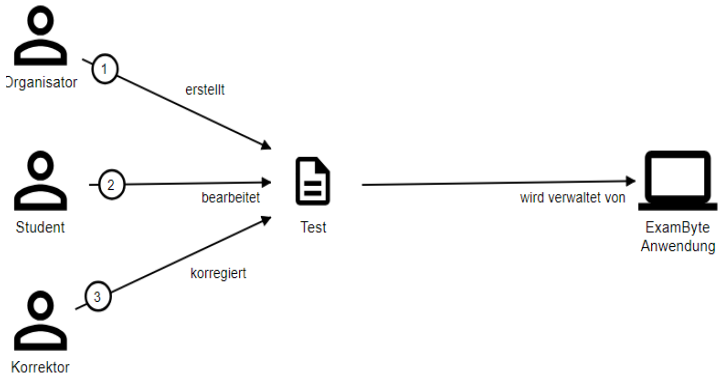
1. Einführung und Ziele
   - Was ist ExamByte?
     - ExamByte ist eine Anwendung zur Leistungskontrolle von Studenten.
     - Dort werden Tests von Organisatoren erstellt, von Studenten bearbeitet und von Korrektoren korrigiert.

   - Wesentliche Features
     - Organisatoren können Tests erstellen und diese vor Beginn in einer Testvorschau anschauen und bearbeiten.
     - Studenten können Tests bearbeiten, und abgelaufene Tests anschauen.
     - Korrektoren erhalten eine Liste an noch zu korrigierenden Tests.
     - Nach dem Ergebniszeitpunkt sehen die Studenten ihre Testergebnisse und Korrekturen.

   - Ziele
     - Leicht zugängliche Webseite, zum Erstellen, Korrigieren und Bearbeiten von Tests
     - Benutzerfreundliches User-Interface
     - Überschaubares und ansprechendes Design der Webseite
     - Einhaltung der Anforderungen zur Software

2. Randbedingungen
   - Implementierung in Java
   - Nutzung der PostgreSQL Datenbank
   - Team: Miran Alci und Kerstin Neu
   - Zeitplan: Oktober 2024 bis Februar 2025
   - Vorgehensmodell: Entwicklung des Projekts ist Risikogetrieben und iterativ
   - Entwicklungswerkzeuge: Entwurf auf Papier und anschließend mit IntelliJ

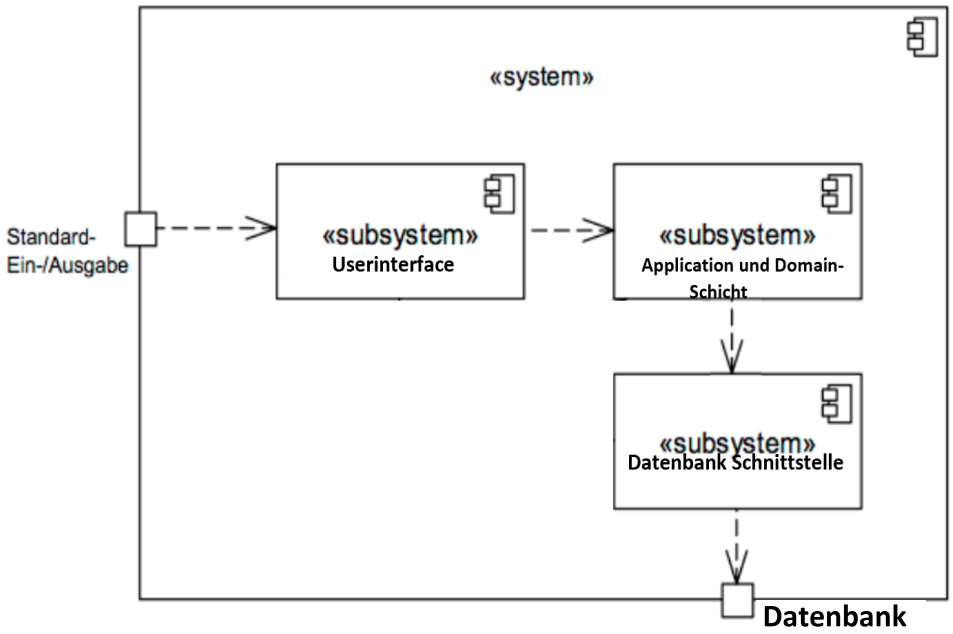
3. Kontextabgrenzung




   - Organisator: Der Organisator ist verantwortlich für die Erstellung der Tests, die nachher von Studenten bearbeitet werden sollen.
   - Student: Studenten bekommen eine Liste an bearbeitbaren Tests, die sie bis zum Endzeitpunkt bearbeiten können. Nach Ergebniszeitpunkt können sie dann ihre Korrekturen einsehen.
   - Korrektor: Korrektoren bekommen von Studenten bearbeitete Tests, denen sie eine Punktzahl und ein Feedback geben.

- Test: Tests bestehen aus einer Liste an Aufgaben. Diese sind entweder Multiple Choice Fragen oder Freitextfragen. Zu jeder Frage wird auch die Antwort und die Erklärung gespeichert.
- ExamByte: ExamByte verwaltet die Tests und stellt sicher, dass die Tests jeweils den richtigen Personen angezeigt werden.

