Version
1.0
Document
Technical Challenge
Date
02/09/2024

SQLI
Technical Challenge

Software Development
Candidate

SQLI
DIGITAL
EXPERIENCE

Dear Victor,

We enjoyed talking to you and meeting you online, so we are thrilled to move forward and get to know your skills even better.

For the next step, we would kindly like to ask you to prepare a solution for a Technical Challenge, that specifically focuses on web API Development.

You can present your findings any way you like, this can be a private GitHub, compressed documents, WeTransfer, … As you prefer.

If you have any questions, do reach out to us.

We look forward to receiving your solution and hope to welcome you at SQLI!

You find the Challenge on the next pages.

Dry cleaning service

SQLI dry-cleaning company is known for its speedy service. It guarantees to dry-clean anything within two business hours or less. The problem is when the customer drops off the clothes, he needs to know what time they are guaranteed to be done.

SQLI DIGITAL EXPERIENCE

1. Assignment

It is your job to write a Web API that will determine the guaranteed time given a business hour schedule.

The API must contain different endpoints to allow the configuration of the opening and closing time for each day. By default, the store is in business 7 days a week. You can use additional libraries, as long as you bundle them (and explain why).

The API should at least provide the following endpoints:

You must be able to update the opening and closing hours like this:

```
DaysSchedule(string openingHour, string closingHour)
```

The instance allows for further customization of the opening hours by specifying the day of the week or a specific date.

```
Days(string dayOfWeek, string openingHour, string closingHour)
```

And for a specific date:

```
Dates(string date, string openingHour, string closingHour)
```

You must also be able to specify that the store is closed for a specific day of week or date.

```
DaysClose(string daysOfWeek)
DatesClose(string dates)
```

The close endpoints accept multiple arguments (comma separated):

```
Sunday, Wednesday
2010-12-25, 2010-11-15
```

Also, all parameters should be passed by a simple Uri string:

Curl

```
curl -X 'GET' \
  'http://localhost:5180/schedule-calculator?minutes=120&date=2010-06-07%2009%3A10' \
  -H 'accept: text/plain'
```

Request URL

```
http://localhost:5180/schedule-calculator?minutes=120&date=2010-06-07%2009%3A10
```

The **`Days`** method should change the opening and closing time for a given day. The **`DaysClose`** and **`DatesClose`** methods should specify which days the shop is not open. Notice days should be a string for specific dates (**`Sunday, Monday, etc. for days and for dates must have this format: yyyy-mm-dd`**). Any given day can only have one opening time and one closing time there are no off-hours in the middle of the day.
The date parameters should be in a 24h format, such as “19:00”.

Finally, the endpoint **`Get(int minutes, string date)`** determine the resulting business time given a time interval (in minutes) along with a starting datetime (as a string). The returned object should have the next format: `Mon Jun 07 11:10:00 2010`

2. Examples

Given the following configuration:

```
DaysSchedule("09:00", "15:00");
Days(friday, "10:00", "17:00");
Dates("2010-12-24", "8:00", "13:00");
DaysClose(Sunday, Wednesday);
DatesClose("2010-12-25");
```

The business hour calculator will yield the following results:

```
// example #1
http://localhost:5180/schedule-calculator?minutes=120&date=2010-06-07%2009%3A10
Get(120, "2010-06-07 09:10");
// => Mon Jun 07 11:10:00 2010

// example #2
http://localhost:5180/schedule-calculator?minutes=15&date=2010-06-08%2014%3A48
Get(15, "2010-06-08 14:48");
// => Thu Jun 10 09:03:00 2010

// example #3
http://localhost:5180/schedule-calculator?minutes=420&date=2010-12-24%206%3A45
Get(420, "2010-12-24 6:45");
// => Mon Dec 27 11:00:00 2010
```

In the first example the time interval is 2 hours (120 minutes). Since the 2 hours fall within business hours the day does not change, and the interval is simply added to the starting time.

In the second example an interval of 15 minutes is used. The starting time is 12 minutes before closing time which leaves 3 minutes remaining to be added to the next business day. The next day is Wednesday and therefore closed, so the resulting time is 3 minutes after opening on the following day.

The last example is 7 hours (420 minutes) which starts before opening on Dec 24th. There are only 5 business hours on Dec 24th which leaves 2 hours remaining for the next business day. The next two days are closed (Dec 25th and Sunday) therefore the deadline is not until 2 hours after opening on Dec 27th.

As a last step, please attach a document (can be inside the solution) that includes the full API Endpoint to call them (configuration and calculation), for example:
The Endpoint that returns the deadline: http://localhost:5180/schedule-calculator