Cloud Computing / IoT Plattform (Google Cloud Platform)

Inhalt

1 Cloud Computing

"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models." – NIST (National Institute of Standards and Technology)

1.1 Was ist Cloud Computing?

Cloud Computing bietet Ihnen die Möglichkeit, Rechenpower, Speicherplatz und auch Software als Service zu beziehen, ohne dass Sie die Hardware (Computer / Server) selbst besitzen. Diese Services stehen über das Netzwerk (Internet) jederzeit und überall zur Verfügung.
Folgende 5 wesentlichen Eigenschaften beschreiben eine Cloud.

1. **Selbstbedienung bei Bedarf:**
   Der Nutzer kann bei Bedarf selbst Ressourcen (CPU-Zeit, Speicher, etc.) konfigurieren.

2. **Zugriff via Internet**
   Die Dienste einer Cloud sind via Internet erreichbar.

3. **Bündelung der Ressourcen**
   Die Ressourcen einer Cloud werden in einem Rechenzentrum mit den Ressourcen anderer Kunden gebündelt. Der Kunde kennt den genauen Standort des Rechenzentrums nicht.

4. **Flexible und schnelle Anpassung an den eigenen Bedarf**
   Die Ressourcen können sehr flexibel bereitgestellt, umkonfiguriert oder abgegeben werden, zum Teil auch automatisch. Damit kann man dem Bedarf entsprechend skalieren.

5. **Exakte Abrechnung mit hoher Transparenz**
   Cloud-Plattformen überwachen die Nutzung der Ressourcen, um eine sehr genaue Abrechnung der erbrachten Dienste zu ermöglichen.

1.2 Vorteile einer Cloud

Einer der Hauptvorteile der Cloud sind sicherlich die reduzierten Kosten. Doch weshalb sind die geringer als beispielsweise bei einem lokalen Server? Das hat mehrheitlich mit der guten Skalierbarkeit einer Cloud zu tun, man bezahlt nur die Ressourcen, welche auch benötigt werden (pay-as-you-go). Während bei einem lokalen Server die Ressourcen bereits gekauft sind, verursachen sie Kosten, ob sie nun gebraucht werden oder nicht.
Dazu kommt die komplette Wartung des Servers, welche bei einer Cloud vom Cloud Anbieter übernommen wird und die davon abhängige Verfügbarkeit des Servers garantiert wird.
Nicht zu vergessen die Flexibilität, jederzeit können die benötigten Ressourcen neu konfiguriert werden, teilweise auch automatisch, beispielsweise bei einer höheren Anzahl von Zugriffen auf eine Webseite.

Auch das Thema Sicherheit muss angesprochen werden, die Hardware der Cloud ist vor verschiedenen Einflüssen (Umwelt, Personen) gut geschützt in Datenzentren verstaut. Einen Nachteil, welche die Cloud mit sich bringt, ist die Datenhoheit. Unter Umständen werden sensible Daten aus der Hand gegeben und verlassen ihre vier Wände. Beachten Sie dabei, dass je nach Standort der Cloud andere länderspezifische Gesetze gelten.

Doch wann sollten Sie in eine Cloud wechseln? Sobald Sie Skalierbarkeit, Verfügbarkeit, Bequemlichkeit und geringere Kosten benötigen.

1.3 Service Modelle

In der Cloud werden drei Service-Modelle unterschieden, Software as a Service (SaaS), Platform as a Service (PaaS) und Infrastructure as a Service (IaaS). Diese unterscheiden sich hauptsächlich darin, was der Anbieter und was der Nutzer betreut, siehe Abbildung 1. Als Vergleich dazu die Vor-Ort (On-Premises) Lösung (lokaler Server).
In den folgenden Abschnitten werden die drei Service-Modelle kurz erläutert und Beispiele genannt.

