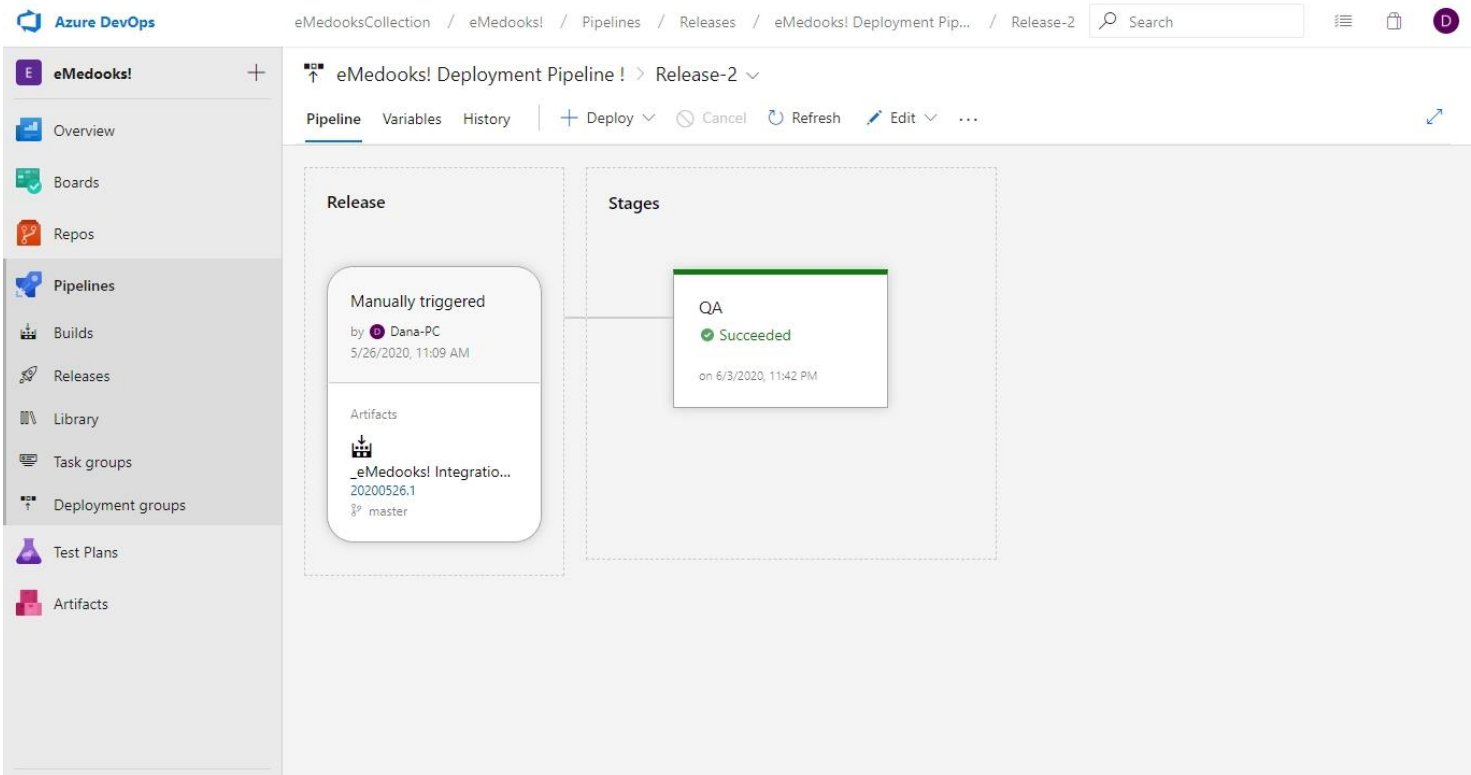
الشكل (5-13) قصص المستخدمين منظمة ضمن Azure Boards



الشكل (5-14) اختيار الخدمة المناسبة لعملية نشر إصدار جديد من التطبيق باستخدام Azure Pipelines

