ingenieur
wissenschaften
htw saar

STL SYSTEMTECHNIKLABOR

Software Quality Engineering

Trinkspiel

Sorry, we got drunk while developing this project

Agenda - 45 min / 5 Personen = faire 9 min pro Person

1. Einleitung (Alex)
2. Das Projekt (
   a. Agil, iterativ,Rahmenbedingungen, erste Live Demo)(Alex)
   b. Teststrategien, Githüb) (Nico)
3. Frontend (Caro)
   a. zweite Live Demo
   b. Umsetzung
   c. Tests
4. Backend (Tristan & Mohammed)
   a. Umsetzung
   b. Tests
   c. Dritte Live DEMO
5. Lessons learned (Tristan & Mohammed)

PIM-SQE

Trinkspiel

Website

Virtuelles Kartenspiel

User

PIM-SQE

Trinkspiel

Live Demo

0 Einleitung und Grundlagen

1 Das Projekt

2 Frontend

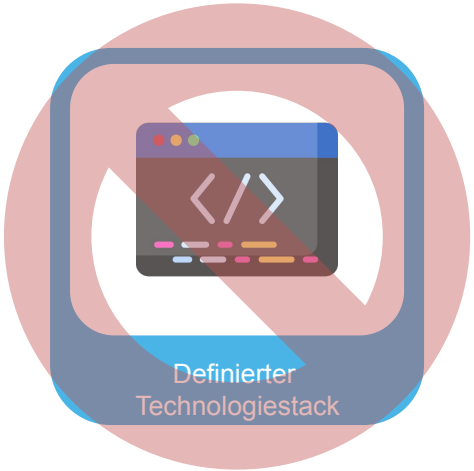
3 Backend

4 Lessons Learned

I. Rahmenbedingungen

Freepik, “Timeline icons", https://www.flaticon.com
Freepik, “users", https://www.flaticon.com/
/

II. Rahmenbedingungen – Kein Reales Projekt



II. Rahmenbedingungen – Kein Reales Projekt

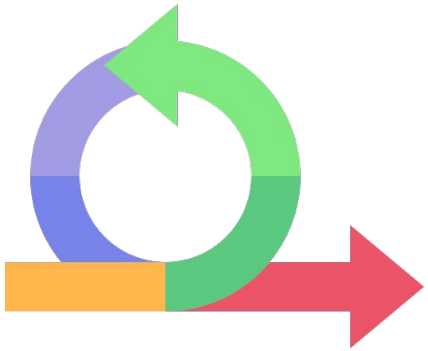
↳ machte esdas einfacher?
Nein!

- Übernahme aller Rollen
- Mangelndes Knowhow
- Keine definierten Ziele
- Folien boten nur im geringen
- Maß Hilfestellung → Erwartungshaltung?