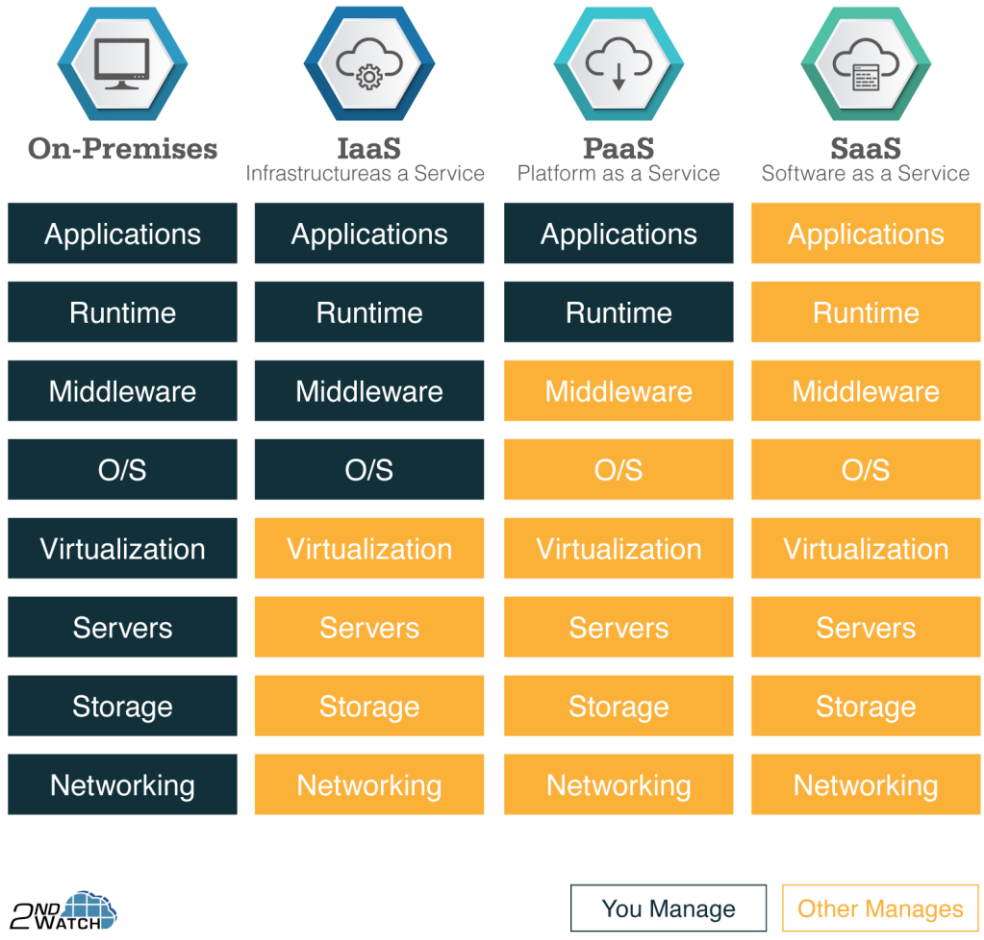

Abbildung 1: Service Modelle - https://www.2ndwatch.com/wp-content/uploads/2021/08/2W_CloudComputingServiceModels_Infographic_2021-P1.png

1.3.1 Software as a Service (SaaS)

Software as a Service ist ein Bereitstellungsmodell von Software, welche zentral gehostet wird und über das Web erreichbar ist. Dabei spielt es keine Rolle von welcher Plattform (Smartphone, Laptop, Workstation, Tablet) oder Betriebssystem die Software benutzt wird.
Eines der wohl bekanntesten Beispiele ist «Microsoft Office 365». Google bietet mit «Google Workspace» eine ähnliche Lösung an. Auch die Entwicklungsumgebung «Visual Studio Code» wird als Software as a Service angeboten und kann über das Web benutzt werden.

1.3.2 Platform as a Service (PaaS)

Mit Platform as a Service wird den Nutzern eine Plattform geboten, welche es dem Nutzer erlaubt selbst Applikationen zu erstellen und zu hosten. Die Infrastruktur selbst wird vom Cloud-Anbieter betreut.
Ein Beispiel ist die «Google App Engine», siehe Kapitel 0.

