CloneCademy

Qualitätssicherungsdokument

Gruppe 12:	Ilhan Simsiki <ilhan.simsiki@stud.tu-darmstadt.de> Leonhard Wiedmann <leonhard.wiedmann@stud.tu-darmstadt.de> Tobias Huber <tobias.huber@stud.tu-darmstadt.de> Claas Völcker <c.voelcker@stud.tu-darmstadt.de>
Teamleiter:	Alexander Nagl <alexander.nagl@t-online.de>
Auftraggeber:	iGEM Team TU Darmstadt vertreten durch Thea Lotz <lotz@bio.tu-darmstadt.de> Fachbereich Biologie
Abgabedatum:	14.07.2017

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Bachelor-Praktikum SoSe 2017
Fachbereich Informatik

Inhaltsverzeichnis

Anmerkungen und Hinweise

Die Verfasser dieses Dokumentes verwenden als Maßnahme für die Gleichstellung aller Geschlechter die sogenannte „gendergerechte Sprache“. Wir weichen nur von dieser Regelung ab, wenn entweder alle Mitglieder einer Gruppe sich ein und demselben Geschlecht zugehörig fühlen oder die Vorgaben der Veranstalter*innen keine Alternative zulassen (z.B. beim Deckblatt).

1 Einleitung

CloneCademy ist ein Projekt für das iGEM-Team der TU Darmstadt. Die *international Genetically Engineered Machine competition* (iGEM) ist ein internationaler Wettbewerb für Studierende auf dem Gebiet der Synthetischen Biologie. Dieser wird seit 2003 von der iGEM-Foundation veranstaltet.

Im Rahmen des Wettbewerbs wird eine Online-Lernplattform für das iGEM-Team der TU Darmstadt erstellt. Das Ziel dieser Plattform ist es, durch interaktive Unterrichtseinheiten Prinzipien der Molekularbiologie sowie der synthetischen Biologie zu erlernen und sowohl eigene Lernfortschritte, als auch die anderer Teams begutachten zu können. Darüber hinaus soll es auch anderen Interessierten (z.B. andere iGEM-Teams, Lehrende an Universitäten und Schulen, etc.) möglich sein, eigene Inhalte einzupflegen und zur Verfügung zu stellen.

Das Ziel des Projektes ist eine voll funktionsfähige Webanwendung für Desktop- und Laptoprechner und die Browser *Firefox* und *Chrome* bereit zu stellen. Eine mobile Nutzung der Seite ist nicht Teil des Bachelor Praktikums, eine spätere Implementierung einer plattformübergreifenden Lösung (z.B. einer App) ist jedoch auf Wunsch der Auftraggeber*innen durch Nutzung der REST-Schnittstelle möglich.

Der minimale geplante Funktionsumfang umfasst die Möglichkeit einen Nutzungsaccount auf der Seite anzulegen und eingestellte Kurse zu bearbeiten. Außerdem wird es dedizierte Moderator*innen geben, die Inhalte auf der Webseite hochladen können.

Zusätzlich sind mehrere Erweiterungen geplant. Allem voran soll es möglich sein, unterschiedliche Fragetypen zu nutzen. Als minimales Ziel wird hierfür die Möglichkeit gegeben, Multiple Choice Fragen zu stellen. Weitere Arten von Fragen werden im Laufe des Projektes mit den Auftraggeber*innen besprochen und beschlossen.

Da auch die Interaktion verschiedener Nutzer*innen und der Lernfortschritt erklärte Ziele der Plattform sind, sind hierfür zusätzliche Erweiterungen geplant. Dazu zählen ausführliche Statistiken zum Lernfortschritt und die Möglichkeit, diesen mit anderen Nutzer*innen der Plattform zu teilen und zu vergleichen.

Das erste Release auf einem Server der Auftraggeber*innen wird Ende Juli/Anfang August stattfinden. Bis dahin soll das Produkt die oben genannten minimalen Anforderungen erfüllen und stabil laufen. Weitere Releases werden dann zusammen mit den Auftraggeber*innen abhängig von der Geschwindigkeit des Teams geplant.