4. Lösungsstrategie
   - Architekturüberblick gegliedert nach arc42
   - Verwendung der Onion-Architektur im Code
   - Nutzung der Programmiersprache Java
   - Modul-, Klassen- und Methodenname vorzugsweise in Deutsch
   - Unveränderliche Objekte erleichtern die Implementierung
   - Hohe Testabdeckung zur Kontrolle




5. Bausteinsicht
- Userinterface: Das Unserinterface zeigt die Tests je nach Rolle passend an. Für Organisatoren werden die erstellten Tests angezeigt und man kann neue Tests erstellen. Studenten sehen die verfügbaren Tests und können sie bearbeiten oder ggf. die Korrekturen anschauen. Korrektoren sehen die Abgaben der Tests, die noch zu Korrigieren sind.
- Application- und Domain Schicht: Hier wird die Logik der oben beschriebenen Funktionalität verwaltet. Dort werden Tests aus der Datenbank gelesen und passend angezeigt.
- Datenbank Schnittstelle: Dort wird es ermöglicht, neue Tests zu speichern und auf diese Tests zuzugreifen, die aus der Datenbank geladen werden.
- Datenbank: Hier werden Tests gespeichert.

6. Querschnittliche Konzepte
   - Domänenmodell: Es gibt für jede Rolle (Organisator, Student, Korrektor) eine eigene Ansicht des Testes. Diese findet man im aggregates Ordner, wo wir all unsere Aggregate speichern.

- Organisator Test: Liste an Aufgaben (Multiple Choice oder Freitext Aufgabe), dazu Lösungen und Erklärungen, TestIds, Datumsdaten, uvm.
- Studententest: Speichert Liste an Aufgaben und Liste an Abgaben der Studententest, TestId
- Korrektoren: Speichert Abgaben der Studenten, mit Testaufgabe und Abgabe
- Validierung: Die Datumsangaben müssen die richtige Reihenfolge haben (Startzeitpunkt darf nicht nach dem Endzeitpunkt sein). Multiple Choice Fragen müssen immer eine Antwort enthalten. Korrekturen müssen bei nicht-leerer Abgabe und nicht-voller Punktzahl vorhanden sein.

7. Glossar

- TestFormular: TestFormular ist das Formular der Tests von Organisatoren. Sie beinhalten alle Informationen zum Test. Titel, Startzeitpunkt, Endzeitpunk, Ergebniszeitpunkt, TestID, Liste an Mc-Fragen, Freitext-Fragen, Antwortmöglichkeiten

- Freitext Fragen: Freitext-Fragen bestehen aus Titel, Fragestellung, Punktzahl und einer Erklärung. Studenten sollen dann die Frage beantworten indem sie einen Text schreiben, der diese Fragestellung beantworten soll. Korrektoren korrigieren und bewerten diese Antwort dann im Nachhinein.

- Multiple-Choice-Frage: Besteht aus Titel, Punktzahl, Fragestellung, Antwortmöglichkeiten und einer Erklärung. Studenten sollen dann die Antworten, die sie für richtig halten, ankreuzen. Diese werden bei Ergebniszeitpunkt automatisch korrigiert. Dabei wird auch automatisch eine Erklärung generiert, welche Antwortwahlen richtig, und welche falsch waren.

- Antwortmöglichkeit: Diese besteht aus einem Namen (mögliche Antwort) und einem Wahrheitswert antwort, der aussagt, ob die Antwort richtig oder falsch ist.

- Studi Test: Studi-Tests bestehen aus einer Id, TestDaten, Liste an Freitext Aufgaben, Liste an Multiple-Choice-Aufgaben, Liste an Multiple-Choice-Antworten, Liste an Freitext-Antworten. Dieses Aggregat verwaltet also die generellen Test Daten, die Aufgaben und die zugehörigen Antworten der Studenten.

- TestDaten: Diese beinhalten den Start-, End- und Ergebniszeitpunkt und den Titel des Testes.

- Freitext-Aufgaben: Besteht aus Aufgabe (Titel der Aufgabe), Aufgabenstellung, Test-Titel und Punktzahl. Die Klasse dient dazu, dass der Student eine Freitextfrage ohne Antwort einsehen kann.

- Multple-Choice-Aufgaben: Besteht aus Aufgabe (Titel der Aufgabe), Aufgabenstellung, Test-Titel, Punktzahl und einer Liste an Antwortmöglichkeiten. Die Klasse dient dazu, dass der Student eine Mc-Frage ohne Antwort einsehen kann.

- Antwortmöglichkeit: Besteht aus der möglichen Antwort.

- Freitext-Antworten: Besteht aus der Antwort (vom Studenten), der Studi-Test-Id, der Aufgaben-Id und der Id des Studenten, welcher die Antwort abgegeben hat. Hier wird

verwaltet, welcher Student welche Antwort zu welche Aufgabe abgegeben hat.

- Multple-Choice-Antworten: Besteht aus Antworten (Liste der angekreuzten Antwortmöglichkeiten als String gecastet), der Studi-Test-Id, der Aufgaben-Id und der Id des Studenten, welcher die Antwort abgegeben hat. Hier wird verwaltet, welcher Student welche Antwortmöglichkeiten zu welche Aufgabe abgegeben hat.

- Abgabe: Besteht aus der Studi-Antwort, der Aufgabe (Titel und Aufgabenstellung), der Aufgabenstellung, dem Titel der Aufgabe, der AufgabenId, der StudiId, dem Studi-Test-Titel, dem Feedback, der Punktzahl, der maximal erreichbaren Punktzahl, der Studi-Test-Id und der Antwort-Id. Hier werden die einzelnen Abgaben mit Korrektur zu jeder beantworteten Frage gespeichert. Diese können dem jeweiligen Studenten zugeordnet werden.