Änderung Interface Beschreibung Backend <-> Python

Abstract

Diese Beschreibung ist Teil der Schnittstelle zwischen dem Backend und dem Python Web Service zur Generierung von Statistikcharts. Die Schnittstelle wird verwendet um Daten des Python Web Services zu Speichern und auf bestimmte Schnittstellenparameter eine differenzierte Verhaltensweise zu erzeugen. Die Schnittstelle selbst basiert auf JSON, welches vom Python Web Service erzeugt wird. Als Ziel soll es möglich sein Seriencharts zu erzeugen, in denen man mithilfe der auf Angular 9 basierten Oberfläche bequem zwischen Zeiträumen wechseln kann. Durch die vorgeschlagene Änderung muss nichts am Input in den Python Web Service geändert werden.

Verwendungszweck

Die bisherige Schnittstelle muss erweitert werden, um mehr funktionalität zur Verfügung zu stellen. Bisher war die Schnittstelle nur darauf ausgelegt, Datensätze im Gesamten entgegenzunehmen, sowie Layoutinformationen und ein Aktualisierungsdatum. In der Änderung sollen weitere Möglichkeiten zur Darstellung und Kontrolle möglich sein.

Bisherige Struktur

Die bisherige Struktur beinhaltet folgendes:

- Titel
- Traces (Datensätze, welche angezeigt werden)
- Layout (Das Aussehen des Graphen)
- updateTime (Die Zeit des letzten Updates aus Sicht des Python Web Services)

```
{
    "title" : "",
    "traces" : [ ],
    "layout" : { },
    "updateTime" : ""
}
```

Mit dieser Struktur ist es nicht möglich Seriencharts zu erzeugen, welche über die Angular Oberfläche durchklickbar sind.

Mögliche Umsetzungen

Plotly

Es wäre möglich in Plotly einen Wechsel der Monate über Schaltflächen zur Verfügung zu stellen.

Vorteile:

- Keine Änderung am Interface/Backend/Frontend

Nachteile:

- Aufwändige Anpassung der bestehenden Python Skripte
- Die Python Scripte sind Unübersichtlicher
- Kein einheitliches Design
- Höhere Netzwerklast, da immer alle Daten an das Frontend geschickt werden müssen.
- Längere Ladezeiten auf Client-Seite, da Plotly im Webbrowser die Anzeige jedes mal neu generiert.

Interface Anpassung (Backend/Frontend/Python)

Die Anpassung des Interfaces gibt die Möglichkeit auch für zukünftige Änderungen bereits Werte vorzusehen. Bei einer Anpassung soll es Möglich sein, abwärts kompatibel zu bleiben. Durch die Änderung des Interface müssen nicht mehr Informationen generiert und übertragen werden, da auf bestimmte Parameter reagiert werden kann und beispielsweise ein Kalender

Vorteile:

- Einheitliches Design
- Entlastung des Client Browsers, durch kleinere Datenmengen
- Niedrigere Netzwerklast
- Keine Anpassung bestehender Python Skripten nötig, da Abwärtskompatibel

Nachteile:

- Änderung in allen Bereichen
- Höhere Auslastung des Backend-Servers (minimal)
- Häufigeres Nachladen, wenn Ansicht gewechselt wird

Neutral:

- Bei Wechsel der Ansicht muss das Chart neu geladen werden, fällt aufgrund der kleineren Datenmenge aber nicht ins Gewicht.

Mögliche neue Struktur

Aufgrund der Menge der Vorteile der Interface Anpassung im gegensatz zur Generierung der Timeseries Funktionalität im Client mit Plotly wird ein erweitertes Interface angestrebt.

Ein erster Entwurf der neuen JSON Struktur.

Hinweis zur Verwendung von zusätzlichen Zeichen

Optionale Parameter werden mit `*` eingeklammert.

Das Zeichen `|` gibt an, welche verschiedenen Möglichkeiten es gibt.

Mit `#` ist ein Default Wert markiert.

Werte in `<>` müssen durch Zahlen ersetzt werden.

```
{
    "title" : "",
    "traces" : [ ],
    "layout" : { },
    * "updateTime" : "", *
    * "timeseries" : true|#false, *
    * "accuracy" : "year"|"month"|"day"|"week"|#"none", *
    * "multiple" : true|#false, *
    * "options" : [], *
}
```

Definition des "traces"-Arrays:

1. In der bisherigen Form. Hierbei werden die gesamten Daten in das Traces Array gegeben. Da die Optionalen Werte `timeseries : false`, `accuracy : none` und `multiple : false` gesetzt sind, ändert sich am Verhalten nichts. Die Traces sind damit identisch mit denen, die auch von Plotly verwendet werden.

```
"traces" : [
    {
        "x" : [234,324,3232],
        "y" : [34,32,32]
    }
]
```

