NEXTGEN
- DEVELOPMENT -

Glossar

NextGen Development

Version 1.1

6. April 2025

Teammitglieder:
Julian Lachenmaier
Ayo Adeniyi
Din Alomerovic
Cedric Balzer
Rebecca Niklaus

Verantwortlich für dieses Dokument:
Ayo Adeniyi
Qualitätssicherungsteam

Projektmanagement

- **Kanban**
  Agile Methode mit visuellem Board („To-Do", „In Arbeit", „Erledigt") zur Workflow-Optimierung. Visualisiert den Arbeitsfluss und begrenzt parallele Aufgaben.

- **Scrum**
  Agile Framework mit kurzen Iterationen (Sprints) und regelmäßigen Reviews. Enthält Rollen (Product Owner, Scrum Master), Artefakte (Product Backlog) und Events (Daily Scrum).

- **Meilensteine**
  Wichtige Zeitpunkte im Projektverlauf für Phasenabschlüsse. Markieren erreichte Ziele und dienen als Entscheidungspunkte für weitere Planung.

- **Risikomanagement**
  Prozess zur Identifizierung, Bewertung und Minderung projektgefährdender Risiken. Beinhaltet regelmäßige Risikoanalysen und Notfallpläne.

- **ClickUp**
  Projektmanagement-Tool für Aufgabenverfolgung und Teamkollaboration. Unterstützt Aufgabenlisten, Gantt-Diagramme und Zeitplanung.

Softwareentwicklung

- **Versionskontrolle**
  System (z.B. GitHub) zur Nachverfolgung von Codeänderungen. Ermöglicht Versionierung, Kollaboration und Rückverfolgbarkeit aller Änderungen.

- **Branching-Modell**
  Strategie (z.B. Git-Flow) zur parallelen Entwicklung in separaten Branches. Definiert Strukturen für Feature-Entwicklung, Releases und Hotfixes.

- **Pull-Requests**
  Anfrage zum Mergen von Codeänderungen nach Peer-Review. Ermöglicht Code-Überprüfung, Diskussion und Qualitätssicherung vor Integration.

- **Continuous Integration (CI)**
  Praxis des häufigen Zusammenführens von Codeänderungen in einem zentralen Repository. Automatisiert Build- und Testprozesse bei jedem Commit.

- **Continuous Deployment (CD)**
  Automatisiertes Ausliefern von Softwareänderungen an Produktion. Ermöglicht schnelle und zuverlässige Releases nach erfolgreichen CI-Tests.

Web & API

- **REST**
  Architekturstil für verteilte Systeme, der HTTP-Methoden (GET, POST etc.) nutzt. Basiert auf zustandsloser Kommunikation und Ressourcen-Orientierung.

- **Endpunkt**
  Spezifische URL einer API für eine bestimmte Funktion (z.B. /api/events). Jeder Endpunkt reagiert auf bestimmte HTTP-Methoden und liefert strukturierte Antworten.

- **API**
  Programmierschnittstelle zur Kommunikation zwischen Softwarekomponenten. Definiert Endpunkte, Anfrage-/Antwortformate und Authentifizierung.

- **Endpunkt**
  Spezifische URL einer API für eine bestimmte Funktion (z.B. /api/events). Reagiert auf bestimmte HTTP-Methoden mit definierten Antworten.

- **JSON**
  JavaScript Object Notation - Datenformat für API-Kommunikation. Unterstützt hierarchische Strukturen und ist menschenlesbar.

- **OAuth**
  Standardisiertes Protokoll für API-Autorisierung (z.B. Login mit externen Diensten). Ermöglicht sichere Delegierung von Zugriffsrechten.

Testing

- **Unit-Test**
  Test isolierter Funktionen oder Module. Überprüft korrektes Verhalten einzelner Komponenten unabhängig vom Gesamtsystem.

- **Integrationstest**
  Test des Zusammenspiels mehrerer Komponenten. Sichert korrekte Interaktion zwischen Modulen und Subsystemen.

- **Postman**

Tool für API-Tests und -Dokumentation. Ermöglicht Anfrageerstellung, Testautomatisierung und Mock-Server.

- **Test Coverage**
  Metrik, die angibt, wie viel Prozent des Codes durch Tests abgedeckt sind. Wichtiger Indikator für Testqualität (Ziel meist ¿80%).

