LectioScraper

Informationsteknologi B

Eksamensprojekt

I dette projekt har jeg udviklet et program, der er i stand til at *scrape* skemaet fra lectio.dk og uploade det til Google Calendar og en MySQL-database. Samtidig reflekterer jeg over de valg, jeg har gjort undervejs gennem produktudviklingen, samt hvilke juridiske og sikkerhedsmæssige risici, der kan være ved mit program og web-scraping generelt.

Andreas Fiehn
3.b2, H. C. Ørsted Gymnasiet - Lyngby
Informationsteknologi, B
Lærer: Kristian Krabbe Møller
16-04-2018
Antal normalsider: 19

Underskrift

Indhold

Indledning 3
Produktkrav 3
Struktur 4
Produktet 5
Hvad er scraping? 5
Valg af sprog 5
Værktøjer 6
Brackets 6
Wamp server 6
PHP Simple HTML DOM Parser 7
JQuery 8
Bootstrap 8
Google calendar API 10
Fullcalendar 15
Moment.js 17
Font Awesome 17
Github 18
Kodeopbygning 19
Klasser 20
Flowchart 25
Layout 28
Sikkerhed 29
DDoS 29
SQL-injections 30
Google API 30
Hvad er tilladt? 31
Demonstration og test af produktet 32
Vurdering og perspektivering 33

Forslag til forbedring og videreudvikling 33
Kilder 35
Værktøjer 35
Rapporten 35
Produktet 35
Litteraturliste 35
Bilag 38
Koden 38
Opsætning af database 38
index.html 39
script.js 41
Lesson.class.php 43
LessonFullcalendar.class.php 51
LessonGoogleCalEvent.class.php 57
LectioScrape.class.php 62
setDatesSession.php 68
uploadToGoogleCal.php 68
oauth2callback.php 73
sendScheduleToDatabase.php 73
fullcalendarFeed.php 74

Indledning

Vores samfund bygger på data. En stor del af denne data er tilgængelig på internettet, og kan tilgås igennem en browser. Men hvad hvis denne data er uorganiseret, og der kun er begrænset adgang til den?

Jeg ser Lectios skoleskemasystem som forældet. Skemadataene er offentlige, men man er begrænset ved, at man kun kan tilgå deres skema på lectio.dk. De har ingen officiel smartphone-applikation, ingen velfungerende mobilside, og skemaet ser generelt ensformigt og kedeligt ud. Skemadataene er tilgængelige, men ikke organiseret, som man kunne ønske sig. Planlægning ville blive langt nemmere, hvis Lectio tilbød, at man kunne synkronisere sit skema med andre kalendertjenester som Google Calendar. Lectio har nogle mangler, som jeg vil forsøge at eliminere med et stykke software, der *scraper* skemaet fra Lectios side og organiserer det på en ny måde.

Produktkrav

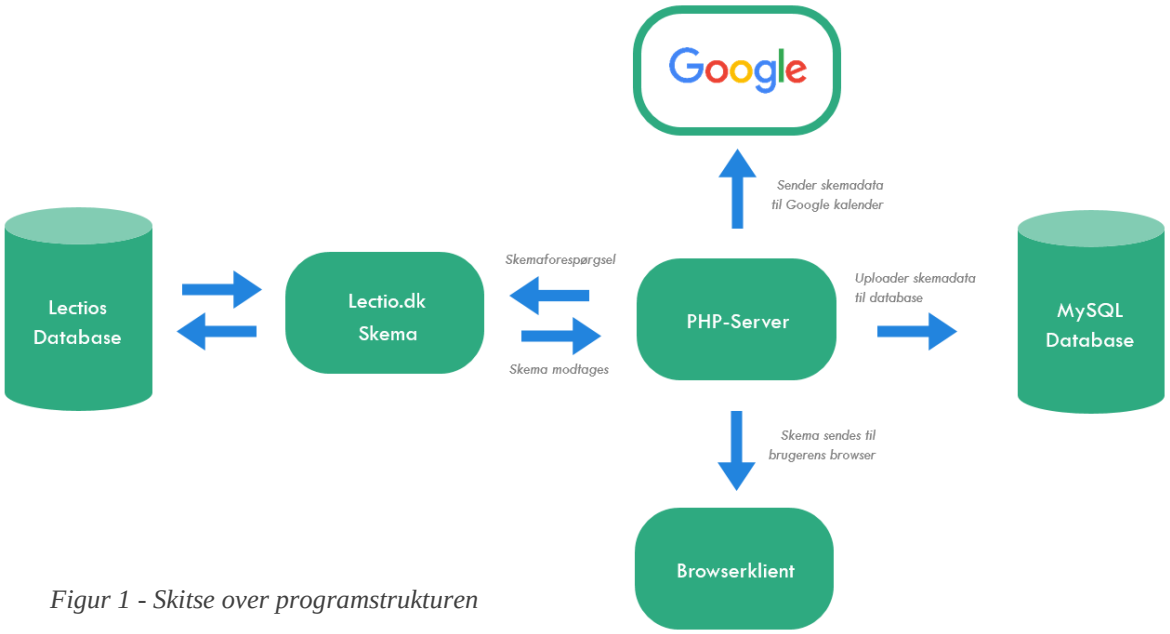
Jeg har opstillet en række krav til mit program. Dette er de krav jeg vil tilstræbe, at mit produkt opfylder.