2 Qualitätsziele

2.1 Datensicherheit (Security)

Im Rahmen des Projekts CloneCademy wird eine Webanwendung entwickelt, auf welche über das Internet zugegriffen werden kann. Daher ist die Sicherung gegen unbefugten Zugriff und Änderung der Daten der Webseite in diesem Projekt das wichtigste Qualitätsziel. Da CloneCademy sowohl persönliche Daten der Nutzer*innen als auch Metadaten über die Nutzung der Plattform und die Inhalte der einzelnen Lerneinheiten persistent speichert, ist es sehr wichtig, dass Internetnutzer*innen diese Daten nicht verändern oder einsehen können, solange sie dazu nicht die benötigten Rechte besitzen.

Die größte Bedrohung geht von bekannten Webangriffen und Misskonfigurationen im Backend einer Webseite aus. Während dieses Projekts wird darauf geachtet, dass die Plattform gegen die wichtigsten Sicherheitslücken einer Webanwendung gesichert ist. Da eine umfassende Sicherung gegen alle möglichen Angriffsvektoren und Schwachstellen den Projektumfang weit übersteigen würde, befassen wir uns im Rahmen dieses Projektes vor allem mit der Sicherheit der Nutzeroberfläche und der Schnittstelle.

Die betrachtenten Angriffsvektoren[1] sind:

- Möglichkeiten zur Ausführung fremden Codes (Injection & Cross-Site Scripting)
- Fehlerhafte Zugriffskontrolle
- Nutzung von Komponenten mit bekannten Schwachstellen

Um die Sicherheit der Anwendung zu gewährleisten, folgen wir einem mehrschrittigen Plan.

Ein Entwickler des Teams wurde zum Sicherheitsbeauftragten ernannt, der die korrekte Durchführung aller genannten Maßnahmen sicher stellt. Seine Aufgabe ist es, auf die Einhaltung aller Programmierrichtlinien zu achten und Tests zur Validierung der Schutzziele durchzuführen. Die genaue Vorgehensweise wird im Folgenden detailliert ausgeführt.

Während der ersten Gespräche mit den Auftraggeber*innen wurden Nutzerrollen definiert (Admin, Moderator*in und Nutzer*in)[2]. In jeder User Story wird festgehalten, ob und welche Zugriffsbeschränkungen einzelne Module der Webseite oder Daten besitzen. Um die Einhaltung der Nutzer*innenrollen zu ermöglichen, werden die zur Verfügung gestellten Authentifizierungswerkzeuge der genutzten Frameworks (Django, Django REST, Angular2) verwendet.

[1] Paraphrasiert nach der verbreiteten Liste des Open Web Application Security Project (OWASP)
Link zum Download: https://github.com/OWASP/Top10/raw/master/2017/OWASP%20Top%2010%20-%202017%20RC1-English.pdf

[2] Detaillierte Beschreibung der Rollen finden sich im Anhang

Während der Implementierung halten alle Entwickler die Best Practices im Web Development ein. Als Referenz dafür werden die Handreichungen des Open Web Application Security Project (OWASP)[3] verwendet. In den wöchentlichen internen Code-Reviews wird die Qualität des Codes hinsichtlich dieser Richtlinien überprüft.

