MOC5x Projektarbeit WS 2015/16

☐ **Gruppe 1**
DI (FH) Gabriel Horn, MSc

☐ **Gruppe 2**
Philipp Pendelin, MSc

Abgabetermin: 14.01.2016, 24:00 Uhr

Name: _____

Aufwand (h): _____

Schnitzeljagd

Bei einer Schnitzeljagd (engl. "amazing race") suchen mehrere Spieler auf einem Gelände nach einer Reihe von verstecken Hinweisen, um Antworten auf Fragen zu finden, die den Spieler wiederum zum nächsten Hinweis führen. Als Gewinner gilt der Spieler, der alle Hinweise in der kürzesten Zeit gefunden hat. In Anlehnung an Geocaching lässt sich eine moderne Schnitzeljagd so gestalten, dass man dem Spieler zusätzlich noch die Geo-Koordinaten, bei denen sich der nächste Hinweis befindet, zur Verfügung stellt, die gemeinsam mit den Hinweisen selbst von einem Server zur Verfügung gestellt werden. Ausgestattet mit einem mobilen Endgerät, das den Ort des nächsten Hinweises über eine Karte visualisiert, macht sich der Spieler auf die Suche. Sobald er einen Hinweis gefunden hat, kann er nach Lösen einer zum Hinweis gehörenden Aufgabe über das Mobiltelefon automatisch den nächsten Hinweis herunterladen und seine Jagd fortsetzen.

Im Zuge dieser Projektarbeit soll eine solche mobile Anwendung zur Schnitzeljagd entwickelt werden, die als Frontend für einen bestehenden Schnitzeljagd-Dienst fungiert. Primäre Aufgaben dieser Anwendung sind das Laden und Visualisieren von Hinweisen sowie das Entgegennehmen von Antworten zum Freischalten von neuen Hinweisen.

Dienst (bestehend)

Als Backend für die Schnitzeljagd fungiert ein bestehender Dienst, welcher mehrere unterschiedliche Schnitzeljagd-Routen verwalten kann. Eine Route besteht jeweils aus eine Liste von Kontrollpunkten die der Spieler der Reihe nach besuchen muss. Für jeden Kontrollpunkt sind dessen Position, dessen Name und ein zusätzlicher Hinweis (z. B. "Gesucht ist die Farbe des Hauses bei den gegebenen Koordinaten.") mit passendem Geheimnis ("rot") hinterlegt. Mit dem jeweiligen Hinweis, den die mobile Anwendung entsprechend anzeigen muss, kann der Spieler eine Aufgabe lösen, sobald er bei den Koordinaten des Kontrollpunkts ankommt. Mit der korrekten Lösung (= Geheimnis) lässt sich im Backend für einen Spieler und eine Route der nächste Kontrollpunkt freischalten. Alle erfolgreich "gelösten" Kontrollpunkte auf einer Route werden für jeden Spieler getrennt dauerhaft vom Backend gespeichert. Eine Liste aller im bestehenden System angelegten Kontrollpunkte (inklusive der dazugehörigen Geheimnisse) steht im Moodle-Kurs zum Herunterladen bereit.

Vom mobilen Client aus kann über eine JSON-basierte Schnittstelle auf das Backend zugegriffen werden. Als Zugangsdaten verwenden Sie für beiden Schnittstellen sowohl als Benutzername als auch als Passwort Ihre Matrikelnummer in der Form „Sxx10307xxx“.

JSON-Schnittstelle

Endpunkt: https://demo.nexperts.com/MOC5/AmazingRaceService/AmazingRaceService.svc
Metadaten: https://demo.nexperts.com/MOC5/AmazingRaceService/AmazingRaceService.svc/help

Es stehen folgende Operationen zur Verfügung:

Aufruf via GET-Request *Parameterübergabe über **GET-Parameter in HTTP-Request**, Ergebnis als **JSON-Payload in HTTP-Response***	
`bool ValidateCredential(string userName, string password);`	Prüft eine Kombination aus Benutzername und Passwort auf Gültigkeit.
`IEnumerable<Route> GetRoutes(string userName, string password);`	Liefert eine Liste mit allen Routen. Für jede Route werden zusätzlich eine Liste aller bereits besuchten Kontrollpunkte sowie der nächste zu besuchende Kontrollpunkt mitgeliefert. Ist dieser leer (null), so hat der Spieler diese Route vollständig absolviert.
Aufruf via POST-Request *Parameterübergabe als **JSON-Payload in HTTP-Request**, Ergebnis als **JSON-Payload in HTTP-Response***	
`bool ResetRoute(RouteRequest request);`	Setzt eine einzelne Route zurück. Dabei werden für den aktuellen Spieler alle besuchten Kontrollpunkte dieser Route gelöscht. Liefert immer True zurück.
`bool ResetAllRoutes(Request request);`	Löscht sämtliche vom Spieler besuchten Kontrollpunkte aller Routen. Liefert immer True zurück.
`bool InformAboutVisitedCheckpoint(CheckpointRequest request);`	Informiert das Backend über einen besuchten ("gelösten") Kontrollpunkt. Stimmt das übergebene Geheimnis mit dem gespeicherten Geheimnis für den Kontrollpunkt überein, so wird vom Backend der nächste Kontrollpunkt auf der Route freigeschaltet und True zurückgeliefert. Danach kann der nächste Kontrollpunkt für die Route über GetRoutes() abgefragt werden.