- Mit program skal kunne *scrape* skemadata fra Lectio.
- Skemaet skal vises på en præsentabel måde til brugeren, hvor timerne er farvekodet i stedet for det ensformige design, man finder på Lectios hjemmeside.
- Skemaet skal kunne uploades til Google Calendar
- Skemaet skal uploades til en database.

Der er rige muligheder for videreudvikling og udbygning med andre funktioner. Mange af disse muligheder har jeg dog været nødt til at fravælge for at få en passende afgrænsning af produktet. Jeg har reflektereret over videreudviklingsmulighederne i afsnittet vurdering og perspektivering til sidst i rapporten.

Struktur

Jeg har på Figur 1 opstillet en figur over, hvordan jeg forestiller mig, at mit program skal fungere.



Figur 1 - Skitse over programstrukturen

Jeg antager, at Lectio har en database stående et sted, som tilgås, når brugeren besøger deres side. Denne database er brugeren ikke i stand til direkte at tilgå, og jeg har derfor også kun mulighed for at hente skemadataene ved at besøge webadressen. Min server skal hente skemasiden og behandle den, så den kan uploades til Googles kalendersystem. Udover kalenderen vil jeg også uploade skemaet til en MySQL-database, samt vise skemaet til brugeren i browseren på en tiltalende måde. Jeg vil i første omgang ikke hente dataene tilbage fra hverken Google Kalender eller databasen, hvilket er årsagen til, at pilene på min skitse kun peger i én retning.

Produktet

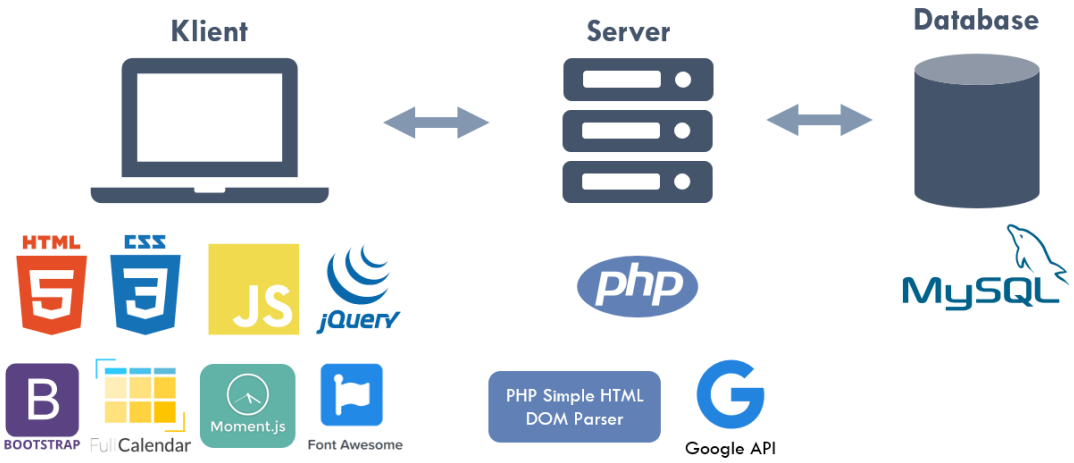
Hvad er scraping?

Indhold på Internettet tilgås oftest igennem en browser, som uden tvivl er et nyttigt redskab. Problemet er dog, at browseren bestyres manuelt, hvilket kan gøre indsamling og sortering af data på internettet meget tidskrævende. Her kommer *scraping* ind i billedet, da det kan bruges til at automatisere indsamling og tolkning af data på internettet. *Scraping* handler om at *udplukke* brugbart data fra hjemmesider. Dette kan gøre på mange forskellige måde.

Ofte har virksomheder udviklet værktøjer, som gør det nemt at tilgå deres data, men i andre tilfælde er man nødt til at ty til andre metoder for at hente data. En af de simpleste metoder til scraping af web-data er manuel copy og paste fra hjemmesiden, hvilket jeg helst vil undgå. Andre metoder handler om automatisk at hente HTML-siden og fortolke den, så data kan udtrækkes fra siden. Det er også muligt at *scrape* det visuelle output fra en hjemmeside og oversætte det til et brugbart format. Der findes utallige muligheder for at indsamle data fra internettet.

Valg af sprog

Jeg har valgt at skrive hovedparten af mit program i PHP. PHP er et serverbaseret sprog, og grunden til, at jeg har valgt netop PHP frem for andre sprog som Java eller Python, er, at jeg har tidligere erfaring med PHP, og det er nemt at sætte op på sin computer med det rette software. På klientsiden benytter jeg HTML og CSS samt JavaScript. HTML er et såkaldt *markup-language*, der bruges til at beskrive strukturen af websiden. CSS er et *stylesheet-language*, og det bruges til at definere udseendet på hjemmesiden. JavaScript er uden tvivl det mest populær programmeringssprog, når det kommer til klientbaseret webprogrammering, hvis man altså ikke medregner HTML og CSS som reelle programmeringssprog. Grunden til at jeg har valgt disse sprog til klientsiden af min webapplikation, er at de er det mest anvendte og kompatibiliteten er stor. Jeg anvender en MySQL-database, da det er nemt at sætte op, og der findes meget dokumentation for, hvordan det virker. Jeg har på Figur 2 lavet en skitse over mit program og hvilke sprog samt tilføjelser, jeg bruger på henholdsvis klientsiden, serversiden og databasesiden. Jeg kommer nærmere ind på tilføjelserne senere i rapporten.