Um die Qualität des Codes im Python-basierten Backend automatisiert zu testen, wird das verbreitete Werkzeug *bandit*[4] verwendet. Als statisches Analyse-Werkzeug für das Frontend wird *ng2lint*[5] verwendet. Beide Programme unterstützen die Code-Review, indem die gelieferten Meldungen als Grundlage für eine detaillierte Analyse des Codes genutzt werden. Zusätzlich wird eine Sicherheits-Checkliste[6] geführt, die die Best Practices zur sicheren Webentwicklung und das Überprüfen aller Reports der verwendeten Werkzeuge enthält. Um eine dauerhaft hohe Qualität des Codes zu gewährleisten, werden die Analyse-Tools automatisiert bei einem Push auf dem Development Branch des Git Repositories verwendet. Alle gefundenen Schwachstellen werden vor dem finalen Commit auf dem stable Branch zur Abnahme der User Story behoben. Wenn dies nicht möglich ist, wird eine Begründung verfasst und eine realistische Einschätzung über die Risiken der gefundenen Fehler an die Auftraggeber*innen gemeldet. Diese entscheiden, ob die Risiken tragbar sind und die User Story trotz der Sicherheitslücke als erfolgreich gemeldet wird.

Um die tatsächliche Sicherheit des Endproduktes gegen externe Angriffe zu zeigen, wird ein automatisiertes Penetrations-Werkzeug verwendet. Das OWASP empfiehlt hierfür das *Zed Attack Proxy Project*, welches eine Reihe bekannter Angriffe auf die Plattform ausführt und die gefundenen Schwachstellen meldet. Da ein gesamter Test der Anwendung sehr zeitaufwändig ist, führt der Sicherheitsbeauftragte diese Tests monatlich durch und meldet die Ergebnisse an die jeweiligen Entwickler des betroffenen Moduls, damit diese innerhalb des nächsten Monats behoben werden können. Ist eine Behebung nicht möglich, werden die sicherheitsrelevanten Gefahren dieser Lücke zusammen mit möglichen weiteren Schritten zum Schutz der Anwendung im Betrieb an die Auftraggeber*innen gemeldet.

Da ein Fokus auf Sicherheit zwar wichtig ist, aber Korrekturen erst vorgenommen werden können, sobald das Kernprodukt nicht mehr verändert wird, werden alle vorgestellten Maßnahmen ab Juli durchgeführt.

2.2 Bedienbarkeit (Entwurf)

Clonecademy ist eine Lernplattform und steht als solche einer großen Vielfalt an Benutzer*innen zur Verfügung. Daher muss die Plattform für alle Benutzer*innen einfach und intuitiv bedienbar sein, ohne das eine ausführliche Einarbeitung notwendig ist. Um dies zu gewährleisten wird das Design aller Bereiche der Webseite nach diesem Kriterium evaluiert. Die grundlegende Bedienung der Webseite wird ohne Einführung erlernbar sein und komplizierte Elemente werden mit einer ausreichenden Erklärung versehen.

[3] https://www.owasp.org/images/0/08/OWASP_SCP_Quick_Reference_Guide_v2.pdf
[4] https://wiki.openstack.org/wiki/Security/Projects/Bandit
[5] https://www.npmjs.com/package/ng2lint
[6] siehe Anhang

Auch für dieses Ziel wurde ein Mitglied des Teams benannt, um Nutzungsstudien durchzuführen und auf die Einhaltung aller Richtlinien zu achten.

Um das oben formulierte Ziel zu erreichen, verwenden wir als Grundlage für das Design der Seite das Material Design[7] für Angular2. Dies sorgt dafür, dass die Webseite einen einheitlichen und übersichtlichen Aufbau hat. Um Knöpfe und andere Schaltflächen übersichtlicher zu machen, werden diese mit Icons[8] versehen. Bei der internen Code Review prüft der zuständige Entwickler bei allen veränderten oder neuen Bereichen der Webseite, ob sie noch den Richtlinien eines einheitlichen Designs entsprechen[9].

Da die Bedienbarkeit der Oberfläche nicht objektiv gemessen werden kann, werden zuzüglich zu den Designrichtlinien Nutzungsstudien durchgeführt. In den Studien setzen sich die Proband*innen ohne vorherige Erklärung mit der Plattform auseinander und werden dazu aufgeforder, verschiedene Aufgaben zu erfüllen. Durch Aufzeichnen des Bildschirms während der Studie wird das Verhalten der Nutzer*innen erfasst, um dieses später zu reflektieren. Zusätzlich wird die persönliche Meinung der Proband*innen nach der Studie mit einem Fragebogen[10] erfasst.