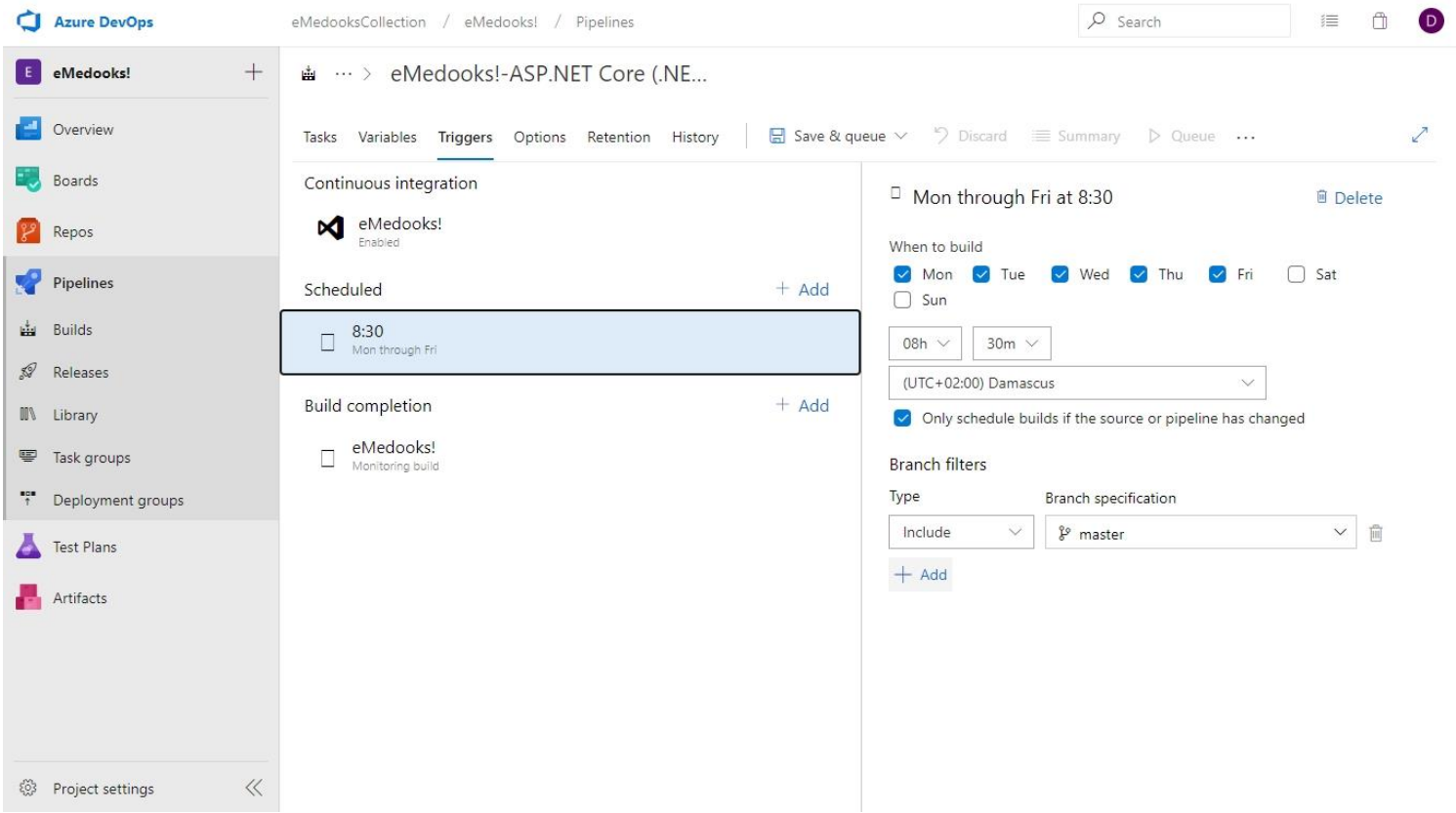


الشكل (5-15) اختيار الميزات التي ستنشر في الإصدار الحالي



الشكل (5-16) نجاح عملية الإصدار

حُكِمت العملية بشكل كامل بإحدى أهم المفاهيم التي تتبناها الDevOps وهي الأتمتة، حيث استخدمت التقنيات المقدمة من Azure DevOps Server 2019 بهدف أتمتة كل ما يمكن أتمتته،