Figur 2 - Programstruktur med tilhørende programmeringssprog og tilføjelser

Værktøjer

Brackets

Jeg har valgt at skrive mit program i Brackets, som er et opensource tekstredigeringsværktøj. Brackets er skrevet i JavaScript og har et stort udviklingsfællesskab, hvilket betyder, at der findes mange tilføjelser til programmet. Brackets er optimalt til web-udvikling, hvilket er en af grundende til, at jeg har valgt det som mit udviklingsmiljø.

Wamp server

For at lave mit program uden at skulle hoste en server, har jeg valgt at benytte WampServer, som er et stykke software, der gør det muligt at køre en Apache-server lokalt på min computer. Årsagen til, at jeg har valgt Wamp frem for mere populære alternative som XAMPP er Wamp's simplicitet. Når WampServer installeres på computeren er det klar til brug efter installationen. PHP og MySQL er inkluderet i installationen, og samtidig følger der også en række andre tilføjelser med. Den tilføjelse, jeg nok gør mest brug af, er PCRE (Perl Compatible Regular Expression), som er et bibliotek med funktioner, der gør det muligt at skrive *regular expressions* med samme syntaks som i Perl 5, som er en

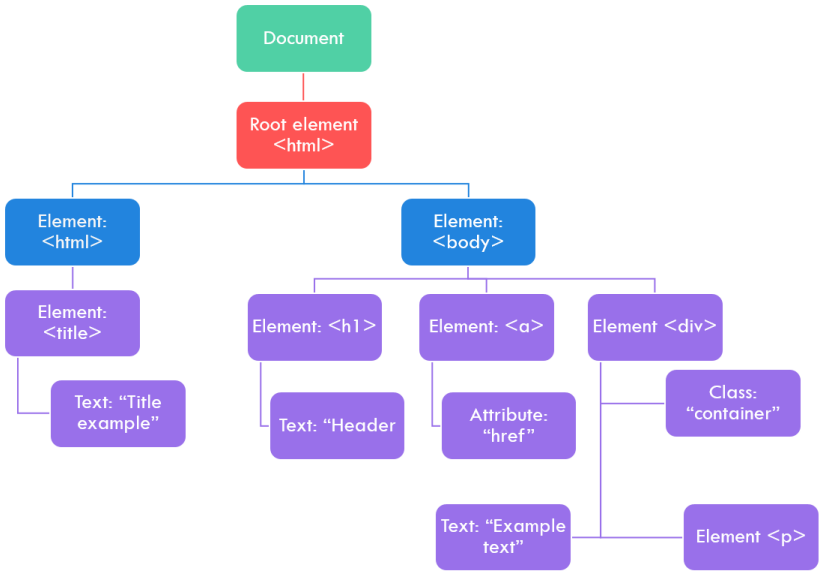
relativt populær syntaks til *regular expressions*. Jeg kommer nærmere ind på, hvordan jeg har anvendt *regular expressions* senere i opgaven, da det er en helt central del af mit program.

PHP Simple HTML DOM Parser

For at hente skemasiden gør jeg brug af et simpelt open-sourcebibliotek - PHP Simple HTML DOM Parser [1]. For at forstå dette bibliotek er man nødt til at have en smule kendskab til HTML DOM.

HTML DOM står for Document Object Model og er noget man oftest benytter i JavaScript.

Når browseren indlæser en side, laver den en såkaldt HTML DOM-model af siden. Denne model kan sammenlignes med en træstruktur, som vist på Figur 3. Modellen gør det muligt dynamisk at opdatere HTML-siden med JavaScript.



Figur 3 - Illustration af HTML DOM struktur. Inspiration hentet fra: https://www.w3schools.com/js/js_htmldom.asp

PHP Simple HTML DOM Parser henter en ønsket HTML side og konstruerer et objekt, der i strukturen minder om HTML DOM-modellen. Det er nu nemt at tilgå de forskellige elementer fra HTML-siden.

[1] Chen, "PHP Simple HTML DOM Parser".

På hjemmesiden for Simple HTML DOM Parser sammenligner de det med jQuerys måde, hvorpå man kan finde HTML-tags ved hjælp af id, klasser og lignende.

Jeg har inkluderet biblioteket ved hjælp af PHP's *require_once*-funktion, som inkluderer kode, hvis det ikke allerede er inkluderet.

```
        //Includes simple_html_dom library
        require_once __DIR__.'/simple_html_dom.php';
```

Typisk vil man i PHP bruge `file_get_contents($url)`, når man skal hente indholdet af et HTML-dokument fra en webside. Ulempen ved denne funktion i PHP er, at den returnerer en ustruktureret tekststreng, som er svær at bruge til noget. Simple HTML DOM Parser gør det muligt at anvende funktionen `file_get_html($url)`, der kan hente siden som et objekt, der er struktureret, så det er nemt at finde det indhold, man skal bruge.

JQuery

jQuery er en tilføjelse til JavaScript, der gør det nemmere og mere belejligt at opdatere HTML-sider dynamisk. Kort forklaret gør jQuery det nemmere at tilgå og manipulere HTML-objekter. jQuery gør brug af DOM-modellen som beskrevet tidligere.