Um zum Schluss ein, von einer Vielzahl unterschiedlicher Nutzer*innen, gut bedienbares Produkt zu haben, werden die Nutzungsstudien in mehreren Iterationen und mit unterschiedlichen Gruppen durchgeführt, um Verbessungsvorschläge aus den vorherigen Studien direkt wieder zu testen.

Nach jeder Nutzerstudie wird eine Checkliste mit den Problemen und Verbesserungsvorschläge der Probanden erstellt, welche bis zum nächsten Treffen mit den Auftraggeber*innen umgesetzt werden muss. Diese können dann die Designänderungen überprüfen und erst durch die Abnahme der Designänderungen ist die Iteration der Studie abgeschlossen.

Die Nutzerstudien erfolgen mindestens eine Woche vor den Releases und wenn die Auftraggeber*innen eine Zwischenevaluation des Designs wünschen.

2.3 Veränderbarkeit (Entwurf)

Für die Webanwendung CloneCademy ist es wichtig, dass die Anwendung im Nachhinein noch veränderbar ist. Es muss möglich sein, neue Inhalte in die Datenbank einzupflegen und auch den Quellcode der Webseite selbst ohne große Mühe erweitern zu können, da eine Lernplattform immer an die neusten Entwicklungen im Bereich eLearning angepasst werden muss.

Um diese Erweiterbarkeit zu gewährleisten, wird im Projekt auf folgende Punkte geachtet:

- Quellcode
- Kommentare

[7] https://material.angular.io/
[8] https://material.io/icons/
[9] Checkliste siehe Anhang
[10] siehe Anhang

- externe Dokumentation

Es wird ein Verantwortlicher für das Qualitätsziel benannt, der Ansprechpartner für alle Rücksprachen oder Fragen ist und dafür sorgt, dass die Qualität der oben aufgeführten Bereiche wie unten beschrieben sichergestellt wird.

Qualität des Quellcodes Grundlegend werden während der Entwicklung der Software die Styleguides der verwendeten Programmiersprachen und Frameworks umgesetzt. Diese sind der *Angular Style Guide*[11] für das Frontend und *PEP 8*[12] für das Backend. Um die Einhaltung dieser beiden Styleguides zu überprüfen, werden bei jedem Commit statische Analysetools verwendet, welche überprüfen, ob alle Richtlinien umgesetzt wurden.

Die verwendeten Tools sind *pep8*[13], welches die Einhaltung der generellen Python Richtlinien überprüft, *django-lint*[14], welches Framework spezifische Fehler markiert (z.B. Verwendung von veralteten Methoden) und *ng2lint*[15], welches den Frontend Code überprüft. Alle durch diese Tools gefundenen Fehler werden bis zum nächsten internen Gruppentreffen der Entwickler behoben, bevor der Code zur Abnahme durch die Auftraggeber*innen freigegeben wird.

Bei der internen Code Review vor der Abnahme einer User Story überprüft der für die Erweiterbarkeit zuständige Entwickler, ob alle von den Tools gemeldeten Fehler korrigiert wurden. Sollte dies nicht der Fall sein, müssen alle gefundenen Fehler korrigiert werden, bevor die Userstory den Auftraggeber*innen zur Abnahme vorgestellt wird.

Kommentare Jede Methode und jede Klasse im Code erhält einen Kommentar. Zusätzlich werden unklare Stellen im Code (z.B. komplexe Steuermechanismen oder Abfragen) mit eigenen Erklärungen versehen. Alle Kommentare werden in Englisch verfasst.