Referenzimplementierungen der Klassen Route, Checkpoint, Request, RouteRequest und CheckpointRequest in C# und Java finden Sie im Moodle.

Entwickeln Sie eine einfache mobile Anwendung für die von Ihnen gewählte mobile Plattform, die mit Hilfe des gegebenen Dienstes das Teilnehmen an einer Schnitzeljagd ermöglicht. Dabei sind die nachstehend beschriebenen funktionalen und technischen Anforderungen so gut wie möglich umzusetzen.

Funktionale Anforderungen

- Beim Start der Anwendung muss sich der Spieler einmalig durch Eingabe von Benutzername und Passwort anmelden. Das Passwort darf dabei nicht im Klartext angezeigt werden. Nach erfolgreicher Prüfung der Zugangsdaten sollen diese global in der Anwendung gespeichert und bis zum Beenden der Anwendung automatisch für alle weiteren Service-Aufrufe verwendet werden.
- Nach dem Anmelden kann der Spieler eine Route wählen, die er spielen möchte.
- Für die gewählte Route wird der nächste zu besuchende Kontrollpunkt auf einer Landkarte eingezeichnet und dessen Name und Hinweis angezeigt.
- Der Benutzer kann für den aktuell zu besuchenden Kontrollpunkt auf einfache Weise die entsprechende Lösung (= das Geheimnis) eingeben und vom Backend prüfen lassen. Im Erfolgsfall wird der nächste Kontrollpunkt angezeigt.

- Des Weiteren soll der Spieler auch die bereits besuchten Kontrollpunkte einer Route wieder einsehen und seinen (theoretisch) zurückgelegten Weg auf einer Landkarte betrachten können.
- Beim erfolgreichen Abschluss einer Route (bzw. beim erneuten Öffnen einer bereits abgeschlossenen Route) soll der erfolgreiche Spieler mit einer entsprechenden (= entsprechend einfachen *g*) Siegesmeldung (z. B. in Form eines Dialogs) belohnt werden.
- Das Rücksetzen von einzelnen oder allen Routen muss nicht implementiert werden.

Technische Anforderungen

- Die mobile Anwendung ist für die gewählte mobile Plattform (entweder Android oder Windows-Phone) zu realisieren.
- Auf die besonderen Gegebenheiten der gewählten Plattform ist bei der Entwicklung entsprechend einzugehen. Für eine Plattform übliche Konzepte und "Best-Practices" (wie z. B. Model-View-ViewModel bei Windows Phone) sind dabei auf jeden Fall zu berücksichtigen und umzusetzen.
- Für den Zugriff auf das Backend ist ein auf JSON basierender Proxy für den gegebenen Dienst zu implementieren.

Organisatorisches Anforderungen

- Die Projektarbeit ist in Einzelarbeit auszuführen.
- Für die Umsetzung sollen nach Möglichkeit nur die im entsprechenden Lehrveranstaltungsteil vorgestellten Mittel verwendet werden.
- Die Projektarbeit ist spätestens bis zum oben angeführten Abgabetermin elektronisch abzugeben.
- Zusätzlich sind die Ergebnisse der Arbeit in einer kurzen Präsentation im Rahmen der Übung vorzustellen. Für die Präsentation ist eine kurze Live-Demo (Emulator oder echtes Endgerät) vorzubereiten.

Abzugebende Komponenten

- Ordner mit vollständig übersetzbaren Sourcen, je nach gewählter Plattform als vollständiges Eclipse- oder als Visual-Studio-Projekt
- Ausführliche Dokumentation als PDF-Datei, mindestens mit folgendem Inhalt:
    - Allgemeine Lösungsidee
    - Architektur und Struktur der entwickelten mobilen Anwendung
    - Abgedruckter Quelltext (Code, Markup, relevante Konfigurationsdaten etc.)
    - Testfälle inklusive Screenshots (Emulator)

Packen Sie alle abzugebenden Komponenten in eine ZIP-Datei (kein RAR oder anderes Archiv-Format) und laden Sie das Archiv rechtzeitig vor Abgabeschluss über die entsprechende Abgabemöglichkeit im Moodle-Kurs hoch.