Reelt set kan jQuery ikke noget, som man ikke kan gøre i JavaScript alene. jQuery gør det blot meget nemmere at arbejde med dynamisk HTML. I mit produkt gør jeg ikke så meget brug af jQuerys funktioner, men nogle af de andre tilføjelser, jeg anvender, som FullCalendar, Moment.JS og Bootstrap kræver at jQuery er inkluderet. jQuery-biblioteket kan hentes som en JavaScript-fil, og jeg har inkluderet det på min html-side ved at referere til stien, hvor biblioteket ligger.

```
    <script src='lib/jquery.min.js'></script>
```

Bootstrap

Bootstrap er et såkaldt CSS-framework, der gør det nemmere at opbygge et pænt layout på websider. Bootstrap indeholder både CSS-stylesheets og JavaScript og det inkluderes på samme måde som jQuery. Herefter kan man anvende deres CSS-klasser og funktioner, som man ønsker.

```
    <!--Bootstrap CSS-->
    <link rel="stylesheet" href="Bootstrap/css/bootstrap.css">
    <!--Bootstrap JS-->
    <script src="Bootstrap/js/bootstrap.js"></script>
```

På Bootstraps hjemmeside kan man finde dokumentation til deres biblioteker, og noget af det, der gør det meget simplere at lave layout med et CSS-framework som Bootstrap, er deres grid-system. Siden er delt op i en form for gitter bestående af 12 kolonner. Ved så at tilføje klasser til sine `<div>`-tags, kan man nemt positionere sine elementer på siden efter dette gittersystem.

Der medfølger også klasser til nemt at lave navigationsbjælker, knapper, modals og anden styling. Jeg anvender bl.a. modals, som er en form for pop-up, når jeg skal vise ekstra skemainformationer efter klik på et kalenderevent. Nedenstående ses et udsnit fra min HTML-kode, som viser anvendelsen af Bootstrap-klasserne til oprettelsen af en navigationsbjælke øverst på siden.

```
    <!--Navbar-->
    <nav class="navbar navbar-expand-lg navbar-light bg-light">
        <a class="navbar-brand" href="#">LectioScrape</a>
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle
navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarNav">
            <ul class="navbar-nav">
                <li class="nav-item active">
                    <a class="nav-link" href="#">Skema <span class="sr-only">(current)</span></a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="about.html">Om siden</a>
                </li>
            </ul>
        </div>
    </nav>
```

Bootstrap er udgivet under MIT-licensen, hvilket giver mig lov til at bruge deres software kommercielt såvel som privat [2].

[2] Bootstrap, “License FAQs”.

Google calendar API

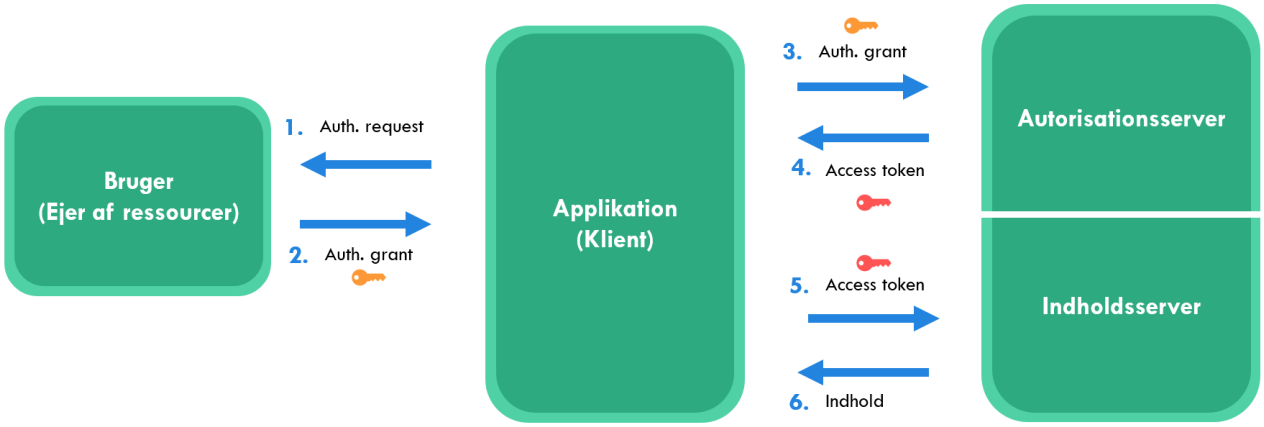
Som beskrevet under produktkravene, skal jeg uploade mit lectioskema til Google Calendar, så man kan holde bedre styr på sin planlægning gennem hele dagen. Dette kræver, at jeg har adgang til Google Calendar, og jeg har mulighed for at oprette kalendere samt slette og oprette begivenheder på kalenderen. Til dette bruger jeg Googles API.

API står for Application Programming Interface og er kort forklaret blot et stykke kode, der tillader, at to systemer arbejder sammen. I dette tilfælde er det min PHP-server der skal interagere med Googles kalendersystem. Jeg har lavet en simpel illustration af, hvordan en API fungerer på Figur 4.

Figur 4 - Illustration af API. To systemer er i stand til at arbejde sammen, da API'en gør dem "kompatible".