Bei der Code Review überprüft der Beauftragte für das QS-Ziel, ob alle Klassen und Methoden kommentiert sind und ob die Kommentare das Verständnis des Codes unterstützen[16]. Zusätzlich markieren alle an der Code Review beteiligten Entwickler die Stellen im Code, die sie nicht für unmittelbar verständlich halten. Diese werden auch mit Kommentaren versehen. Sollte es nicht möglich sein, den gesamten Code ausführlich zu dokumentieren, wird die Userstory den Auftraggeber*innen in dieser Iteration nicht zur Abnahme präsentiert.

Wiki Das Projekt verwendet zu klaren Trennung zwischen Back- und Frontend eine sogenannte REST-API. Diese stellt einzelne Schnittstellen da, über die die Daten des Backends abgerufen werden können. Diese werden deshalb gesondert dokumentiert, da eine korrekte Implementierung und Nutzung essentiell für das Projekt ist.

Dazu wird ein Wiki regelmäßig aktualisiert, in dem alle Schnittstellen zwischen Backend und Frontend definiert sind. Ist für eine Userstory eine neue Schnittstelle nötig, wird diese von al-

[11] https://angular.io/guide/styleguide
[12] https://www.python.org/dev/peps/pep-0008/
[13] https://pypi.python.org/pypi/pep8
[14] https://pypi.python.org/pypi/django-lint
[15] https://www.npmjs.com/package/ng2lint
[16] Checkliste siehe Anhang

len Entwicklern zusammen entworfen und dann im Wiki dokumentiert[17]. Die Beteiligung aller ist notwendig, da einen Schnittstelle nicht nur für eine spezifische Aufgabe im Rahmen einer Userstory dienen sollte, sondern allgemeine Funktionalitäten bietet.

Die Dokumentation im Wiki wird spätestens vor den Releases vervollständigt. Dazu überprüft der Beauftragte spätestens zwei Wochen vor Release, ob alle Schnittstellen aufgeführt sind, und ob die Dokumentation den vorgegebenen Rahmen erfüllt. Beim nächsten internen Treffen wird das Wiki auf den aktuellen Stand gebracht und alle Dokumentation noch einmal überprüft. Bevor die Dokumentation nicht vollständig ist, wird die Software nicht zum Release freigegeben.

Testabdeckung Bei Codeerweiterung muss bereits bestehender Code weiterhin funktionieren. Um dies automatisiert zu überprüfen, muss der bereits implementierte Code getestet werden können. Das Team stellt eine umfangreiche Testabdeckung des Codes sicher. Umfangreich heißt in diesem Fall, dass eine vollständige Function Coverage und eine Statement Coverage von mindestens 80% erreicht wird. Eine Überprüfung der Abdeckung erfolgt im Backend durch das mitgelieferte Testframework des Django-Projekts. Im Frontend werden Tests mit Hilfe des Frameworks Selenium[18] durchgeführt.

Die Function Coverage wird vor jeder internen Code Review sichergestellt. Die Tests der API Schnittstellen werden bei deren Definition geschrieben, damit sie den jeweiligen Entwicklern als Hilfestellung dienen können. Da eine umfassende Statement Coverage während des Projektes nicht zu jeder Zeit implementiert werden kann, ist diese zu jedem Release anzufertigen. Damit wird gezeigt, dass der Code stabil ist und die Tests können als Basis für die Implementierung der kommenden Userstories verwendet werden.

Eine Userstory wird erst zur Abnahme präsentiert, wenn mindestens für alle durch diese Story verwendeten und neu implementierten Funktionen Tests vorliegen. Ein Release wird erst freigegeben, wenn eine Code Coverage von mindestens 80% erreicht ist. Die Überprüfung dieser Regeln erfolgt durch den für die Bedienbarkeit zuständigen Entwickler.

[17] Rahmen im Anhang
[18] http://www.seleniumhq.org/

A Checklisten

Im folgenden sind die verwendeten Checklisten aufgeführt. Für die Ziele *Datensicherheit* und *Erweiterbarkeit* haben wir einen Checkliste für die interne Code Review ausgearbeitet. Für die Bedienbarkeit wird keine Checkliste bereitgehalten, sondern die Checklisten ergeben sich aus den Rückmeldungen aus den Nutzungsstudien.