1.3.3 Infrastructure as a Service (IaaS)

Mit Infrastructure as a Service werden dem Nutzer die Rechen- und Speicher-Ressourcen mithilfe einer Instanz einer Virtuellen Maschine zur Verfügung gestellt. Es wird die Möglichkeit geboten, das Betriebssystem sowie installierte Applikationen selbst zu wählen und zu betreuen.
Ein Beispiel ist die «Google Compute Engine» oder die «Amazon Elastic Cloud (EC2)».

1.4 Bereitstellungs-Modelle

Private Cloud: Offeriert Services über das Intranet für eine spezifische Organisation. Diese Cloud gehört, wird betreut und wird bedient von der Organisation.

Community Cloud: Offeriert Services an Mitgliedern einer Community, aber nicht an die allgemeine Öffentlichkeit. Diese Cloud gehört, wird betreut und wird bedient aus einer oder mehreren Mitgliedern der Community.

Public Cloud: Offeriert Services über das Internet an die allgemeine Öffentlichkeit. Diese Cloud gehört, wird betreut und wird bedient von einem Cloud Provider.

Hybrid-Cloud: Ist eine Kombination aus mindestens einer Private Cloud und mindestens einer Public Cloud. Der Zweck ist die Anmietung von öffentlichen Cloud-Diensten, wenn die Kapazität der privaten Cloud nicht ausreicht.

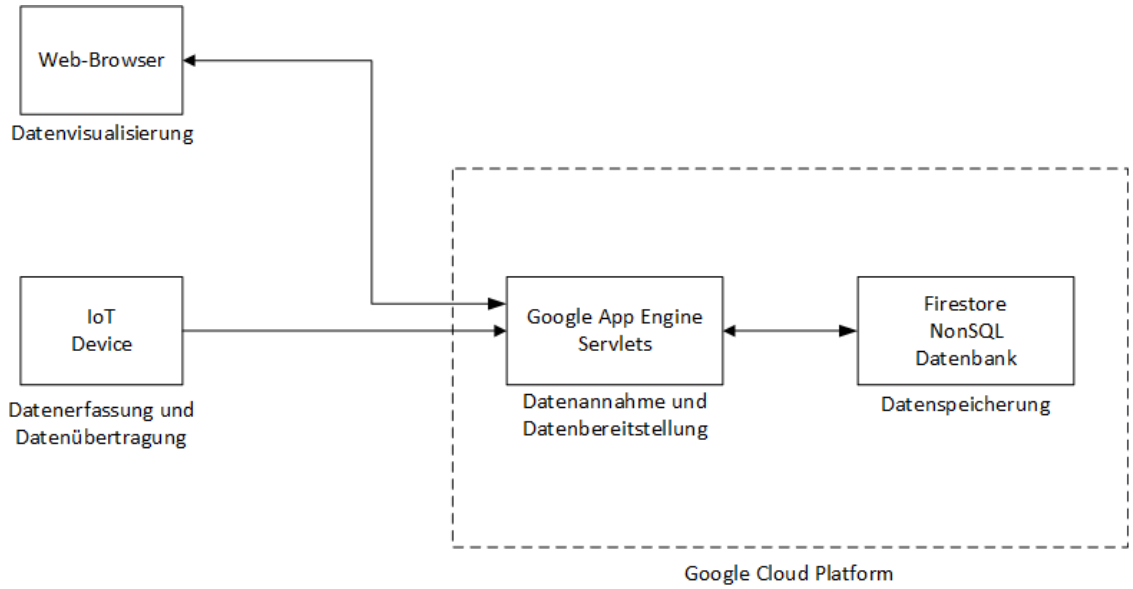

Abbildung 2: Möglicher Aufbau einer Cloud-Anwendung

Die Google Cloud Plattform enthält wie die meisten Cloud-Anbieter verschiedene Service Produkte und bietet so für viele Anforderungen eine Lösung. Gerne dürfen Sie sich bei Interesse selbst etwas durch die verschiedenen Services der Google Cloud Plattform lesen.
In **Fehler! Verweisquelle konnte nicht gefunden werden.** ist ein möglicher Aufbau einer Cloud-Anwendung dargestellt.