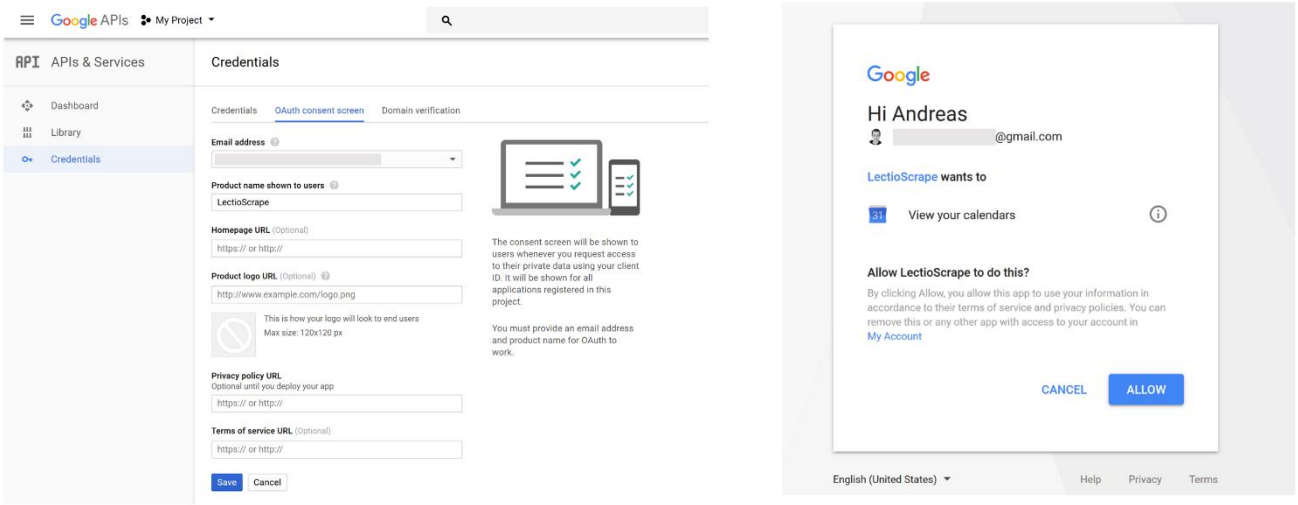
Når jeg skal have adgang til brugerens kalender, er der noget sikkerhed, man skal igennem. Her anvender Google Oauth 2.0, som er et system til adgangstilladelse [3]. Det fungerer ved, at klienten sender en forespørgsel til brugeren - et såkaldt *request* - og brugeren kan vælge om han vil give tilladelse eller nægte adgang. Såfremt der er givet tilladelse sendes dette til applikationen (*grant*), som sender det videre til serveren sammen med sin egen hemmelige kode. Serveren sender en såkaldt *access token* tilbage, som applikationen herefter kan bruge til at hente indhold. Den *access token*, der er blevet genereret giver kun adgang til det indhold, der er blevet forespurgt, og det giver kun adgang til den specifikke applikation, der har forespurgt adgang. Jeg har på Figur 5 lavet en skitse over Oauths virkemåde.

[3] D. Hardt Ed., "The OAuth 2.0 Authorization Framework".



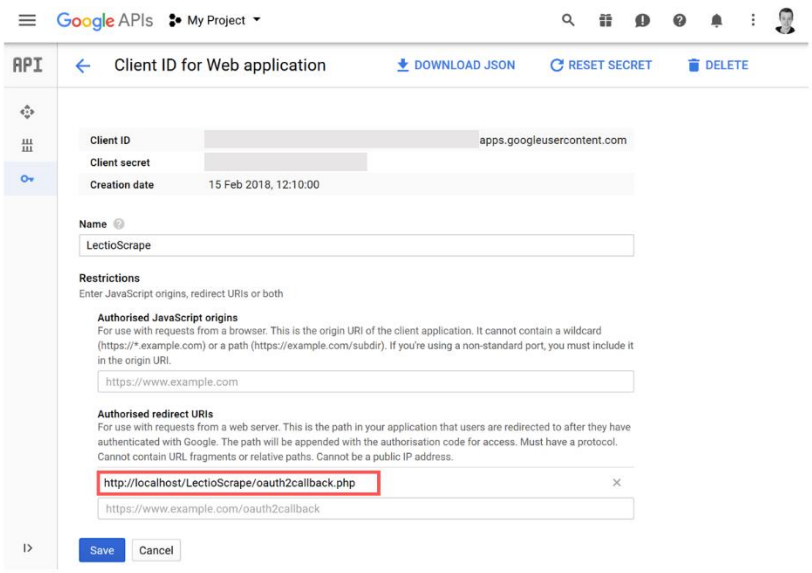
Figur 5 - Figur over Oauths virkemåde

For at kunne tilgå Googles system, skal man logge ind på Googles API-konsol, hvor Oauth-adgangen kan opsættes. Herinde skal man først lave et nyt projekt, hvorefter man så kan tilføje adgang til de forskellige Google-biblioteker under Library. Her har jeg kun brug for adgang til Google Calendar API. Herefter sætter jeg en *"OAuth consent screen"* op hvilket jeg har vist på Figur 6. Denne såkaldte *"consent screen"* er det brugeren vil se, når de tillader adgang til deres kalender. I første omgang tilføjer jeg ikke noget billede eller nogen url på *consent* siden.



Figur 6 - Oprettelse af Google Oauth consent screen (venstre) og et eksempel på denne consent screen (højre).

Herefter skal jeg oprette *credentials*, så jeg får en nøgle, der identificerer min applikation og senere giver mig adgang til brugerens kalender, efter de har givet mig tilladelse. Her er det vigtigt, at jeg opretter et såkaldt redirect-URI, som er den side brugeren bliver ført hen til, efter de har logget ind og givet tilladelse. Jeg fører brugeren hen til `localhost/oauth2callback.php`, hvilket jeg har vist på Figur 7.