A.1 Checkliste Datensicherheit

Grundlegende Fragen klären:

- Programiersprache: Welche Feature und welche Probleme gibt es mit der Technik?
- Kontext: Was muss gesichert werden und wie wertvoll ist es? Welche Angreifer*innen werden befürchtet?
- Zielgruppe: Wer nutzt das Tool, dessen Code reviewt wird? Jemand internes, dem*der vertraut wird, oder eine außenstehende Person?
- Ausfallsicherheit: Welcher Grad von Ausfallsicherheit muss gewährleistet werden?

Bei der Codereview sollte folgendes Bedrohungsmodell vor Augen gehalten werden:

- Angreifende: Befürchtet werden vor allem Nutzende, die bei der Beantwortung der Fragen unerlaubte Methoden benutzen möchten, sowie Dritte, die gespeicherte Nutzerdaten missbrauchen wollen.
- Angriffsoberfläche: Als Oberfläche sollte vor allem die übersichtliche API betrachtet werden.
- Mögliche Angriffe: Da das System im Gegensatz zu z.B. Finanztechnologie als nicht besonders Angriffswert angeschätzt wird, werden die gebrächlichsten Angriffsformen wie zum Beispiel XSS, CSRF, Code und SQL Injection erwartet.
- Als Sicherheitsmaßnahmen wird die korrekte Implementierung von Django, REST und Angular2 verwendet.
- Potentielle technische Auswirkungen: Falsche/Korrupte Daten in der Datebank, v.a. in Trys
- Wichtige wirtschaftliche Auswirkungen: Verlust der Glaubwürdigkeit nach Verlust von privaten Daten

Folgende Fragen sollten zu Jedem Codeabschnitt beantwortet werden:

- html input: Wird jeglicher Input dieser Art korrekt nach Crossitescripts durchsucht bevor er als sicher markiert wird?
- Sind die Djangofeatures zum Schutz vor Cross Site Request Forgery (CSRF) immer aktiviert oder nur in sinnvollem Kontext deaktiviert?
- Ist jede view mit sinnvoller Authentifizierung und Authororisierung versehen, sowie gegebenenfalls auch mit einer Anfragenbegrenzung (Throttling)?
- Werden alle Nutzereingaben nach Fehlern gefiltert?
- Sind alle Security Probleme, die bei der softwarebasierten Analyse aufgetreten sind, behoben worden, oder aus dem Kontext heraus als unkritisch betrachtet worden?

A.2 Checkliste Erweiterbarkeit

A.2.1 Code Qualität

- □ Gibt es Meldungen der Tools?
- □ Gibt es obsoleten Code?
- □ Gibt es unbenutzte Variablen?
- □ Wurden bereits implementierte Funktionen nicht benutzt oder neu implementiert?
- □ Wurden Hilfsmethoden ausgelagert?
- □ Wurde die Datenbankenstruktur geändert?
    - □ Wenn ja, wurden die Änderungen dokumentiert?
- □ Für das Backend:
    - □ Geben alle Views einen gültigen und passenden Status Code zurück?
    - □ Sind alle Schnittstellen wie abgesprochen implementiert?
        - □ Wenn Änderungen vorgenommen wurden: Sind diese begründet und dokumentiert?
- □ Für das Frontend:
    - □ Sind alle POST Methoden richtig formatiert?

A.2.2 Kommentare

- □ Sind alle Klassen kommentiert?
- □ Sind alle Methoden kommentiert?
- □ Gibt es weitere schwer verständliche Stellen, die kommentiert werden sollten?
- □ Sind die Kommentare verständlich für alle Entwickler?
- □ Ist die Dokumentation im Wiki vollständig?

A.2.3 Tests

- □ Sind Tests aller Methoden vorhanden?
- □ Wie hoch ist die Statement Coverage?

B Fragebögen