Fokus liegt auf Testen

monkik, “Covid Icons", https://www.flaticon.com/

Unser Projektansatz



Agil/Iterativ

0 Einleitung und Grundlagen

1 Das Projekt

2 Frontend

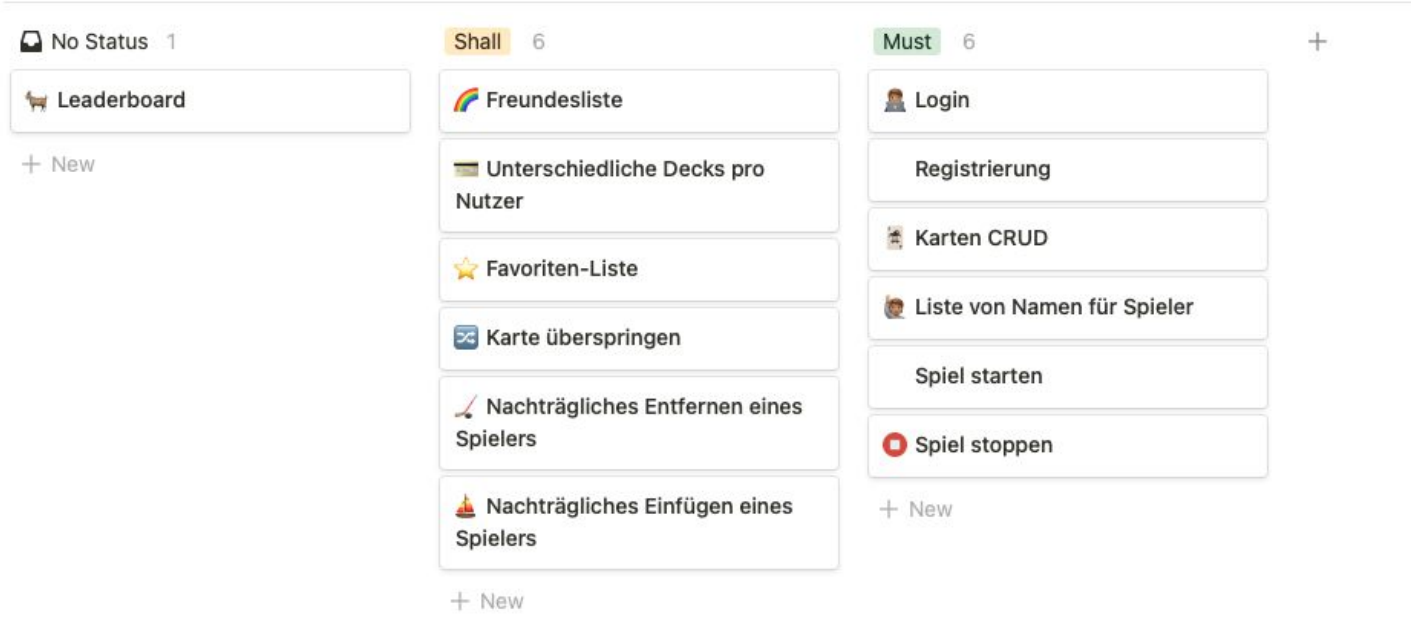
3 Backend

4 Lessons Learned

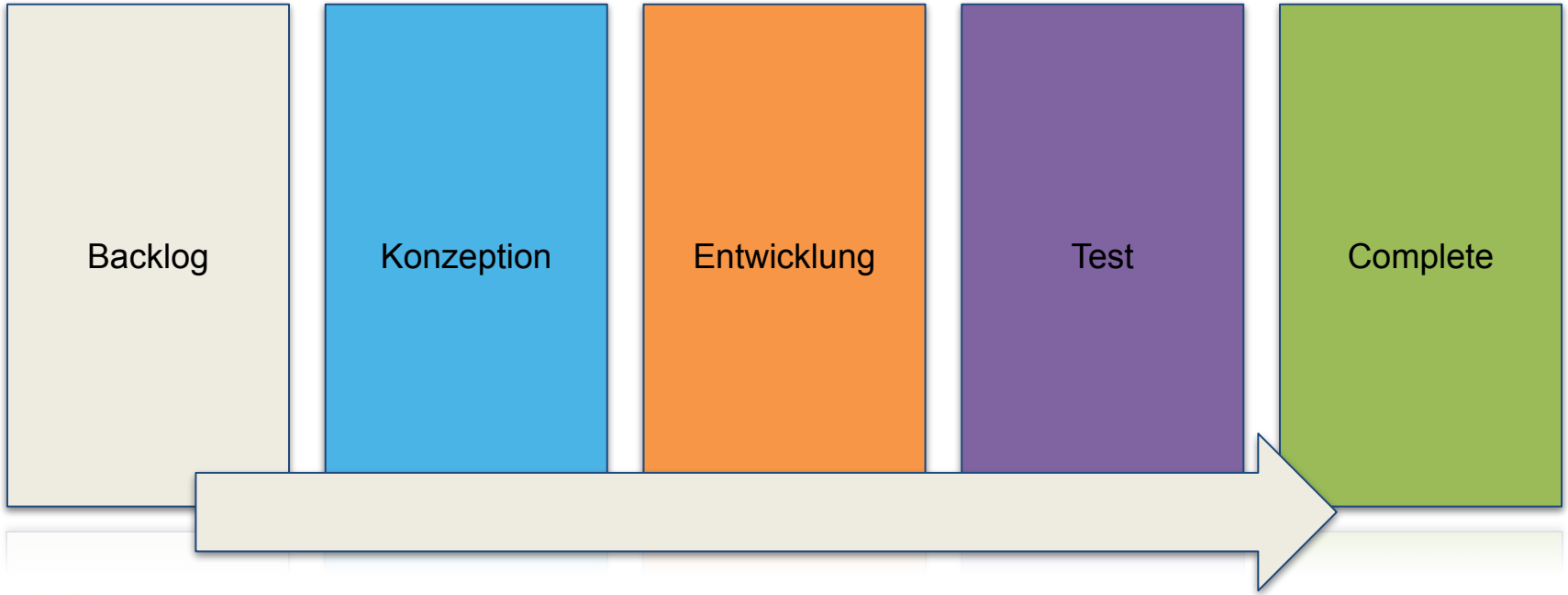
[1] ELEVATE, “unknown", Pexels https://www.pexels.com/