Figur 7 - Oprettelse af credentials og tilføjelse af redirect URI

Når alt dette er sat op kan jeg downloade min *client-secret*, som kort forklaret er den nøgle der identificerer min applikation. Jeg downloader denne som en JSON-fil som jeg så kan referere til, når jeg anvender Googles API. Denne JSON-fil gemmer jeg på serveren med filnavnet *client_secret.json*. Jeg har af sikkerhedsmæssige årsager ikke medtaget den i mit endelige produkt, hvilket betyder, at upload af skemaet til Google kalender ikke er funktionsdygtigt i det afleverede produkt. For at kompensere for dette har jeg, under afsnittet Demonstration og test af produktet, dokumenteret, at det virker.

Udover at hente en client-secret hos Google skal jeg også have installeret selve biblioteket. Dette har jeg gjort ved hjælp af Composer, som er en dependency manager til PHP.

Når Composer er installeret kan jeg i commandopromten (cmd) navigere til min projektmappe, *LectioScrape*, på WampServeren.

```
cd C:\wamp64\www\LectioScrape
```

Og hente Google API-biblioteket til destinationen med følgende kommando:

```
composer require google/apiclient:^2.0
```

Det kan virke besværligt at benytte Composer og kommandoprompten til installation af sådanne biblioteker, men hvis man jævnligt skal installere biblioteker til PHP, kan det være en fordel, at kunne anvende denne metode. Et alternativ er at hente biblioteket på Github [4] og placere det i projektmappen. Google API koden er udgivet under Apache License 2.0 [5], hvilket giver mig lov til at bruge koden næsten som jeg vil.

En sidste ting, der skal sættes op, før kommunikationen med Googles server kan finde sted, er tilføjelsen af et CA-certifikat. Kort forklaret gør dette certifikat serveren i stand til at oprette en sikker forbindelse til Googles server. CA-certifikatet kan hentes på Mozillas hjemmeside [6]. Da forbindelsen oprettes med det PHP-plugin der hedder cURL har jeg været inde på cURLs hjemmeside og hente et certifikat i pem-format, som er det format der kræves [7]. Dette certifikat har jeg tilføjet til følgende sti på min WampServer:

```
C:\wamp64\bin\php\php5.6.25\extras\ssl
```

Herefter har jeg linket til certifikatet i php.ini-filen på WampServeren. Denne initialiseringsfil kan findes på følgende sti:

```
C:\wamp64\bin\apache\apache2.4.23\bin
```

Stien afhænger selvfølgelig af, hvor man har installereret sin WampServer.

På Figur 8 ses et skærmbillede af den linje, jeg har tilføjet til initialiseringsfilen. Jeg har blot linket til CA-certifikatet.

[4] "Google API PHP Client Library".
[5] "Google API PHP Client Library".
[6] Mozilla, "CA certdata".
[7] cURL, "CA certificates extracted from Mozilla".

```
1962
1963  ; Preferred Shared Memory back-end. Leave empty and let the system decide.
1964  ;opcache.preferred_memory_model=
1965
1966  ; Protect the shared memory from unexpected writing during script execution.
1967  ; Useful for internal debugging only.
1968  ;opcache.protect_memory=0
1969
1970  [curl]
1971  ; A default value for the CURLOPT_CAINFO option. This is required to be an
1972  ; absolute path.
1973  curl.cainfo = "C:\wamp64\bin\php\php5.6.25\extras\ssl\cacert.pem"
1974
1975  [openssl]
1976  ; The location of a Certificate Authority (CA) file on the local filesystem
1977  ; to use when verifying the identity of SSL/TLS peers. Most users should
1978  ; not specify a value for this directive as PHP will attempt to use the
1979  ; OS-managed cert stores in its absence. If specified, this value may still
1980  ; be overridden on a per-stream basis via the "cafile" SSL stream context
1981  ; option.
1982  ;openssl.cafile=
```

Figur 8 - Skærmbillede af php.ini og det tilføjede link til CA-certifikatet.

For at opsummere får jeg altså mit program til at sende brugeren over på Googles side, hvor de kan logge ind og give mig adgang til kalenderen. Når de har trykket tillad (eller afvis) bliver brugeren ført tilbage til min server sammen med en kode, der kan bruges til at lave en *token*, som giver mig adgang til deres kalender fremover eller indtil den skal fornys.

Når jeg har genereret denne *token* gemmer jeg den i `$_SESSION`, som er en variabel, der er tilknyttet brugerens session. Sessionen er kort forklaret blot den opretholdte forbindelse mellem browseren og serveren, og den udløber, hvis brugeren ikke interagerer med serveren i et stykke tid.

Nedenstående ses et udsnit af koden, der behandler tilladelsen fra brugeren:

```php
//If the $_GET variable does not contain a code from Google the browser is redirected to a login
page where they can grant access
if (! isset($_GET['code'])) {
    //The authentification URL is created
    $auth_url = $client->createAuthUrl();
    //The browser is redirected
    header('Location: ' . filter_var($auth_url, FILTER_SANITIZE_URL));
}
else {
    //If $_GET['code'] exists, an access token is created and stored in the session
    $client->authenticate($_GET['code']);
    $_SESSION['access_token'] = $client->getAccessToken();

    //The user gets redirected to the uploadToGoogleCal script
    $redirect_uri = 'http://' . $_SERVER['HTTP_HOST'] . '/LectioScrape/uploadToGoogleCal.php';
    header('Location: ' . filter_var($redirect_uri, FILTER_SANITIZE_URL));
```

```
}
```

Fullcalendar

For at lave et pænt kalenderlayout på min side, hvor jeg kan vise Lectioskemaet, gør jeg brug af Fullcalendar, som er udgivet under MIT-licensen, der gør, at jeg må bruge deres software til kommerciel såvel som privat brug[8]. Fullcalendar er en JavaScript-kalender, der indeholder funktioner som dynamisk opdatering af begivenheder, hvilket jeg bl.a. gør brug af. Grunden til, at jeg anvender Fullcalendar, er, at det opfylder mit behov og sparer mig en masse tid og kræfter på layoutet. Den dynamiske del er også bestemt en af grundene til, at jeg benytter Fullcalendar.