Folgend konzentrieren wir uns auf zwei Services, die Google App Engine (GAE) und den Google Cloud Datastore, dem Vorgänger des Firestores.

Die Google App Engine stellt eine Cloud Computing Plattform für Webapplikationen bereit, diese unter dem Service Modell Platform as a Service (PaaS). Dabei werden die Webapplikationen automatisch skaliert, das heisst den Zugriffszahlen angepasst.

Der Google Cloud Datastore ist eine der möglichen Speicher-Services, welche die Google Cloud Plattform anbietet. Diese ist eine skalierbare NoSQL Datenbank, welche vollständig vom Cloud-Provider betreut wird.

1.5 IoT Plattformen

IoT-Plattformen sind cloud-basierte Umgebungen und bieten Tools zur Entwicklung und Verwaltung von IoT-Anwendungen. IoT-Plattformen enthalten meist die folgenden Komponenten:

- MQTT Broker, HTTP Server sowie REST-Schnittstellen, um Daten von der Cloud zu lesen und in die Cloud zu übertragen.
- Datenablagemöglichkeiten (Datenbanken und Datastores) zur Speicherung von Daten.
- Tools zur Datenanalyse und Berichterstellung.
- Werkzeuge für die Erstellung von Web- und mobilen Applikationen

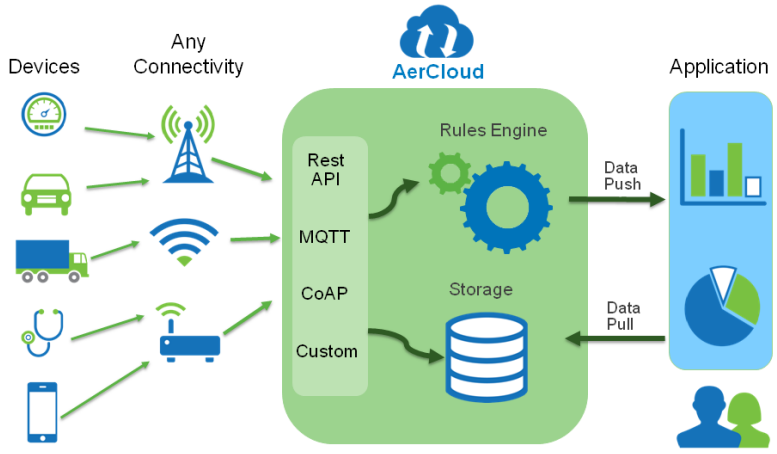

Abbildung 3: Quelle: http://www.aeris.com/technology/aercloud/

2 Aufgaben

Im GitLab der OST finden Sie das Google Cloud Projekt «GCP_Datastore» (https://gitlab.ost.ch/wn_cloudcomputing/gcp_datastore). Dieses Projekt können Sie klonen. Sofern Sie Ihre Anpassungen in GitLab speichern wollen, empfehle ich Ihnen einen Fork des Projekts zu erstellen. Ein Fork erstellt eine Kopie des Projekts und ermöglicht es Ihnen daran zu arbeiten, ohne dass das Originale Projekt beeinflusst wird.

Builden und starten Sie das Projekt gemäss dem README des Projektes. Darin ist beschrieben wie die Entwicklungsumgebung installiert wird. Der erste Build-Vorgang kann einige Minuten in Anspruch nehmen.

Die Aufgaben werden zuerst mit der lokalen Google Cloud Umgebung gelöst, bevor die eigentliche Google Cloud zum Einsatz kommt, siehe Kapitel 2.3.

Eine Lösung zu den Aufgaben finden Sie im selben GitLab-Projekt unter dem Branch «exercise-solutions».