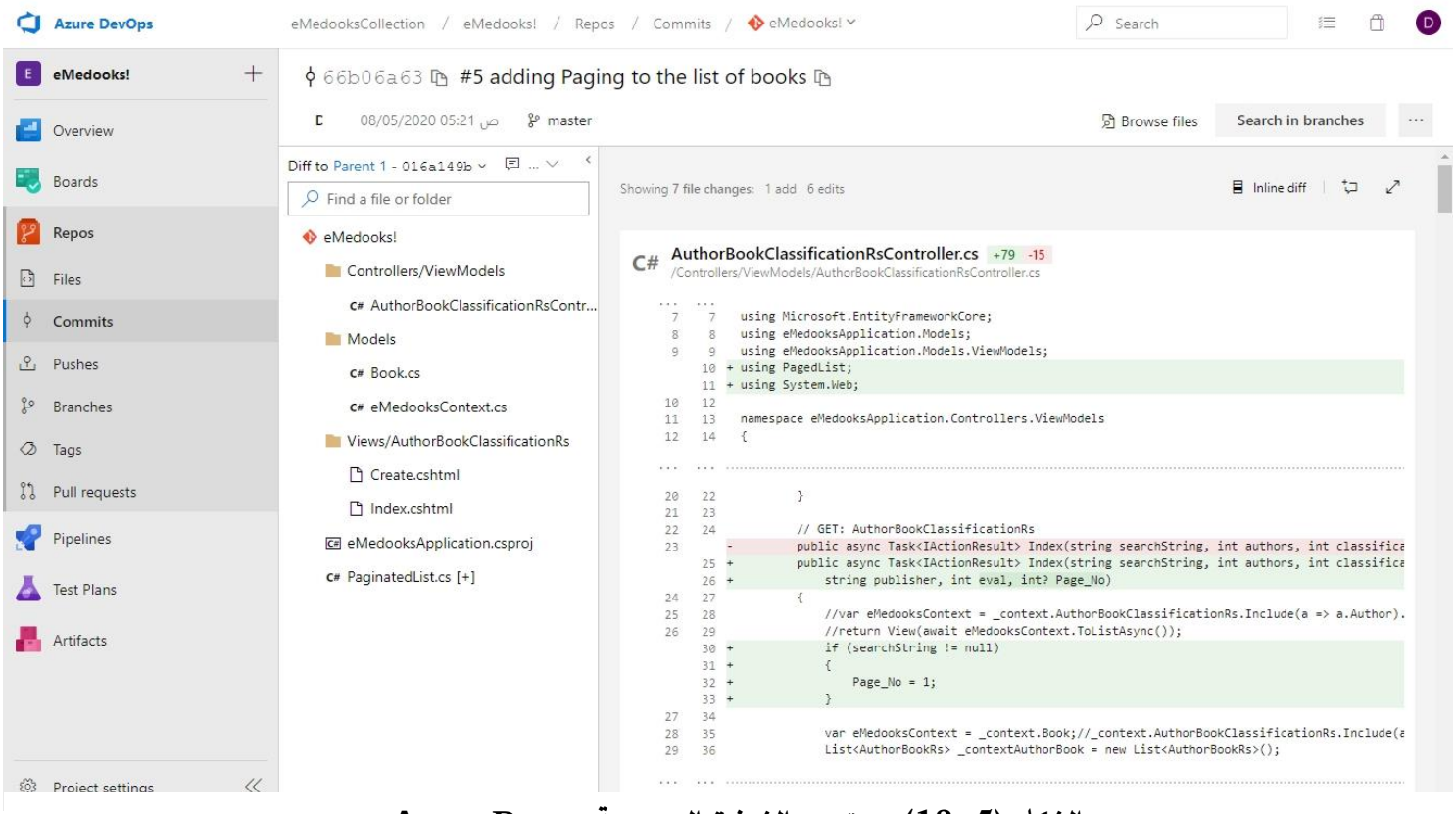


الشكل (5-17) ضبط إعدادات نشر الإصدارات الجديدة باستخدام Azure Pipelines

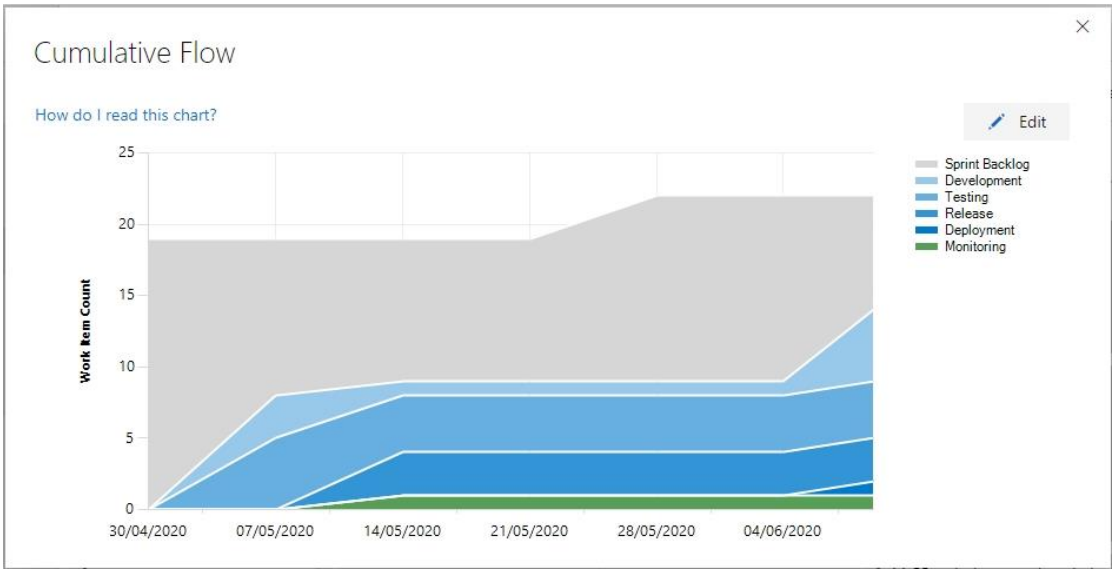
(حددت في الصورة عند كل عملية دمج شيفرة مصدرية إلى مستودع الشيفرة المصدرية الرئيسي يتم نشر إصدار جديد في تمام الساعة 8.30 صباحاً)

وتمت كل عمليات دمج أفرع الشيفرة المصدرية وإنشاء ملف البناء وإنشاء الإصدار وعملية النشر كلها وفق مسارات العمل المؤتمتة Automated Pipelines، من خلال Azure Pipelines.

اختلفت في هذه الحالة مدة كل Sprint، حيث كان بالإمكان نشر عدة إصدارات للمنتج إلى مجموعة أجهزة مخصصة للاختبار ضمن الدورة الواحدة، أي لم يتم الانتظار من الانتهاء من الSprint بكل متطلباته حتى يتم تفعيل الميزات الجديدة. استمر العمل على المشروع ما يقارب 6 أشهر بدءاً من عملية جمع المتطلبات وصولاً إلى نشر آخر الميزات الجديدة لل Sprint الثالث، بمعدل 6 إلى 7 ساعات عمل يومية.



الشكل (5-18) مستودع الشيفرة المصدرية Azure Repos



الشكل (5-19) المخطط التراكمي لقياس عدد المهام المنجزة خلال الSprints

- **الأدوات:** استخدم Azure DevOps Server 2019 لإدارة كل عملية التطوير بما يقدمه من ميزات:
  - Azure Boards.
  - Azure Pipelines.

- Azure Repos.
- Azure Test Plans.
- Azure Artifacts.

واستخدم فريق المطورون الأدوات التالية لتطوير التطبيق:

- Visual Studio 2019 Enterprise Edition.
- مع قاعدة بيانات Microsoft SQL Server 2016 Enterprise Edition.
- نظام Windows Server 2012 لتهيئة المخدم.
- مخدم IIS Server 8.x.

خاتمة

قمنا في هذا الفصل باستعراض منهجية البحث التي استخدمناها لوضع إطار عمل DevOps – الذي قمنا بطرحه في الفصل الرابع– تحت الدراسة لمعرفة الأثر الذي يحدثه عند تطبيقه، وطرحنا مجموعة من الأسئلة التي ستساعدنا في استخلاص النتائج، كما قمنا بعرض وصف معمق عن التطبيقين الذين وضعناهما تحت الدراسة، وشرحنا أهم الوظائف التي يقدمها كل منهما، بالإضافة إلى شرح أسلوب العمل المتبع في تطوير كل تطبيق مع الأدوات المستخدمة.

لاحقاً، سنقوم بالإجابة عن أسئلة البحث من خلال تحليل البيانات المجموعة من دراستي الحالة واستخلاص نتائج البحث.

الفصل السّادس

النتائج والمقترحات المستقبلية

مقدمة

سنقوم في هذا الفصل باستعراض النتائج المستخلصة من دراستي الحالة وخصوصاً تلك المتعلقة بإطار عمل DevOps المقترح، عن طريق الإجابة عن الأسئلة التي اعتمدنا عليها في صياغة الدراستين. كما سنقوم بعرض مجموعة من المقترحات للأعمال المستقبلية الممكن المتابعة فيها في بحث لاحق، منطلقين من النتائج التي توصلنا إليها في هذا البحث.

1.6 منهجية البحث

لتحليل النتائج المستخلصة من دراستي الحالة قمنا باستخدام تحليل السمات، والذي يقوم على تصنيف البيانات المستخلصة إلى سمات أو مجموعات، ومن ثم تطبيق التحليل المتقاطع لمعرفة نقاط التشابه والاختلاف بين الحالتين[7] .