En kalender laves på siden ved at indsætte en `<div>` på HTML-siden. I JavaScriptet kaldes Fullcalendarfunktionen, og denne `<div>` kommer til at indeholde kalenderen. Når man definerer kalenderen i JavaScript skal indstillinger sættes op. Her vælger jeg bl.a. at kalenderen skal vise én uge af gangen. Jeg definerer også en eventlistener, som bliver kaldt ved klik på et kalenderevent. På denne måde kan jeg vise ekstra skemainformationer ved klik på skemabrikkerne.

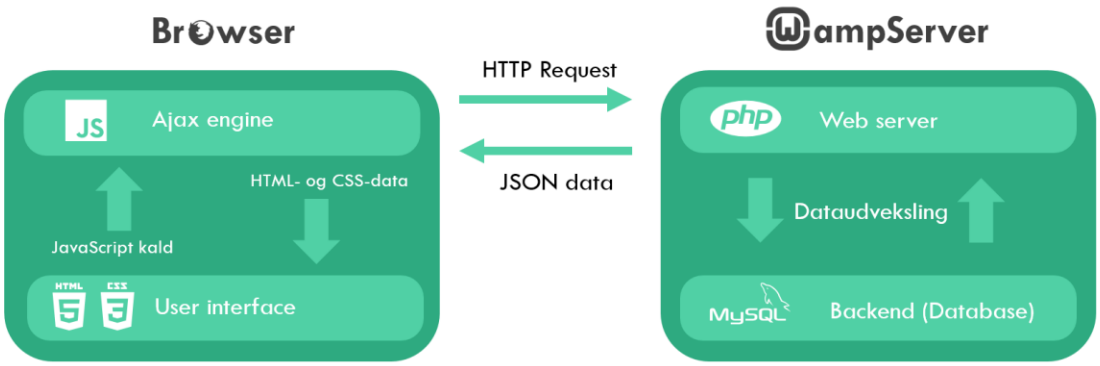
Nedenstående ses min kode, der definerer kalenderen i JavaScript (jQuery).

```
            //When the page is ready the calendar will be initialized
            $('#calendar').fullCalendar({
                //Links to the json feed
                events: 'fullcalendarFeed.php',
                editable: false,
                header: {
                    left: 'prev',
                    center: '',
                    right: 'next'
                },
                //Only view one week at a time
                defaultView: 'agendaWeek',
                minTime: '07:00:00',
                maxTime: '19:00:00',
                themeSystem: 'bootstrap4',
                displayEventTime: true,
                weekNumbers: true,
                timeFormat: 'H(:mm)',
                slotEventOverlap: false,
                eventOverlap: false,
                //Set event click listener. When an event on the calendar is clicked, this will
trigger
                eventClick: function(event, jsEvent, view) {
```

[8] https://github.com/fullcalendar/fullcalendar/blob/master/LICENSE.txt

```
                //Sets modal information
                $('#eventModalTitle').html(event.title);
                $('#eventModalBody').html(event.description);
                $('#eventModalLectioButtonLink').prop("href",event.lessonURL);
                //Opens the modal
                $('#fullCalendarModal').modal();
            },
            firstDay: 1
        })
```

Noget af det vigtigste ved min kalender er, hvordan jeg får skemabrikkerne frem på kalenderen. Fullcalendar er udstyret med en funktion, der gør, at man blot kan linke til et JSON-feed. JSON (JavaScript Object Notation) er blot tekst, der er arrangeret på en bestemt måde, så det følger en standard. Jeg har forbundet kalenderen til `fullcalendarFeed.php`, som er et PHP-script, der serverer en JSON fil indeholdende skemabrikker afhængigt af den uge, man forespørger. Går man videre til næste uge i Fullcalendar, vil JavaScriptet altså forespørge den næste uge, som så vil blive serveret som en JSON-fil. Dette foregår alt sammen med den metode, der hedder AJAX, som er en forkortelse for *Asynkron JavaScript And XML*. AJAX er en måde, hvorpå man kan udveksle informationer med serveren undervejs, som brugeren bruger webapplikationen. Dette princip har jeg vist på Figur 9.



Figur 9 - Illustration af AJAX kald. I dette eksempel bliver dataene leveret fra serveren i JSON-format. Typisk bliver dataene sendt i XML

Brugeren interagerer med brugergrænsefladen. Herefter vil sidens JavaScript sende en http-forespørgsel til webserveren, som så henter data fra databasen, og sender det til browseren. Typisk vil dette være i XML-format, men da jeg benytter JSON har jeg illustreret dette på figuren. JavaScriptet vil

så sørge for, at HTML/CSS-brugergrænsefladen bliver opdateret i henhold til den hentede data. Alt dette foregår uden, at siden genindlæses.