2.1 Cloud Anwendung: Daten im Google Cloud Datastore speichern

2.1.1 Aufgabe A: Code verstehen

Öffnen Sie in einem ersten Schritt das File «SensorServlet.java».
Wie Sie bestimmt festgestellt haben, beinhaltet das File eine kleine Java-Klasse. Genauer gesagt ein Java-Servlet. Servlets sind Java-Klassen mit einer Instanz innerhalb des Web-Servers, welche URL Anfragen auf den konfigurierten Servlet-URL-Pfad empfängt und behandelt. Sollten Sie mit dem Begriff URL-Pfad noch etwas Mühe haben, in Abbildung 4 ist der Aufbau einer URL erklärt.

Abbildung 4: Uniform Resource Locator (URL) - https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_URL/mdn-url-all.png

Das Servlet erweitert die Abstrakte-Klasse «`HttpServlet`», welche einige überschreibbare Funktionen besitzt. Unteranderem die «`doGet(…)`» - Methode, welche GET-Anfragen behandelt oder die «`doPost(…)`» - Methode, welche POST-Anfragen behandelt. Die Methoden enthalten die Parameter «`HttpServletRequest`», über welchen die Informationen der erhaltenen Anfrage zur Verfügung gestellt werden und «`HttpServletResponse`», über welche die Antwort an die Anfrage zurückgesendet werden kann. Die Informationen der Anfrage enthalten auch die in der URL übergebenen Key-Value-Pairs, welche über die Methode «`getParameter(Key)`» ausgelesen werden können.

In dieser ersten Aufgabe fokussieren Sie sich auf die «`doPost()`»-Methode. Der Inhalt der «`doGet()`»-Methode wird in der zweiten Aufgabe, siehe Kapitel 2.2 beschrieben.

Der «`DatastoreService`» gibt Ihnen verschiedene Möglichkeiten, um mit dem Datastore zu interagieren. Sie haben die Möglichkeit eine neue Entität hinzuzufügen, zu aktualisieren, zu entfernen und abzufragen. Den Service erhalten Sie über folgenden Aufruf: `DatastoreServiceFactory.getDatastoreService()`.

Um einen Tabelleneintrag zu erstellen, muss eine neues «Entity (Entität)» erstellt werden. `Entity sensorData = new Entity("Sensor","Key");`
Mit dem ersten Parameter wird bestimmt, in welcher Tabelle der Eintrag ergänzt wird, bzw. wenn diese nicht existiert, so wird diese erzeugt. Mit dem zweiten Parameter kann dem Eintrag einen Key mitgegeben werden. Über diesen Key ist es möglich diesen Tabellen Eintrag spezifisch abzufragen oder zu überschreiben. Bevor Sie nun die erstellte Entität in den Datastore senden, können Sie über die Methode «`setProperty(…)`» verschiedene Attribute der Entität hinzufügen.

Bei Interesse finden Sie in der Google Cloud Dokumentation weitere Informationen zum Google Cloud Datastore und dem Handling von Entitäten.

Die Datenübertragung in die Cloud findet unverschlüsselt statt und ist deshalb für reale Anwendungen betreffend Sicherheit mangelhaft. Im Wissensnugget «IT Security» lernen Sie die wesentlichen Basiskonzept für eine sichere Kommunikation.

2.1.2 Aufgabe B: Daten in Datenstore speichern

Daten können Sie im Datastore über die erstellte Schnittstelle speichern. Beachten Sie dabei, dass das Servlet auf einen POST-Request reagiert, eine URL Eingabe im Browser erzeugt einen GET-Request. Es gibt verschiedene Möglichkeiten einen POST-Request zu erstellen, eine davon ist «Postman» oder sofern Sie mit Visual Studio Code arbeiten die Extension «ThunderClient».
Einfachheitshalber wurde im Projekt selbst eine kleine Applikation implementiert mit zu den Aufgaben entsprechenden Formularen. Über diese Formulare können die notwendigen POST- und GET-Requests ebenfalls ausgeführt werden.