2.6 الاستنتاجات المستخلصة من دراستي الحالة

سنقوم باستعراض أهم النتائج المستخرجة من دراستي الحالة الذين قمنا بشرحهما في الفصل الخامس بنتيجة استخراج السمات من البيانات (انظر الجدول (5-2))، وإجراء التحليل المتقاطع للحالتين، والذي هدف إلى إيجاد نقاط الالتقاء بين الحالتين وتحديد الممارسات التي بقيت نفسها بين أسلوبي العمل وما هو الاختلاف، والصعوبات التي واجهتهما وكيف حلت، بالاعتماد على الأسئلة التي قامت بوضع مجال وحدود هذا الجزء من الدراسة.

1- ما هي الصعوبات التي واجهت سير العمل في الحالتين؟ وماهي الحلول؟

- **في الحالة الأولى:** استطاع الفريق من خلال انتهاج منهجية سريعة من فهم المتطلبات الخاصة بالتطبيق بشكل أكثر دقة ولكن كانت الصعوبة بأنه نتيجة للاجتماع مع مستويات مختلفة من المستخدمين، حدثت تضاربات عديدة في المتطلبات. فكان لا بد من إيجاد حل لذلك من خلال ترتيب المتطلبات وتحديد الأهم وإقصاء ما يمكن إقصاءه، كما ظهرت مشكلة في أثناء النشر نتيجة عمل المبرمجين على أجهزة تختلف عن بيئة النشر، مما أدى إلى ظهور بعض المشكلات

أبرزها عدم عمل بعض الوظائف المطورة. ويعود السبب في ذلك إلى قيام الفريق التقني بتنصيب المكتبات والحزم والبرمجيات الأساسية في بيئة النشر دون التنسيق مع المطورين لمعرفة ما هي المكتبات الواجب تنصيبها، والأدوات المتناسبة مع العمل، كاختيار نظام إدارة قواعد البيانات المناسب. وواجهت عملية التطوير مشكلة أخرى عند إجراء التعديلات والدمج، خصوصاً أنها كانت كثيراً ما تتم بشكل يدوي وبعد ذلك يتم رفعها إلى فرع المشروع الرئيسي في github.

- **في الحالة الثانية:** تركزت الصعوبات في تبني أدوات جديدة كلياً بالنسبة للفريق واستخدامها لتنظيم العمل، ولكن فور تعلمها لوحظ مساهمتها الفعلية في تسريع العمل وزيادة الإنتاجية.

- **بمقاطعة العمل بين الحالتين** نجد:

1- أدى دمج أسلوبي العمل إلى تحسن كبير في عملية التطوير، وتلخصت الصعوبات في عملية جمع المتطلبات وفهمها وهو الأمر الذي يتطلب مهارات هندسة المتطلبات، بالإضافة إلى ضرورة وجود محلل نظام خبير، الذي بإمكانه المساهمة في زيادة القدرة على الحصول على متطلبات أدق بفترة زمنية أقل.

2- سرعت مفاهيم الأجيل عملية التطوير والتحقيق في الحالة الأولى، لكنها واجهت مشكلة في عملية دمج الشيفرة المصدرية والأفرع الخاصة بالمبرمجين وحتى عند استخدام Github نتيجة لمعالجة التضاربات يدوياً، الأمر الذي حلَّ في الحالة الثانية من خلال Azure Repos الذي ربط بال Visual Studio 2019 وكان يقوم بتحديد مواقع التعديلات وإعطاء عدة خيارات للمبرمج للقيام بعملية الدمج.

3- سببت عمليات التطوير والاختبار اليدوية كل فرد على جهازه في ظهور العديد من المشكلات التي لم تكن لتظهر لو تم إعداد البيئة لجميع المستخدمين من قبل جهة واحدة، وساهمت الأدوات الداعمة المستخدمة في الحالة الثانية بالإضافة للتعاون والتواصل بين مختلف أعضاء الفريق من التخفيف من أعباء هذه المشكلة.

4- واجهت عملية النشر مشكلة كبيرة أيضاً في الحالة الأولى حيث كانت تأخذ وقتاً طويلاً، وكان التطبيق ينصب دائماً خارج ساعات العمل، لعدم إعاقة أو إيقاف العمل للمستخدمين، وحلت هذه المشكلة في الحالة الثانية من خلال استخدام مسارات النشر المؤتمتة التي جعلت من هذه العملية عمليةً مجدولة زمنياً تتم بشكل أتوماتيكي عند ضبط الإعدادات الخاصة بها.

2- بناء على الحلول المستنتجة ما هو الاختلاف الذي طرأ ضمن المنظمة من ناحية التعاون وتنظيم العمل ضمن الفريق؟

وزعت المهام في الحالة الاولى على 4 مبرمجين وكانت عملية إعداد البيئة وتجهيزها منوطة بفريق آخر منفصل وليس لديه خبرة بالبرمجة. وقد سبب هذا الأمر مشكلات نتيجة عدم التنسيق بين الفريقين حيث غالباً ما وجد نقص في المكتبات الواجب تواجدها في بيئة النشر، أو تم تنصيب إصدارات مختلفة لبعض البرمجيات الضرورية لعمل التطبيق، مما أدى ذلك في كثير من الأحيان إلى وضع التطبيق في موضع الفشل.

في حين قمنا في الحالة الثانية وبناء على ما ظهر من مشاكل في التنسيق بين الفريقين بدمج كل أطراف العمل ووضعهم في صورة الهدف المراد تحقيقه، وعزز استخدامنا لAzure Board التواصل بين جميع الأعضاء، حيث استخدم لعرض جميع المتطلبات من قصص مستخدمين وحالات اختبار ومتطلبات تقنية، وأيضا ساهم في تحويل عملية التطوير إلى عملية مرئية لكافة الأطراف – الأمر الذي يعتبر حلاً لمشكلة حقيقية تواجه مجتمع تطوير البرمجيات–، ومفهومة وقابلة للتعديل والمناقشة بشكل أعمق وأوضح وأكثر تركيزاً على الهدف النهائي للعمل.