Das Projekt

UseCases


No Status 1
Leaderboard
New
Shall 6
Freundesliste
Unterschiedliche Decks pro Nutzer
Favoriten-Liste
Karte überspringen
Nachträgliches Entfernen eines Spielers
Nachträgliches Einfügen eines Spielers
New
Must 6
Login
Registrierung
Karten CRUD
Liste von Namen für Spieler
Spiel starten
Spiel stoppen
New

Kanban als Vorgehensmodell

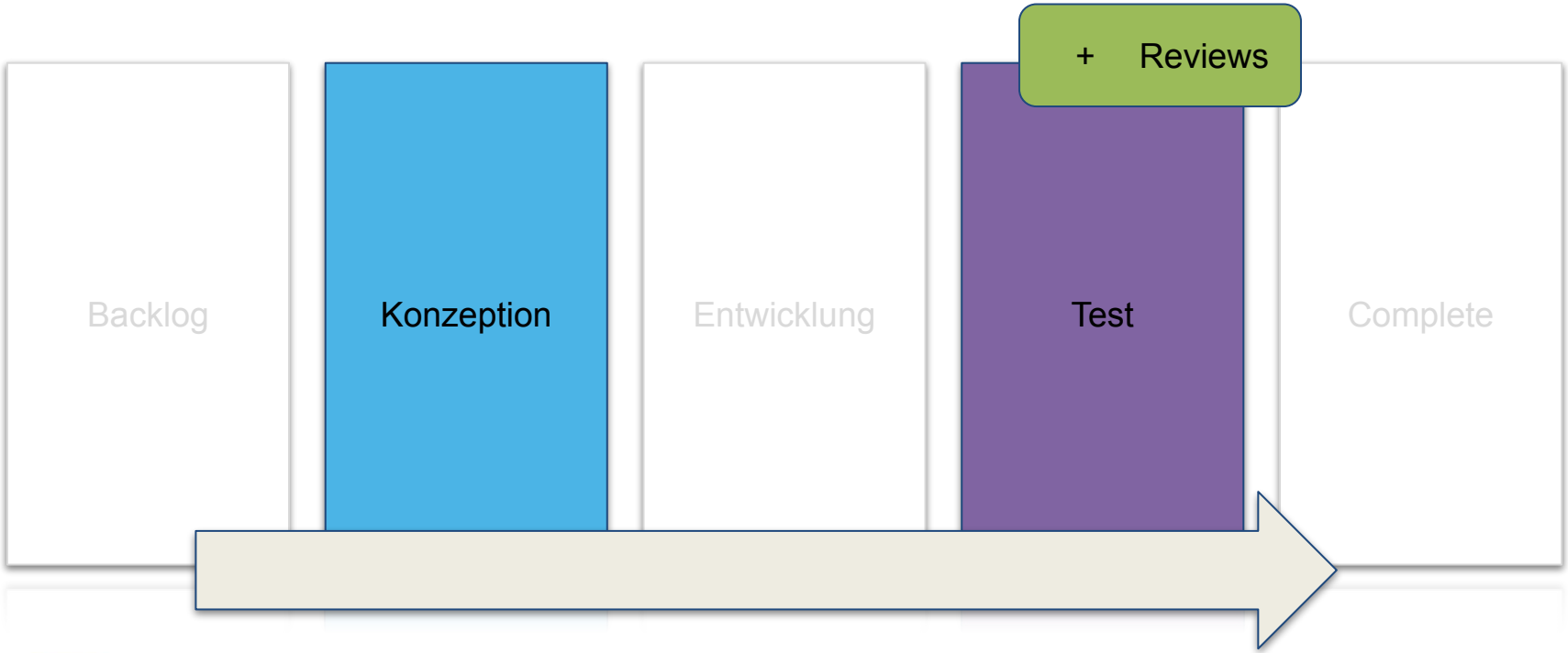


Inhalte der Konzeptions-Phase pro Ticket

Konzeption

- Beschreibung der Funktionalität
- Mockup der Benutzeroberfläche
- Testspezifikation
  - Testfall-Nr
  - Name
  - Setup / Voraussetzungen
  - Testdaten
  - Testschritte
  - Erwartetes Ergebnis

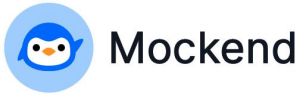
Unabhängigkeit



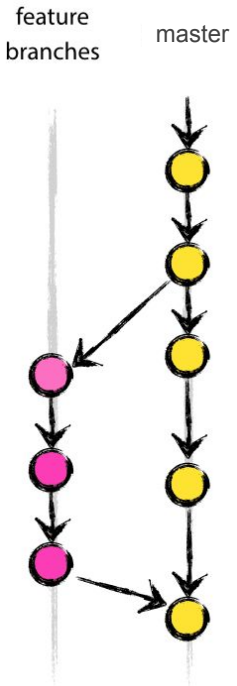
Teststufen

- TopDown + BigBang

Komponenten-Test	System-Tests	Komponenten-Integrations Tests	System-Integrations Tests



Merge-Konzept



- Github Repo
  - Frontend / Backend einzeln
- Vorgehen
  - Master Branch als “zentrale Wahrheit”
  - Tickets in Feature-Branches realisiert
  - Pull-Request pro Feature-Branch
- *“Tests on push”*, durch Github Actions (Frontend)

0 Einleitung und Grundlagen

1 Das Projekt

2 Frontend

3 Backend

4 Lessons Learned

Frontend Projektvorgehen

Iterationen:

1. Registrierung & Login

2. Hauptmenü
   Spiel-Funktionalität + Konfiguration
   Karten

Testvorgehen

- Definition Testfälle & grobes Mockup der Benutzeroberfläche
- Review
- Implementierung Funktionalität
- Implementierung Testfälle -> alle grün
- Review
- Integration in Master & Überprüfung Testfälle

Testumgebung

- Allgemein: - lokale System des jeweiligen Entwicklers

  - Github Actions

- Frontend: auf einem GitHub Server zu bauen, auszuführen, testen

  Vorteil: Rückmeldung über Status des Projektes

Übersicht

Teststufen	Komponententest	Integrationstest	Abnahmetest
Testmethode	Unit Tests	End-2-End Tests Regressions Tests	Usability Tests
automatisiert / manuell	automatisiert	automatisiert	manuell

Unit Tests

- Jasmine Test-Framework
- Karma als Test-Runner
- Implementierungs - &Testcode mit Webpack

Testfall - Beispiel:

- Registrierung
- überprüft ob der Benutzer registriert wurde

```
it( expectation: 'should register the user',    assertion: () => {
  updateForm( username: 'testUsername',  password: 'testPassword');
      component.onSubmit();
      expect(spy).toHaveBeenCalledWith( params: { username: 'testUsername', password: 'testPassword' });
      expect(component.registerError).toBeFalsy();
});
```

End-2-End-Tests

- Test-Framework: Cypress
- Gemeinsame Ausführung von Tests & Applikation
- Kontinuierliche Kommunikation von Client & Server

cypress.io

```
it( title: 'has the correct title', fn: () => {
    cy.title().should( chainer: 'equal', value: 'Drinking Game');
});
```

```
/// Register and Login Buttons do exist and have the correct content.
cy.get('[data-testid="login-button"]').should( fn: $p => {
    expect($p.first()).to.contain( value: 'Login');
});
cy.get('[data-testid="register-button"]').should( fn: $p => {
    expect($p.first()).to.contain( value: 'Registrieren');
});
```

ingenieur
wissenschaften
htw saar

STL SYSTEMTECHNIKLABOR

Demo

Manuelle & Regressions Tests

Manuelle Tests:

- Benutzer führt die Tests selbst aus

Regressions Test:

- Sicherstellung der Tests bei Modifikationen
- Verhindern von neuen Fehlern

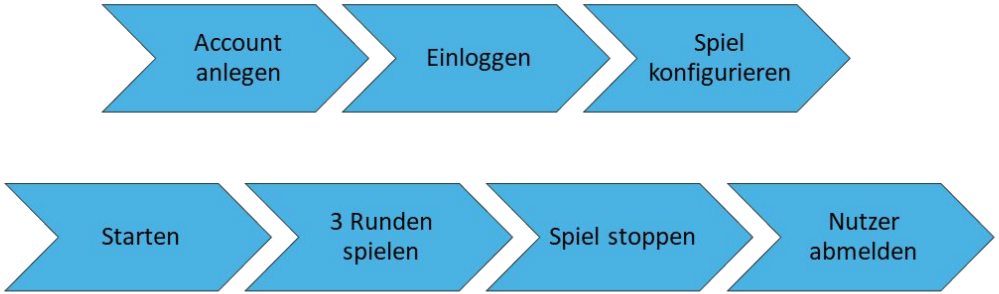
Usability Tests

Wichtige Aspekte für die Auswertung

?	?	?	?
Anwendung leicht verständlich & selbsterklärend	Spieldurchlauf ohne Vorkenntnisse durchführbar	Entspricht der Zeitrahmen des Spielstarts der Vorgabe	Unentdeckte Bugs oder Fehler

Usability Tests

Testablauf

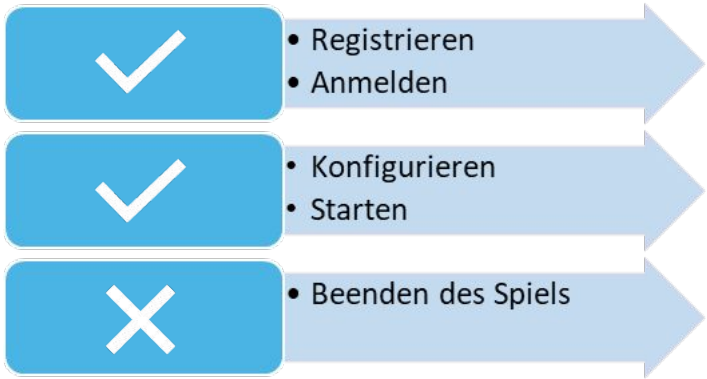


Usability Tests - Testperson A

Person A

- 23 Jahre
- Studiert Wirtschaftsinformatik
- Vorkenntnisse: ähnliche Apps

Auswertung: - 4 Minuten

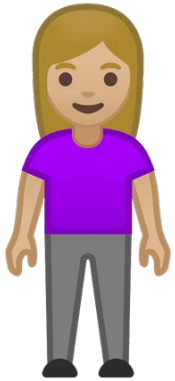


ingenieur wissenschaften
htw saar

STL SYSTEMTECHNIKLABOR

Usability Tests - Testperson B

Person B



- 28 Jahre
- Studiert Medizin (Zahntechnik)
- Vorkenntnisse: keine

Auswertung: - 10 Minuten

Backend Projektvorgehen

- 3 Features in 2 Iterationen
- 1. Iteration: Registrierung und Login
- 2. Iteration: Karten
- Konzeptionen abgeleitet aus Anforderungen