2. Um das neue Format nutzen zu können, muss `timeseries : true` sein und `accuracy` ungleich `none`, sonst wird das erste Format angenommen und es kann zu Problemen in der Ausgabe führen, da Plotly die Daten nicht versteht! Im Traces array wird nun für jede Zeiteinheit ein Objekt erzeugt. `"time : <year>-<month>-<day>"` ist so definiert, dass anhand der `accuracy` bestimmt wird wie genau time definiert wird. Wenn `"accuracy" : "year"` dann ist `"time" : "<year>"` zu setzten. Zusätzlich kann die Option `"accuracy" : "week"` gesetzt werden. Damit können Charts mit der Genauigkeit von Wochen als Timeseries verwendet werden, indem die Woche als Zeit mit geben: `"time" : "<year>-<weekNumber>"`. Das folgende Beispiel erläutert den Fall für `"accuracy" : "day"`:

```
"traces" : [
    {
        "time" : "<year>-<month>-<day>",
        "timetraces" : [],
    }
]
```

3. Der Parameter `options` wird aktuell noch nicht verwendet. Wird aber für zukünftige Anpassungen der Schnittstelle verwendet.

Änderungen im Backend

Da die Daten im Backend gecached werden, ist es nötig auch hier Änderungen durchzuführen. Hierfür muss das JSON welches vom Python Web Service geschickt wird analysiert werden und die Einträge entsprechend in der Redis Datenbank gespeichert werden.

Idee

Hierbei empfiehlt es sich mit den bisher im Redis gespeicherten Daten in eine zweite Instanz auf dem Server zu wechseln, so dass alle Statistik Daten gekapselt von den restlichen Daten liegen.

Aktuell werden die Daten in folgendem Format gespeichert:

- `statistic.allChartNames`: Beinhaltet alle Charts nach Gruppen sortiert.
- `statistic.chart.<chartName>.data`: Beinhaltet die Daten, welche die Scripts zurückliefern.
- `statistic.chart.<chartName>.script`: Speichert den Python Script Namen, welcher an den Python Web Service übergeben wird, um Daten zu erzeugen.

Um die neuen Daten zu speichern soll folgendes Format verwendet werden:

- `statistics.<chartName>.data.<year>-<month>-<day>`
- `statistics.<chartName>.data`

Im Objekt `statistic.allChatNames` soll nun zusätzlich das `script` sowie die neuen daten für `timeseries`, `accuracy` und `multiple` gespeichert werden. Die Informationen über welchen Zeitraum Daten gespeichert sind, kann mithilfe von Redis gesucht werden.

Zusätzlich muss die API in Richtung Frontend erweitert werden, damit Timeseries spezifische Informationen vom Backend geholt werden können.

Änderungen im Frontend

Im Zuge der Änderungen sollte es auch möglich sein, ein Chart anzupassen ohne dieses löschen und neu anlegen zu müssen. Dafür wird noch ein neuer Dialog benötigt, der sich darum kümmert.

Für die Timeseries Funktionalität wird zusätzlich ein Kalender und Buttons benötigt. Mit der Kalender Funktion soll es möglich sein, zu definieren, welche Daten angezeigt werden. Ist die Option `multiple : false` gesetzt, so ist es nur möglich einen bestimmten Monat anzuzeigen. Mit `multiple : true` ist es möglich einen Range an Daten zu wählen. Wenn die Timeseries Funktionalität nicht zur Verfügung steht, so soll der Kalender und die Buttons auch nicht angezeigt werden, damit bleibt das bisher bestehende Verhalten erhalten.

Beispiele

Im Folgenden werden einige Beispiele für die JSON Schnittstelle gegeben:

Einfacher Plot ohne Timeseries

```
{
    "title" : "Titel",
    "traces" : [
```

```
        {
            "x" : [234,324,3232],
            "y" : [34,32,32]
        }
    ],
    "layout" : {
        <Layout infos>
    },
}
```

Plot mit Timeseries, Accuracy year und multiple false

```
{
    "title" : "Titel",
    "traces" : [
        {
            "time" : "2020",
            "timetraces" : [
                {
                    "x" : [234,324,3232],
                    "y" : [34,32,32]
                }
            ],
            "time" : "2019",
            "timetraces" : [
                {
                    "x" : [322,322,3212],
                    "y" : [37,36,34]
                }
            ],
        }
    ],
    "layout" : {
        <Layout infos>
    },
    "updateTime" : "<timestamp>",
    "timeseries" : true,
    "accuracy" : "year",
    "multiple" : false,
}
```

Plot mit Timeseries, Accuracy day und multiple true

```
{
    "title" : "Titel",
    "traces" : [
        {
            "time" : "2020-06-24",
            "timetraces" : [
```

```
                {
                    "x" : [234,324,3232],
                    "y" : [34,32,32]
                }
            ],
            "time" : "2020-06-25",
            "timetraces" : [
                {
                    "x" : [322,322,3212],
                    "y" : [37,36,34]
                }
            ],
        }
    ],
    "layout" : {
        <Layout infos>
     },
    "updateTime" : "<timestamp>",
    "timeseries" : true,
    "accuracy" : "day",
    "multiple" : true,
}
```