أصبحت جميع الأطراف قادرة على فهم عمل بعضها البعض وإبداء رأيها في المشكلات والتوصل الى حلول بشكل جماعي، بطرق تناسب جميع الأطراف وبطريقة أسرع وبوقت أقل. وساهمت عملية إسناد المهام وقصص المستخدمين وربطها بالأفراد عن طريق اللوح الالكتروني في معرفة المسؤول عن كل واحدة منها، الأمر الذي انعكس بشكل إيجابي على الفريق حيث أصبح كل خطأ يحل من قبل الشخص المسؤول عنه بشكل كامل، فخُفِفَ التوتر الناجم عن ظهور هذا النوع من المشكلات، وانتهت عملية إلقاء اللوم بين الأفراد عند ظهور أمر مشابه.

3- ما هي التغيرات التي طرأت على مخرجات كل مرحلة من مراحل عملية التطوير؟

أضافت عملية دمج فرق العمل وتعزيز مفهوم قابلية رؤية تقدم العمل باستخدام لوح Azure من تسريع جميع مراحل العمل. لكن وكما ذكرنا سابقاً أن عملية جمع المتطلبات بقيت نفسها في الحالتين، ولكن الذي اختلف هو طريقة تنظيم المتطلبات باستخدام Azure Board، حيث قلل التنظيم الالكتروني للعمل من خلال Azure DevOps Server 2019 من الوقت الذي احتاجته كل مرحلة، (انظر الجدول (6-1))، ومن زمن الاجتماعات. كما تحسن معدل تنفيذ الاختبارات نتيجة السرعة في اكتشاف الأخطاء والمسؤول عنها. كما قلت نسبة ضياع المعلومات بما يقارب 90% نتيجة لرفعها جميعها وتسجيلها إلى المخدم المركزي وإدارتها منه. كما أضافت عملية مراجعة كل sprint وتحسين وتقليل الهدر –إحدى مبادئ المنهجيات Lean– من سرعة الsprint وإنجازه.

الجدول (6-1)

المدة الخاصة بكل مرحلة من مراحل التطوير

جانب الدراسة	دراسة الحالة الأولى "نظام سند الموزع"	دراسة الحالة الثانية "المكتبة الالكترونية"
عملية جمع المتطلبات	استغرق ما يقارب شهر للحصول على فهم شامل لمفهوم المشروع.	من شهر إلى شهرين للحصول على توصيف دقيق للمشروع واعتماده "إنجاز العمل الورقي".
التحليل والتحقيق	تقسيم العمل إلى مراحل sprints يتراوح كل منها لمدة شهر، مع تحديد نقاط الإنجاز لكل مرحلة. في النهاية حددت 3 sprints.	حدد في الاجتماع عدد الsprints المطلوبة، وطول ال sprint الأول كان شهر، ويتم تعديل مدة كل sprint لاحقاً باجتماع المراجعة من خلال محاولات تحسين العملية وإزالة الهدر.
الاختبار	يحدث بعد الانتهاء من الSprint وتتراوح مدة الاختبار ألفا ما يقارب أسبوعين، والاختبار بيتا ما يقارب أسبوع إلى أسبوعين.	دمج ضمن عملية التحقيق، حيث أسند الدور إلى مختبر.
النشر	كان يتم بعد عملية تخطيط واختبار، وكانت الموافقة يدوية حيث يأخذ ما يقارب الشهر للموافقة، "الموافقة من المركز الرئيسي ل UNHCR في دمشق".	تمت أتمتة العملية من خلال Azure DevOps Server، والقبول من قبل الشخص المسؤول عن مسار عمل النشر وفور عملية التثبيت يتم في اليوم التالي نشر الميزة الجديدة، والتراجع عنها في حال ظهور مشكلات.

1.2.6 المناقشة

بعد أن قمنا في الفصل الرابع بوضع إطار عمل DevOps، توجهنا نحو إيجاد طريقة لوضع إطار العمل قيد التطبيق، لمعرفة مدى فعاليته في إدارة عملية التطوير البرمجي، لذلك اخترنا تطوير تطبيق وب بالاعتماد على إطار العمل هذا، ومقارنة أسلوب العمل والنتائج مع عملية تطوير تطبيق مشابه له بالاعتماد على منهجية تطوير سريعة Agile.

بناء على الدراسة والتحليل بعد الانتهاء من الحالتين، وجدنا أنه من الواجب أن تكتسب الفرق المعرفة بمفاهيم الAgile وممارساتها بالإضافة إلى اكتساب المعرفة الخاصة بمفاهيم التحسين المستمر للعمليات وأدوات DevOps، وتحفيز المنظمات على انتهاج هذا النهج كسلوك، للمساهمة في تحسين عمليات تطوير البرمجيات وزيادة رضا الزبون عن المنتج النهائي.

وجدنا أيضاً أنه على الرغم من الفوائد التي تجلبها DevOps وتشجعيها إلى أتمتة كل ما يمكن أتمتته إلا أنه يتبين أن الاختبار اليدوي يلعب دوراً كبيراً في عملية اكتشاف الأخطاء، مما يجعلنا غير قادرين على إقصاءه من معادلة التطوير. لذلك كلما تم دمج اختبار المستخدمين للمنتج بمراحل أولية وبشكل أسرع كلما زاد ذلك من جودة المنتج.

3.6 نتائج البحث

انتهى البحث بالنتائج التالية:

1. يسهِّل استخدام الأدوات المناسبة والمتوافقة مع بعضها والقابلة للتوسعة والتكامل من انتهاج منهجية ال DevOps وتحسين عملية إدارة مختلف مراحل عميلة التطوير البرمجي، كما تساهم آلية تنظيم الفريق وانتهاجهم منهجية Agile مسبقاً في تحقيق DevOps.
2. تحول DevOps عملية التطوير إلى عملية مرئية، بحيث تسمح الأدوات المناسبة من استعراض قصص المستخدمين، والاطلاع على مسارات العمل المؤتمتة ومعرفة نجاحها، ومواضع عنق الزجاجة ونقاط التأخير في العملية، وأسباب فشلها والسبب وراء ذلك. ويفيد استخدام مستودع للشيفرة المصدرية مع إمكانية تتبع مثبّته وساعة تثبيته، في إعطاء نظرة واضحة عن مدى التقدم في العمل.
3. بناء على دراستي الحالة استطعنا الحصول على تحسين وسرعة في العمل عند استخدام إطار العمل الجديد، مقابل استخدام منهجية سريعة فقط، (أول دورة في الحالة الثانية استغرقت شهراً، وفي الدورة الثانية تقلصت المدة إلى النصف، والثالثة إلى الربع ورصد الأمر من خلال إحصائيات Azure Server التي توفرها).
4. لا تخفف DevOps من الأعباء أو الوقت المبذول في عملية جمع المتطلبات ولكنها تساعد في تنظيمها وحفظها من خلال استخدام الأدوات، حيث ساهمت في زيادة مدى فهمها ودقتها لأنها مرئية من قبل كافة الأطراف بالإضافة لإمكانية التعديل المباشر السريع والمتزامن مما يؤمن الوصول إلى النتيجة التي تقدم القيمة الحقيقية للزبون.
5. واجهنا مشكلة من ناحية الهدر في الوقت بسبب الاجتماعات الكثيرة ضمن الحالتين، الأمر الذي أصبح من الممكن تلافيه من خلال تحويل الاجتماعات إلى اجتماعات افتراضية Online باستخدام وسائل الاتصال الفيديوية.
6. لا تستطيع الDevOps التدخل بالأعمال المتعلقة بالجهود الذهنية للعاملين وفقها كالعمليات البرمجية وعمليات جمع المتطلبات، حيث يعود الأمر في هذين الأمرين لخبرة المبرمجين ومحللي النظام.

7. توصلنا إلى وضع تعريف خاص بنا، بناء على التجربة العملية لمنهجية DevOps وهو كالتالي:

"أسلوب عمل إداري خاص بالمنظمات البرمجية، يهدف إلى تنظيم العمل فيها بالتعامل معها ككتلة واحدة بهدف هدم جميع الحواجز في عملية التطوير البرمجي وبناء منظومة تطوير قائمة على الحوار والتعاون بين مختلف الأطراف، وإلى تخفيف الأعباء الملقاة على الفرق المختلفة من خلال حل إحدى أكبر مشاكل هندسة البرمجيات وهي غير المرئية وتحويلها كذلك، من خلال الأتمتة. كما تعتمد على الأدوات لتحويل العمليات التكرارية إلى عمليات مؤتمتة مما يؤدي إلى زيادة الإنتاجية والكفاءة وتقليل الوقت".

نهاية، وبعد ما قمنا به من عمل نجد أننا نحتاج أكثر إلى التعمق بأفضل الممارسات لهندسة البرمجيات ودمجها بعملية الDevOps لإمكانية تحسين إطار العمل ليشمل جوانب الحفاظ على جودة البرمجية المطورة، والتي تواجه تحدياً كبيراً مقابل عمليات التطوير السريعة الحاصلة، وأيضاً لزيادة أمان البرمجيات المطورة وتوسعة الاختبارات لتشمل الاختبارات الأمنية الدفاعية.

4.6 المقترحات المستقبلية

قمنا ضمن البحث الحالي بالتعمق بمفاهيم منهجيات تطوير البرمجيات المختلفة، واستطعنا من خلال ذلك فهم الأسباب التي أدت إلى ظهور منهجية DevOps، المنهجية الأكثر رواجاً حالياً في عالم تطوير البرمجيات، والتي مازالت أيضاً قيد الدراسة والتطوير في الجانب الأكاديمي، لذلك وبناء على ما استنتجناه من بحثنا، نجد أنه يمكننا في خطوات لاحقة الاستمرار في التعمق بهذه المنهجية ورفد دراستنا بجوانب بحثية أخرى كما يلي:

1. التعمق بدراسة أفضل ممارسات هندسة البرمجيات والبنى والمقاييس الخاصة بضمان الجودة وإيجاد آلية لدمجها ضمن أسلوب العمل وفقاً ل DevOps.
2. دراسة الممارسات الأمنية المثلى الخاصة بعمليات التطوير البرمجي وإضافتها إلى منهجية DevOps، وقد ظهر حديثاً تيار أكاديمي يبحث في هذا الجانب وحمل اسم DevSecOps.
3. إجراء **دراسة حالات مختلفة ومتعمقة** في مقاييس الجودة والاختبارات الأمنية واستخلاص النتائج، التي تدعم الجانب الأكاديمي المتعلق بمنهجية DevOps.

خاتمة

قمنا في هذا الفصل باستعراض النتائج المستخلصة من البحث، كما قمنا بعرض مجموعة من النقاط التي من الممكن أن يتم العمل عليها لاحقاً ضمن أبحاث علمية يمكن تطبيقها واستخلاص نتائجها، للحصول على ممارسات ونتائج أفضل عند تطبيق منهجية DevOps في عملية التطوير البرمجي.

قائمة المصطلحات

List of Terms

A

Administrator	مدير النظام
Agile Development Methodology	منهجية التطوير السريعة
Agile Manifesto	إعلان الأجيل
Analysis	التحليل
Approach	مقاربة
Automation	الأتمتة
Automated Pipeline	مسار العمل المؤتمت
Automated Testing	الاختبارات المؤتمتة

B

Bugs	الخطأ البرمجي
Business Analysist	محلل النظم
Bottle Neck	عنق الزجاجة

C

Case Study	دراسة حالة
Cloud Computing	الحوسبة السحابية
Code	الشيفرة المصدرية
Code Repository	مستودع الشيفرة المصدرية
Continuous Delivery	التوصيل المستمر
Continuous Deployment	النشر المستمر
Continuous Development	التطوير المستمر
Continuous Integration	التكامل المستمر
Computer Science	علوم الحاسب
Containers	الحاويات