Backend Testvorgehen

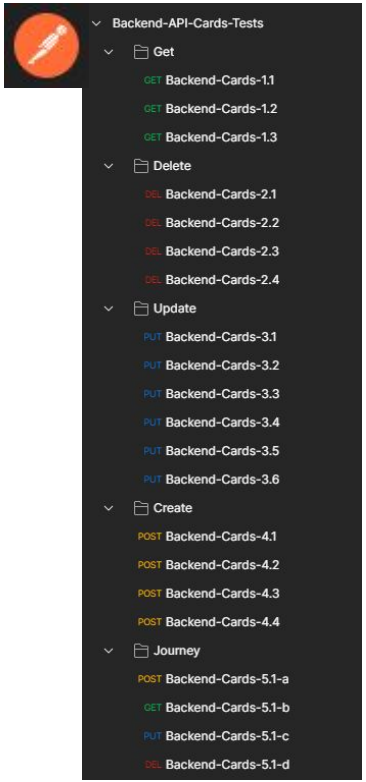
- Testspezifikation als Grundlage (Abgeleitet aus Konzeption)
  - Risikobasierte Priorisierung der Anforderungen (bzw. Tests)
- TDD-artige Entwicklung
- Pair Programming
- Reviews von Code und Tests
- Hohe Testautomatisierung

Testartefakte

- Postmancollection
- Unit tests TS
- Testdaten in SQL
- Testsetup in JS
- Lasttests JMeter Format
- Testfallspezifikation

Backend Testen mit Postman Vorteile

- Schnelle Entwicklung
- Einfache Ausführung
- Test Organisation
- Automatisiert und integriert mit Newman
- Einzeltests sowie auch Journeytests



Backend Testen mit Postman Nachteile

- Keine Metriken (Testabdeckung)

- Reicht nicht aus
  - Nicht passend für alle Aspekte des Backends
  - Lasttests

Backend Komponententests

- Fokus auf “Low-Level” Code
- Verzicht auf breite Komponententest-Basis
  - Jeweils für die Features so entschieden

```
doesCardTextContainIllegalCharacters(text: string) {
    const regex = new RegExp("^([a-z]*|[A-Z]*|[0-9]*|\\s*|[.,?!]*)*$", "g",);
    if (!regex.test(text)) {
        throw new ForbiddenException();
    }
}
```

Testdatenmanagement

- Datensätze für Variablen in Tests (Users, Karten ..usw)
- Testdatenanlage über Skripte
  - Einfache Lösung
  - Besondere Flexibilität
- Nicht optimale Lösung
  - Bei den Karten (ID existiert)
- Hätte man früher betrachten müssen
  - Testdaten sind auch relevant für Systemtests
  - Integration von Testdatensetup in Tests

Backend Lasttests

- Warum Lasttest?
- JMeter Tool
  - Dynamische und statische Testressourcen laden
  - Reale Testumgebung
- Lasttest für Karten über HTTP-Requests
- Testdaten aus CreateCards werden für DeleteCards verwendet
  - Abhängigkeit von den Tests

0 Einleitung und Grundlagen

1 Das Projekt

2 Frontend

3 Backend

4 Lessons Learned

Lessons Learned

- Frühe Testkonzeption und Testplanung helfen enorm
  - Im Gegensatz zu erst Entwickeln dann Testen
- Kommunikation ist sehr wichtig
  - z.B. bei Abstimmung für Systemtests
- Übergeordnetes Testmanagement
  - Fehlende Rolle Testmanager
- Tool- und Technologieauswahl
- CI/CD Pipeline