INTERNATIONALE HOCHSCHULE

Projekt: Software Engineering

Abstract

DLMCSPSE01_D

Art der Arbeit:	Portfolio
Kursbezeichnung:	DLMCSPSE01_D
Studiengang:	FS MAINF-120 Fernstudium Master of Science Informatik 120 ECTS
Datum:	21.10.2024
Name:	Johannes Fell
Matrikelnummer:	92205755
Name Tutorin:	Markus Kleffmann

Lesson Learned: Die Entwicklung von PlantPal

In der Entwicklung der PlantPal-Anwendung stellte die Balance zwischen technischer Komplexität und Benutzerfreundlichkeit eine zentrale Herausforderung dar. Dieser Bericht soll Einblicke in die Lektionen geben, die ich während der Umsetzung des Projekts gelernt habe, sowie die Hürden – sowohl technischer als auch organisatorischer Natur – die mich besonders gefordert haben. Das Projekt wurde am in der Kalenderwoche 33 gestartet und wird in der aktuellen Woche (KW 43) beendet. Die ursprünglich geplanten sieben Wochen für die Umsetzung konnten nicht eingehalten werden, wodurch sich die Dauer insgesamt auf zehn Wochen verlängerte. Dies lag unter anderem an meiner Vollzeitarbeit und privaten Umständen, die das Projekt verzögerten. Bei einem zukünftigen Vorhaben wäre eine detailliertere Zeitplanung mit mehr Pufferzeiten ratsam, um Verzögerungen besser abfedern zu können.

Zu Beginn entschied ich mich für das Kanban-Vorgehensmodell, um eine hohe Flexibilität und dynamische Anpassungen an die Anforderungen zu ermöglichen. Das Kanban-Modell erwies sich durch seinen kontinuierlichen Fluss als besonders geeignet für ein Ein-Personen-Projekt. Im Gegensatz zu fest strukturierten Iterationsmodellen wie Scrum erlaubte Kanban ständige Anpassungen und Re-Priorisierungen der Aufgaben, was bei unerwarteten Herausforderungen von großem Nutzen war. Diese Flexibilität half mir, die Entwicklung effizient zu gestalten und auftretende Probleme zu bewältigen. Eine dieser Herausforderungen war zum Beispiel die komplexe Konfiguration der JavaMail API, weshalb ich schließlich auf MailJet umstieg, um die Komplexität zu verringern. Die Verwendung von JavaMail erwies sich aufgrund der unterschiedlichen Konfigurationsanforderungen der E-Mail-Anbieter wie Yahoo oder Gmail als äußerst umständlich, insbesondere wegen zusätzlicher Sicherheitsmaßnahmen wie der Zwei-Faktor-Authentifizierung. Der Wechsel zu MailJet vereinfachte die Konfiguration erheblich, da alle notwendigen Parameter über eine zentrale Plattform bereitgestellt wurden, was die Integration deutlich erleichterte.

Ein weiteres zentrales Thema war die Wahl der Technologien und Tools. Java erwies sich als eine gute Grundlage für die Desktop-Anwendung, insbesondere weil ich bereits berufliche Erfahrung mit dieser Sprache hatte. Allerdings stellte sich JavaFX als problematischer heraus als erwartet, insbesondere wegen der Schwierigkeiten bei der Erstellung einer ausführbaren Datei. Mit Java 9 wurde das Java Packager Tool als veraltet deklariert, und ab Java 11 wurden die JavaFX-Bibliotheken aus dem JDK entfernt. In meinem Projekt, das auf Java 17 basiert, stellte sich dadurch die Bereitstellung von nativen ausführbaren Dateien als deutlich schwieriger heraus, da diese Bibliotheken manuell eingebunden werden mussten und die Erstellung eines nativen Installers komplexer wurde. Somit musste ich zusätzlich die Tools jpackage und jlink nutzen, um die nativ ausführbare Datei zu erstellen. Dennoch erhielt die Benutzeroberfläche dank JavaFX ansprechende visuelle Effekte, was die Benutzerfreundlichkeit erhöhte und wiederkehrende UI-Komponenten einfacher zu implementieren machte.

Gleichzeitig war SQLite eine strategisch sinnvolle Wahl, da es als leichtgewichtige und serverlose Datenbank ideale Voraussetzungen für eine Desktop-App mit lokaler Datenspeicherung bot. Dabei zeigte sich die Notwendigkeit eines effizienten Verbindungsmanagements, um die Performance der App zu gewährleisten – hier erwies sich der Einsatz von HikariCP als sehr nützlich.

Die Herausforderung, die Anwendung benutzerfreundlich zu gestalten, führte dazu, dass das erweiterte MVC-Entwurfsmuster für die Softwarestruktur eine entscheidende Rolle spielte. Diese Entwurfsentscheidung war besonders wertvoll, da sie die Trennung von Daten, Benutzeroberfläche und Logik ermöglichte und damit Übersichtlichkeit und Wartbarkeit sicherstellte, was für zukünftige Erweiterungen essenziell ist. Der zusätzliche Einsatz eines Service-Layers half dabei, die Controller schlank zu halten und die Geschäftslogik sauber in den Services zu kapseln, wodurch die Wiederverwendbarkeit erhöht wurde.

Ein wichtiges Feature, das besondere Aufmerksamkeit erforderte, war die Implementierung der Benachrichtigungsfunktion – sowohl in Form von In-App-Benachrichtigungen als auch durch die E-Mail-Benachrichtigungen via MailJet. Technisch erforderte dies die Einbindung zusätzlicher Bibliotheken und die Auseinandersetzung mit den Herausforderungen der API-Kommunikation, insbesondere beim Versenden von Erinnerungen zu den Pflegeaufgaben. Eine bewusste Entscheidung war, dass die E-Mails nicht automatisch, sondern nur manuell versendet werden, damit die Nutzer die volle Kontrolle über die Benachrichtigungen behalten und das Risiko von „spamartigen“ Mails minimiert wird.

Die Wissensdatenbank als Feature war ein weiterer Aspekt, bei dem der Nutzer im Mittelpunkt stand. Ziel war es, den Pflanzenbesitzern Informationen zu häufigen Pflanzenkrankheiten und Schädlingen zur Verfügung zu stellen. Hierbei achtete ich besonders darauf, dass die Inhalte übersichtlich präsentiert werden, leicht verständlich sind und auch weitere Informationen eingeholt werden können. Dabei zeigte sich, dass der Schwerpunkt weniger auf technischen Herausforderungen lag, sondern stärker auf der inhaltlichen Aufbereitung und benutzerfreundlichen Gestaltung, um den Nutzern einen echten Mehrwert zu bieten.

Zusammenfassend lässt sich sagen, dass die Entwicklung von PlantPal nicht nur ein tiefgehendes Verständnis für technische Aspekte und Tools erforderte, sondern auch wichtige Lektionen im Bereich der Nutzerorientierung und Prozessgestaltung ermöglichte. Die Verbindung von technologischen Entscheidungen, einem flexiblen Vorgehensmodell und einem klaren Fokus auf die Bedürfnisse der Nutzer trug wesentlich dazu bei, eine funktionale und benutzerfreundliche Anwendung zu entwickeln. Die wichtigste Erkenntnis war, dass die richtige Balance zwischen Flexibilität, Struktur und Benutzerfokus entscheidend für die Qualität eines Softwareprojekts ist. Für zukünftige Projekte ist es wichtig, die Zeitplanung noch präziser anzugehen und mehr Pufferzeiten einzuplanen, um besser auf unvorhergesehene Verzögerungen reagieren zu können.