D

Defects	الأخطاء
Deployment	النشر
Developer	المطور
Development	التطوير

DevOps	منهجية تطوير DevOps
E	
Exploratory Testing	الاختبار الاستطلاعي
Extreme Programming	البرمجة المتطرفة
F	
Framework	إطار عمل
I	
Information Technology	تقنية المعلومات
Inherited code	البرمجيات الموروثة
Issues	المشكلات
Iterative Development	التطوير التكراري
K	
Kanban	منهجية الكان بان
L	
Lean Development Methodology	منهجية تطوير رشيقة
M	
Method	طريقة
Methodology	منهجية
Monitoring	المراقبة
O	
Open Source	مفتوح المصدر
Operations	فريق العمليات
P	
Pair Programming	البرمجة الثنائية
Pipelines	مسارات العمل
Plan-Driven Development	التطوير المقاد بخطة
Platform	منصة الكترونية
Programming	البرمجة
Product Backlog	سجل المنتج
Product Owner	مالك المنتج
Production Environment	بيئة النشر
Q	

Qualitative	النوعية
Quality Assurance	ضمان الجودة
Quantitative	الكمية

R

Repository	مستودع الشيفرة المصدرية
Requirements	المتطلبات
Requirements Engineering	هندسة المتطلبات

S

Scrum Master	مسؤول السكرم
Scrum Methodology	منهجية السكرم
Security	الأمن
Software Engineering	هندسة البرمجيات
System Specification	توصيف النظام
Spiral Model	النموذج الحلزوني
Sprint	الدورة الزمنية للتطوير في السكرم

T

Tools	الأدوات
Tracking	التعقب
Traditional Methodologies	المنهجيات التقليدية

U

User Interface	واجهة الاستخدام
User Stories	قصص المستخدمين

V

V-Model	نموذج V
Version Control System	نظام التحكم بالإصدار
Virtualization	الافتراضية

W

Waterfall Methodology	منهجية الشلال
Web Application	تطبيق وب
Windows Server	نظام ويندوز الخاص بالمخدمات

قائمة الاختصارات

List of Acronyms

(DevOps)	Development and Operation
(DevSecOps)	Development, Security and Operation
(MVC)	Model – View – Controller
(Repo)	Repository
(QA)	Quality Assurance
(SLR)	Systematic Literature Review
(SRS)	Software Requirement Specification
(TFS)	Team Foundation Server
(UI)	User Interface
(UX)	User Experience
(XP)	eXtreme Programming

المراجع

[1] SOMMERVILLE, Ian. Software engineering 9th Edition. ISBN-10, 2011, 137035152.
[2] Manifesto for Agile Software Development, (2020, March 17) https://agilemanifesto.org/.
[3] EBERT, Christof; ABRAHAMSSON, Pekka; OZA, Nilay. Lean software development. IEEE Software, 2012, 5: 22-25.
[4] Brown, D, 2015, WHAT IS DEVOPS?, accessed 17 May 2021,http://www.donovanbrown.com/post/what-is-devops.
[5] SIDDAWAY, Andy. What is a systematic literature review and how do I do one. University of Stirling, 2014, 1: 1-13.
[6] RUNESON, Per, et al. Case study research in software engineering: Guidelines and examples. John Wiley & Sons, 2012.
[7] CRUZES, Daniela S., et al. Case studies synthesis: a thematic, cross-case, and narrative synthesis worked example. Empirical Software Engineering, 2015, 20.6: 1634-1665.
[8] AL-AHMAD, Walid, et al. A taxonomy of an IT project failure: root causes. International Management Review, 2009, 5.1: 93-104.
[9] CHARETTE, Robert N. Why software fails. IEEE spectrum, 2005,42.9: 36.
[10] EVELEENS, J. Laurenz; VERHOEF, Chris. The rise and fall of the chaos report figures. IEEE software, 2010, 27.1: 30.
[11] BOEHM, Barry W. A spiral model of software development and enhancement. Computer, 1988, 5: 61-72.
[12] Principles behind the Agile Manifesto, last visit (2020, March 17) https://agilemanifesto.org/iso/ar/principles.html
[13] Panayotova, E, (2018, Dec 24), What Are the Pros and Cons of Extreme Programming (XP)?, Accessed (17 May 2021), https://simpleprogrammer.com/pros-cons-extreme-programming-xp/
[14] VLAANDEREN, Kevin, et al. The agile requirements refinery: Applying SCRUM principles to software product management. Information and software technology, 2011, 53.1: 58-70.
[15] CHO, Juyun. Issues and Challenges of agile software development with SCRUM. Issues in Information Systems, 2008, 9.2: 188-195.
[16] ABRAHAMSSON, Pekka, et al. Agile software development methods: Review and analysis. arXiv preprint arXiv:1709.08439, 2017.
[17] WILLIAMS, Laurie. Agile software development methodologies and practices. In: Advances in Computers. Elsevier, 2010. p. 1-44.
[18] Globalluxsoft, (2017, Oct 18), 5 Popular Software Development Models with their Pros and Cons, Accessed 17 May 2021,

https://medium.com/globalluxsoft/5-popular-software-development-models-with-their-pros-and-cons-12a486b569dc
[19] Poppendieck, M, (2015, June 5), Lean Software Development: The Backstory, Accessed 17 May 2021, http://www.leanessays.com/2015/06/lean-software-development-history.html
[20] IKONEN, Marko, et al. On the impact of kanban on software project work: An empirical case study investigation. In: 2011 16th IEEE International Conference on Engineering of Complex Computer Systems. IEEE, 2011. p. 305-314.
[21] KIROVSKA, Nevenka; KOCESKI, Saso. Usage of Kanban methodology at software development teams. Journal of Applied Economics and Business, 2015, 3.3: 25-34.
[22] AZOFF, Michael. DevOps: Advances in release management and automation. Technical report, Ovum, 2011.
[23] MEZAK, S, (2018, January 25), The Origins of DevOps: What's in a Name? Accessed 17 May 2021, https://devops.com/the-origins-of-devops-whats-in-a-name/
[24] Amazon, last visit (2020, March 17), What is DevOps? Accessed 17 May 2021, https://aws.amazon.com/ar/devops/what-is-devops/
[25] BOU GHANTOUS, G.; GILL, Asif. DevOps: Concepts, Practices, Tools, Benefits and Challenges. PACIS2017, 2017.
[26] CHEN, Lianping. Continuous delivery: Huge benefits, but challenges too. IEEE Software, 2015, 32.2: 50-54.
[27] DevOps, (2017, July 28), The Role of QA in the DevOps World, Accessed 17 May 2021, http://devops.com/role-qa-devops-world/
[28] ERICH, Floris; AMRIT, Chintan; DANEVA, Maya. Report: Devops literature review. University of Twente, Tech. Rep, 2014.
[29] ERICH, F. M. A.; AMRIT, Chintan; DANEVA, Maia. A qualitative study of DevOps usage in practice. Journal of Software: Evolution and Process, 2017, 29.6: e1885.
[30] FARROHA, B. S.; FARROHA, D. L. A framework for managing mission needs, compliance, and trust in the DevOps environment. In: 2014 IEEE Military Communications Conference. IEEE, 2014. p. 288-29.
[31] ELBERZHAGER, Frank, et al. From agile development to devops: going towards faster releases at high quality–experiences from an industrial context. In: International Conference on Software Quality. Springer, Cham, 2017. p. 33-44.
[32] SURESHCHANDRA, Kalpana; SHRINIVASAVADHANI, Jagadish. Moving from waterfall to agile. In: Agile 2008 conference. IEEE, 2008. p. 97-101.
[33] ABRAHAMSSON, Pekka, et al. Agile software development methods: Review and analysis. arXiv preprint arXiv:1709.08439, 2017.