1. Stellen Sie sicher das im Auswahlmenü der Applikation die Aufgabe B selektiert ist.

2. Speichern Sie Ihren ersten Sensordatensatz, indem Sie den leeren Textfelder Werte übergeben. Als Beispiel tragen Sie den Namen «Sensor» mit dem Wert «42.5» ein. Beachten Sie dabei die Statusbar ganz unten, was teilt Sie Ihnen mit?

3. Sie konnten nun bereits einige Daten im Datastore speichern. Den Inhalt des Datastore können Sie über den Dataviewer prüfen. Schauen Sie in der Konsole nach, in welcher Sie das Projekt gestartet haben, unter welchem Pfad die Adminkonsole läuft. Standardmässig finden Sie diese Konsole über den Pfad «/_ah/admin». Im Datastoreviewer können Sie folgend die Entität selektieren und die darin gespeicherten Werte darstellen lassen.
   Konnten Sie Ihre gespeicherten Sensordaten entdecken?

2.1.3 Aufgabe C: Fügen Sie dem Sensor ein neues Attribut hinzu

Neben den bereits vorhandenen Kennwerten Sensorname, Messwert und Messzeitpunkt soll nun auch noch der Ort der Messung im Datastore gespeichert werden.
Der Ort der Messung wird wie der Sensorname oder der Messwert als Parameter mit dem Key «position» in der URL übergeben.

1. Extrahieren Sie den Key «`position`» aus der URL und speichern Sie den Wert in einer Variablen «`position`» vom Typ «`string`». Zur Überprüfung können Sie den Wert mithilfe von `System.out.println(…)` ausgeben.

2. Fügen Sie dem Sensor nun das neue Attribut (Property) «`position`» hinzu.

3. Sobald Sie Anpassungen am Code vornehmen, müssen Sie das Projekt neu builden und Ausführen, um die Anpassungen verwenden zu können.

2.1.4 Aufgabe D: Neue Entität im Datastore und aktualisieren von Werten

Bis anhin werden im Datastore nur Sensoren Daten gespeichert. Nun soll auch der On/Off Status eines Motors gespeichert werden. Bei jedem Statuswechsel soll der Wert im Datastore aktualisiert werden.
Denkbar wäre es hier auch jeden Statuswechsel als eigenen Eintrag zu erfassen. Dadurch erhalten wir die Möglichkeit zu analysieren, wann und wie lange der Motor in welchem Zustand war.

1. Wie dem Sensor werden auch dem Motor Parameter mitgegeben. Speichern Sie die Parameter «`motor`» und «`state`» als Variablen. Um die erhaltenen Werte zu prüfen, können Sie diese mit Hilfe von `System.out.println(…)` ausgeben.

2. Nun erstellen wir eine neue Entität, unter welchem die Motorendaten gespeichert werden sollen. Wichtig hier, damit der Eintrag aktualisiert werden kann muss der Entität ein Key mitgegeben werden, in diesem Fall den Namen des Motors. Als Attribut (Property) setzen Sie den Motorenstatus.

3. Nicht immer sind die Sensoren Daten und die Motoren Daten im http-Request vorhanden. Prüfen Sie vor dem Erstellen eines Tabelleneintrags, beispielsweise mit einem If-Statement, ob der Parameter «`sensor`» und/oder «`motor`» existiert. Es wäre auch denkbar für den Motor ein neues Servlet zu erstellen, welches Beispielsweise auf den URL-Pfad «/data/motor» reagiert. Dadurch können Bedingungen eingespart werden und der Code bleibt etwas übersichtlicher.

2.2 Cloud Anwendung: Daten im Google Cloud Datastore abfragen

2.2.1 Aufgabe E: Code verstehen

Das Grundgerüst des Java-Servlet und der Datastore-Service wurden bereits in Abschnitt 2.1.1 beschrieben, weshalb hier nicht nochmals darauf eingegangen wird.

In dieser Aufgabe liegt der Fokus auf der «`doGet(…)`»-Methode der Klasse und dem Bereitstellen von Daten.