- **Regressionstest**
  Wiederholungstests bei Änderungen zum Schutz bestehender Funktionalität. Verhindert unbeabsichtigte Seiteneffekte durch neue Features oder Fixes.

Dokumentation

- **Swagger/OpenAPI**
  Spezifikation für REST-API-Dokumentation. Ermöglicht maschinenlesbare API-Beschreibungen und interaktive Testoberflächen.

- **Markdown**
  Leichte Auszeichnungssprache für technische Dokumentation. Unterstützt Formatierung, Tabellen und Code-Blöcke in einfacher Syntax.

- **Confluence**
  Kollaborationswerkzeug für technische Dokumentation. Bietet Wissensmanagement, Versionierung und Integration mit Entwicklungstools.

- **Diagramme**
  UML, Flussdiagramme und Architekturdiagramme zur Systemdarstellung. Visualisieren Strukturen, Abläufe und Komponentenbeziehungen.

Plattformkonzepte

- **Organisator**
  Benutzer mit erweiterten Rechten innerhalb einer Organisation. Kann Events erstellen, Teilnehmer verwalten und weitere Organisatoren einladen. Verantwortlich für gesamten Event-Lebenszyklus.

- **Organisation**
  Repräsentiert Unternehmen oder Verein mit eigener Domain und Event-Portfolio. Hat mindestens einen Organisator und verwaltet eigene Mitglieder.

- **Event**

Physische Präsenzveranstaltung mit festem Ort und Zeitraum. Unterstützt Registrierung, Teilnehmerverwaltung und Prozesssteuerung.

- **Teilnehmer**
  Registrierter Benutzer mit Event-Anmeldung. Kann selbst registrieren oder eingeladen werden. Erhält Event-Informationen und Bestätigungen.

- **Prozesssteuerung**
  Konfiguration von Event-Phasen (z.B. Anmeldung) mit anpassbaren Aktionen. Ermöglicht Automatisierung von Workflows und Benachrichtigungen.

- **Registrierungsprozess**
  Mehrstufiger Anmeldevorgang: 1) Plattformregistrierung, 2) Organisationsbeitritt, 3) Event-Anmeldung. Erfordert Benutzername, Passwort und organisationsspezifische Zugangsdaten.

- **Teilnehmerverwaltung**
  Integriertes Tool für Organisatoren zur Verwaltung von Teilnehmern. Ermöglicht Einladungen, Statusverwaltung, Filterung und Export von Teilnehmerdaten.

- **Flow**
  Konfigurierbarer Prozessschritt, der Aktionen bei bestimmten Ereignissen auslöst (z.B. automatische E-Mail nach Registrierung). Bietet Wenn-Dann-Logik für Event-Prozesse.

- **Automatisierte E-Mails**
  System zur Versendung vorbereiteter Nachrichten an Teilnehmer. Unterstützt Templates und individuelle Gestaltung, konfigurierbar in den Flows.

Qualitätssicherung

- **Quality Gate**
  Qualitätsprüfpunkte im Entwicklungsprozess. Definiert Mindestanforderungen für Code-Qualität, Testabdeckung und Performance vor Release.

- **Code Smell**
  Oberflächliche Hinweise auf mögliche Probleme im Quellcode. Indikatoren für schlechtes Design (z.B. lange Methoden, duplizierter Code).

- **Technical Debt**
  Konsequenzen von kurzfristigen Lösungen, die langfristige Wartung erschweren. Sollte bewusst aufgenommen und zeitnah abgebaut werden.

- **Peer Review**
  Systematische Überprüfung von Code durch andere Entwickler. Dient Qualitätssicherung, Wissensaustausch und Fehlervermeidung.

Sicherheit

- **HTTPS**
  Sichere HTTP-Version mit Verschlüsselung via TLS/SSL. Schützt Datenintegrität und Vertraulichkeit bei der Übertragung.

- **JWT**
  JSON Web Token - Sicherheitstoken für Webanwendungen. Enthält Claims zur Authentifizierung und Autorisierung in kompakter Form.

- **CORS**
  Sicherheitsmechanismus für webbasierte API-Zugriffe. Steuert, welche Webanwendungen auf Ressourcen zugreifen dürfen.

- **OWASP**
  Open Web Application Security Project - Richtlinien für Websicherheit. Enthält Top-10-Risiken und Schutzmaßnahmen für Webanwendungen.