[34] CAWLEY, Oisín; WANG, Xiaofeng; RICHARDSON, Ita. Lean/agile software development methodologies in regulated environments–state of the art. In: International Conference on Lean Enterprise Software and Systems. Springer, Berlin, Heidelberg, 2010. p. 31-36.

[35] WANG, Xiaofeng; CONBOY, Kieran; CAWLEY, Oisin. "Leagile" software development: An experience report analysis of the application of lean approaches in agile software development. Journal of Systems and Software, 2012, 85.6: 1287-1299.

[36] LWAKATARE, Lucy Ellen; KUVAJA, Pasi; OIVO, Markku. Relationship of DevOps to agile, lean and continuous deployment. In: International Conference on Product-Focused Software Process Improvement. Springer, Cham, 2016. p. 399-415.

[37] WANG, Cheng; LIU, Changling. Adopting DevOps in Agile: Challenges and Solutions. 2018.

[38] Hemon, A., Lyonnet, B., Rowe, F., & Fitzgerald, B. (2020). From agile to DevOps: Smart skills and collaborations. Information Systems Frontiers, 22(4), 927-945.

[39] MOUSAEI, M., & GANDOMANI, T. J. DEVOPS APPROACH AND LEAN THINKING IN AGILE SOFTWARE DEVELOPMENT: OPPORTUNITIES, ADVANTAGES, AND CHALLENGES.

[40] Govil, N., Saurakhia, M., Agnihotri, P., Shukla, S., & Agarwal, S. (2020, June). Analyzing the Behaviour of Applying Agile Methodologies & DevOps Culture in e-Commerce Web Application. In 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184) (pp. 899-902). IEEE.

[41] Maroukian, K., & Gulliver, S. R. (2020). Leading DevOps Practice and Principle Adoption. arXiv preprint arXiv:2008.10515.

[42] Saltz, J., & Sutherland, A. (2020, January). SKI: A New Agile Framework that Supports DevOps, Continuous Delivery, and Lean Hypothesis Testing. In Proceedings of the 53rd Hawaii International Conference on System Sciences.

[43] Cool J, (2019 March 5), Now available: Azure DevOps Server 2019, Accessed (17 May 2021), https://azure.microsoft.com/tr-tr/blog/now-available-azure-devops-server-2019/

[44] Microsoft Azure n.d., (Azure DevOps), Accessed (17 May 2021) https://azure.microsoft.com/en-us/services/devops/?nav=min

[45] Microsoft docs, (2019 March 6), What is Application Insights?, Accessed (17 May 2021), https://docs.microsoft.com/en-us/azure/azure-monitor/app/app-insights-overview

[46] Arvind, (2020 January 23), What is DevOps? DevOps Methodology, Principles & Stages Explained, Accessed (17 May 2021), https://www.edureka.co/blog/what-is-devops/

[47] BAXTER, Pamela, et al. Qualitative case study methodology: Study design and implementation for novice researchers. The qualitative report, 2008, 13.4: 544-559.
[48] Bookrunch, n.d., Top 10 eBooks Organizers, Accessed (17 May 2021) https://www.bookrunch.com/top/organizer/
[49] Jenic I, (2020 April 7), 6 best eBook management software for your PC, Accessed (17 May 2021) https://windowsreport.com/ebook-management-tools/
[50] Alfa eBooks Manager, Accessed (17 May 2021) https://www.alfaebooks.com/
[51] Calibre eBook Management, Accessed (17 May 2021) https://calibre-ebook.com/
[52] Library Thing, Accessed (17 May 2021) https://www.librarything.com/home
[53] GoodReads, Accessed (17 May 2021) https://www.goodreads.com/
[54] Mobile Read, n.d. Accessed (17 May 2021) https://www.mobileread.com/forums/showthread.php?t=309688
[55] Single White female Writer, n.d., Accessed (17 May 2021) https://singlewhitefemalewriter.wordpress.com/2017/04/18/the-benefits-of-a-goodreads-profile/
[56] Accessed (17 May 2021) https://jjbookblog.wordpress.com/2017/07/16/jo-talks-books-on-the-pros-and-cons-of-goodreads/
[57] Lopez H, (2021 May 13), 5 Best eBook Manager, Accessed (17 May 2021) https://www.epubor.com/best-5-ebook-manager.html

Syrian Arab Republic
AlBaath University
Faculty of Informatics Engineering
Department of Software Engineering
& Information Systems

Prospects and Challenges of Using DevOps Approach in Software Engineering Community

A thesis submitted in partial fulfilment of the requirements for the Master's Degree in Software Engineering and Information Systems

Prepared By

Eng. Dana Hafez Madwar

Supervised By

Dr. Alida Esper

An Assistant Professor at the Department of Software Engineering and Information Systems,
Faculty of Informatics Engineering
AlBaath University

1442 A.H - 2021 A.D