Um Daten aus dem Datastore abzufragen, wird ein sogenanntes «Query (Abfrage)» erstellt. `Query q = new Query("Sensor")`
Mit dem Parameter geben Sie wie beim Erstellen der Entität an, aus welcher Tabelle Sie die Daten abfragen wollen. Möchten Sie nur bestimmte Einträge der Tabelle erhalten, so gibt es die Möglichkeit von Filtern. So haben Sie beispielsweise die Möglichkeit, nur Daten aus der letzten Stunde abzufragen.
Mit der erstellten Abfrage kann über die «`prepare(query)`»-Methode des Datastore-Services eine vorbereite Abfrage erstellt werden. Dadurch wäre es möglich, sofern dieselbe Abfrage öfters gebraucht wird, die vorbereite Abfrage mehrmals zu gebrauchen, ohne ständig eine neue Abfrage zu erstellen.
Die so vorbereitete Abfrage kann wann immer notwendig ausgeführt werden. Dazu existieren verschiedene Methoden, unteranderem «`asList(…)`», welche das Resultat als Liste zurückgibt oder «`asIterator()`», welches Ihnen direkt einen Iterator zur Verfügung stellt.
Mithilfe des Iterators wird folgend über die erhaltenen Entitäten iteriert und ein JSON-Array erstellt. Anschliessend wird das erstellte JSON-Array als Antwort auf die erhaltene Anfrage zurückgegeben.

Weitere Möglichkeiten zur Datenabfrage entnehmen Sie der Google Cloud Dokumentation.

2.2.2 Aufgabe F: Filtern des Datastores mit einem URL-Parameter

Standardmässig werden die Daten der letzten Stunde zurückgegeben. Nun soll es aber möglich sein, über den URL-Parameter «getLastXMinutes» zu definieren, bis wann die Daten zurückgegeben werden sollen.

1. Extrahieren Sie den Key «`getLastXMinutes`» aus der URL und speichern Sie den Wert in einer Variablen des Typen «`int`». Dazu muss der Wert über «`Integer.parseInt(…)`» geparst werden.
2. Bauen Sie den Abfrage-Filter so um, dass dieser den Wert aus der URL verwendet. Sollte dieser kein gültiger positiver Integer sein, so verwenden Sie den Standardwert von 60 Minuten.

2.3 Deployment on Cloud

Bis anhin konnten alle Aufgaben in der lokalen Variante der Google Cloud ausgeführt werden. Für die folgende Aufgabe benötigen Sie einen Google Cloud Account.

1. Prüfen Sie in einem ersten Schritt ob gcloud installiert ist, schreiben Sie dazu in der Konsole «`gcloud -v`». Ist es nicht vorhanden, so installieren Sie gcloud. Folgend können Sie über «`gcloud auth login`» ihren Google Cloud Account authentifizieren.

2. Erstellen Sie in der Google Cloud ein neues Projekt. Ersetzen Sie die Projekt-Id im «`pom.xml`» in der folgenden Zeile mit Ihrer eigen erstellten Projekt-Id.
   ```
   <deploy.projectId>wn-cc-datastore</deploy.projectId>
   ```
   Über «`gcloud config set project project_id`» hinterlegen Sie Ihre Projekt-Id zusätzlich noch dem gcloud-Service.

3. Erstellen Sie in einem nächsten Schritt eine AppEngine-Anwendung. Dies können Sie über die Konsole «`gcloud app create`» oder über die Google Cloud App Engine, in dem Sie eine neue Anwendung erstellen. Beim Erstellen der Anwendung müssen sie die Position der Cloud wählen, als Beispiel der Standort Zürich (europe-west6).

4. Nachdem Sie nun alles vorbereitet haben, können Sie ihr Projekt mit «`mvn appengine:deploy`» in die Google Cloud deployen. In der Konsolenausgabe finden Sie viele hilfreiche Informationen, unter anderem auch die URL, unter welchem das Projekt erreichbar ist. Das Projekt läuft nun in der Cloud und ist nicht mehr an Ihr Gerät gebunden.

5. Schreiben Sie verschiedene Daten über das Formular in den Datastore und prüfen Sie diese anschliessend im Datastore der Cloud.

