Sprint 4 – Dokumentation

Sprint-Ziel

Im Sprint 4 lag der Fokus auf der Qualitätssicherung unserer Spring Boot Anwendung durch:

- Erweiterung des Product Backlogs,
- Entwicklung von Unit Tests,
- Integration dieser Tests in eine Build-Pipeline,
- Einrichtung eines Git Hooks (pre-push und pre-commit), um Build und Tests lokal zu automatisieren,
- Erweiterung von einige in-App Funktionen,
- sowie auf die Vorbereitung des nächsten Sprints inklusive Sprint-Planung.

1. Erweiterung des Product Backlogs

Unser Product Backlog wurde entsprechend der Scrum-Methodologie überarbeitet und erweitert. Dabei wurde besonders darauf geachtet, User Stories frühzeitig zu erfassen, damit die Sprint-Planung effizient durchgeführt werden kann.

Neu hinzugefügte relevante Einträge im Backlog wie zb:

- Als Entwickler:in möchte ich, dass meine Änderungen automatisch auf korrekte Formatierung und Funktion getestet werden, um versehentliche Fehler beim Push zu vermeiden.
- Als Team möchten wir sicherstellen, dass neue Features durch Unit Tests abgedeckt sind.

2. Entwicklung und Integration von Unit Tests

Es wurden gezielt JUnit-Tests für zentrale Komponenten der Anwendung erstellt, um die bisher implementierte Funktionalität (z. B. Controller, Services) automatisiert zu prüfen. Dabei haben wir auf sinnvolle Testabdeckung geachtet:
Beispielhafte getestete Komponenten:

- AppUserController

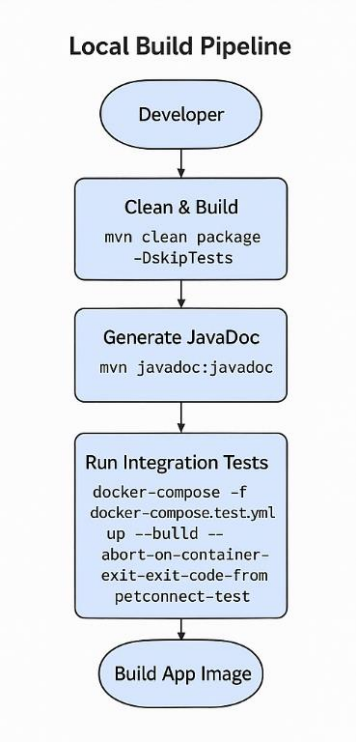
- AppUserService
- UserAvailabilityService

Die Tests wurden unter dem Maven-Testziel integriert und können durch den Befehl mvn test automatisch ausgeführt werden.

3. Integration in die Build-Pipeline (lokal & CI)

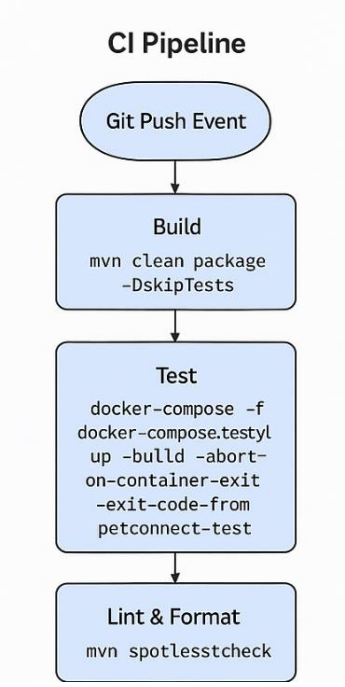
Die Tests wurden in zwei Pipelines integriert:

Lokale Build-Pipeline (bash.sh)



Hier wird die Testumgebung isoliert via Docker aufgebaut (docker-compose.test.yml) und in einem Maven-Container ausgeführt. Die Integrationstests laufen mit einem temporären PostgreSQL-Testcontainer.

CI-Pipeline (GitLab CI/CD)



Die Tests werden im CI-Umfeld innerhalb von Docker-Containern ausgeführt und stellen sicher, dass alle Abhängigkeiten korrekt geladen und getestet werden.

4. Git Pre-Push & Pre-Commit Hook mit Format- und Testprüfung

Ein lokaler **Git Pre-Commit Hook** wurde eingerichtet, um vor jedem Commit sicherzustellen, dass der Code formatiert und funktional ist:
Dieser Hook prüft mit dem Plugin spotless die korrekte Code-Formatierung. Wenn die Prüfung fehlschlägt, wird der commit blockiert und man wird zur Korrektur aufgefordert.

Ein **Git pre-push Hook** wurde lokal eingerichtet. Dieser Hook führt automatisch `mvn test -Dspring.profiles.active=test` aus, bevor ein Push auf das Remote-Repository erfolgt. Der Push wird blockiert, wenn Tests fehlschlagen.

5. Produkt Backlog

Backlog item	Einheit(Story Points)
Als Haustierbesitzer: in möchte ich ein Profil erstellen können, damit ich Infos über mein Tier und mich angeben kann.	**5**
Als Betreuer: in möchte ich mich registrieren und ein Profil anlegen, um meine Services wie Gassi gehen oder Urlaubsbetreuung anzubieten.	**5**
Suchfunktion für Tierbesitzer: innen, um passende Betreuer:innen zu finden	**3**
Filteroptionen für Betreuungsangebote, z. B. nur Notfallbetreuung oder nur für Hunde oder Katzen.	**3**
Bewertungsfunktion, damit Tierbesitzer:innen Feedback zu Betreuer:innen geben können	**3**
Kalenderintegration für Terminbuchungen, sodass Termine direkt vereinbart werden können	**15**
Schnellzugriff auf Notfallkontakte oder Notfallbetreuer:innen	**3**
Chat- oder Nachrichtensystem, um zwischen Besitzer:innen und Betreuer:innenkommunizieren zu können	**20+**
Profilbilder für Nutzer:innen, damit man weiß, mit wem man es zu tun hat	**8**
lokale Build-Pipeline erstellen, um Builds lokal zu testen.	**8**
Als Nutzer möchte ich mich sicher und vollständig ausloggen können, damit meine Sitzung geschützt ist.	**3**
Mobile Optimierung, also dass die App/Seite auch auf dem Handy gut funktioniert	**5**
Als Entwickler **möchte ich**, dass mein Code vor jedem Commit automatisch auf korrekte Formatierung geprüft wird, **damit** keine unformatierten Änderungen ins Repository gelangen.	**15**
Als Entwickler **möchte ich**, dass vor jedem Push alle Unit Tests automatisch ausgeführt werden, **damit** keine fehlerhafte Logik in das zentrale Repository gelangt.	**20**
Als Entwickler **möchte ich**, dass Tests lokal in einer isolierten Umgebung ausgeführt werden, **damit** sie zuverlässig und reproduzierbar sind.	**15**
Als Entwicklungsteam **möchten wir**, dass unsere Tests automatisch in der GitLab CI/CD-Pipeline ausgeführt werden, **damit** die Qualität auch bei automatisierten Deployments gewährleistet bleibt.	**15**