Moment.js

Moment.js er en tilføjelse, der gør det nemmere at manipulere tidspunkter og datoer i JavaScipt. Årsagen, til at jeg anvender denne tilføjelse, er dels, at Fullcalendar ikke fungerer uden, og dels, at jeg på klientsiden nemt kan ændre på datoformatet, så jeg ikke skal gøre det på serversiden. Jeg sikrer altså, at serveren modtager tidspunkter i det rigtige format. Moment.js downloades til projektmappen og inkluderes på samme måde som jQuery og Bootstrap.

```
    <script src='lib/moment.min.js'></script>
```

Moment.js er udgivet under MIT-licensen ligesom Fullcalendar og Bootstrap.

Font Awesome

Font Awesome er en pakke af ikoner, der kan anvendes på websites ved blot at linke til et stylesheet. Jeg har inkluderet deres ikoner ved at linke til deres CDN (Content Delivery Network), da dette er nemmere end at hente alle deres ikoner.

```
    <link href="https://use.fontawesome.com/releases/v5.0.8/css/all.css" rel="stylesheet">
```

Ikonerner kan så bruges ved at anvende HTML's `<i>`-tags og så og give dem en class, der definerer, hvilket ikon det er. Nedenstående ses et eksempel på et Font Awesome ikon i HTML.

```
  <i class="fas fa-user"></i>
```

Klassen *fas fa-user* vil give følgende ikon:

Jeg benytter ikke Font Awesomes ikoner med `<i>`-tags, men til gengæld har jeg sat Fullcalendar op til at benytte ikonerne, så jeg får et mere moderne design.

Font Awesome Free er open source. Koden er udgivet under MIT-licensen mens ikonerneskrifttyperne er udgivet under *SIL Open Font License* [9]. Dette gør, at jeg må bruge dem til både private og kommercielle formål.

Github

Jeg anvender Github til at gemme min kode samt dokumentere mine fremskridt i udviklingen af produktet. Github er en hosting service, der gør brug af Git. Git er et såkaldt *version control* system, som holder øje med ændringer i filsystemet og registrerer disse ændringer [10]. Koden til mit produkt kan findes på:

```
https://github.com/AndreasLF/LectioScrape
```

Det skal nævnes, at min Google API-kode ikke er inkluderet i Github af sikkerhedsmæssige årsager.

Under kodningen af produktet har jeg ført logbog, som også kan findes i Github. Under hvert *commit* (ændring jeg har uploadet til systemet) har jeg skrevet, hvilke ændringer jeg har lavet siden sidst. Jeg har arbejdet ud fra tre begreber, som jeg kort vil forklare.

CHANGE

Først laver jeg ændringer i min kode. Det kan fx være kode, der tilføjer en ny feature til programmet. Denne ændring betegner jeg selvfølgelig med *CHANGE*.

POC

POC står for *Proof Of Concept* og handler om, at man skal kunne teste sin kode. Jeg har så vidt muligt forsøgt kun at lave *CHANGE's*, som kan testes, og jeg har de fleste steder beskrevet, hvordan mine tests er udført. Ofte består testen blot af at udskrive værdier til skærmen.

REFAC

REFAC står for refaktorering og er kort forklaret tilpasning af koden, så den giver mere mening eller bliver mere effektiv. Man kan kalde det for en form for oprydning af koden.

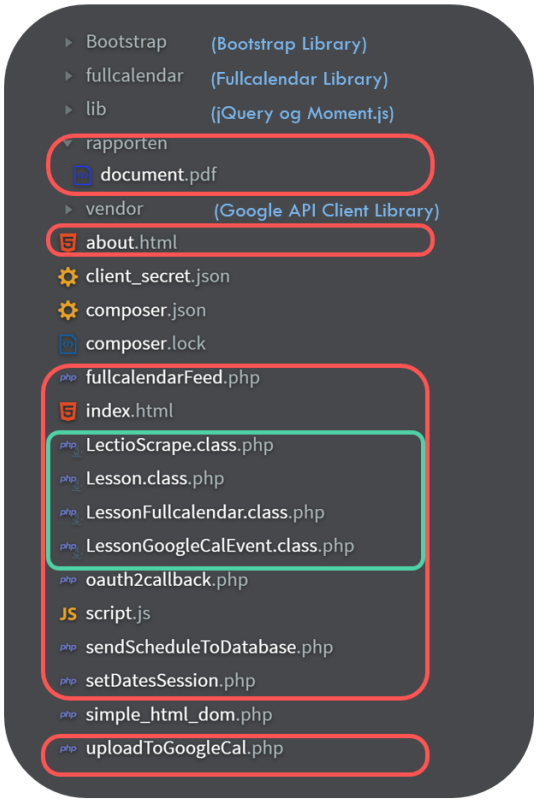
[9] Font Awesome, "License".
[10] Chacon og Straub, *Pro Git*, 8.

Kodeopbygning

Jeg har som sagt forsøgt at følge princippet med CHANGE, POC og REFAC under kodningen af mit program. Koden har udviklet sig undervejs gennem produktudviklingen, og jeg vil ikke umiddelbart komme nærmere ind på processen undervejs. Denne er dokumenteret i Github.

I dette afsnit vil jeg fokusere på, hvordan mit endelige produkt er struktureret.

På Figur 10 ses filerne i min projektmappe. De filer, jeg selv har programmeret, er markeret, så der ikke er tvivl om, hvilke dokumenter, der er biblioteker, og hvilke jeg selv har skrevet.

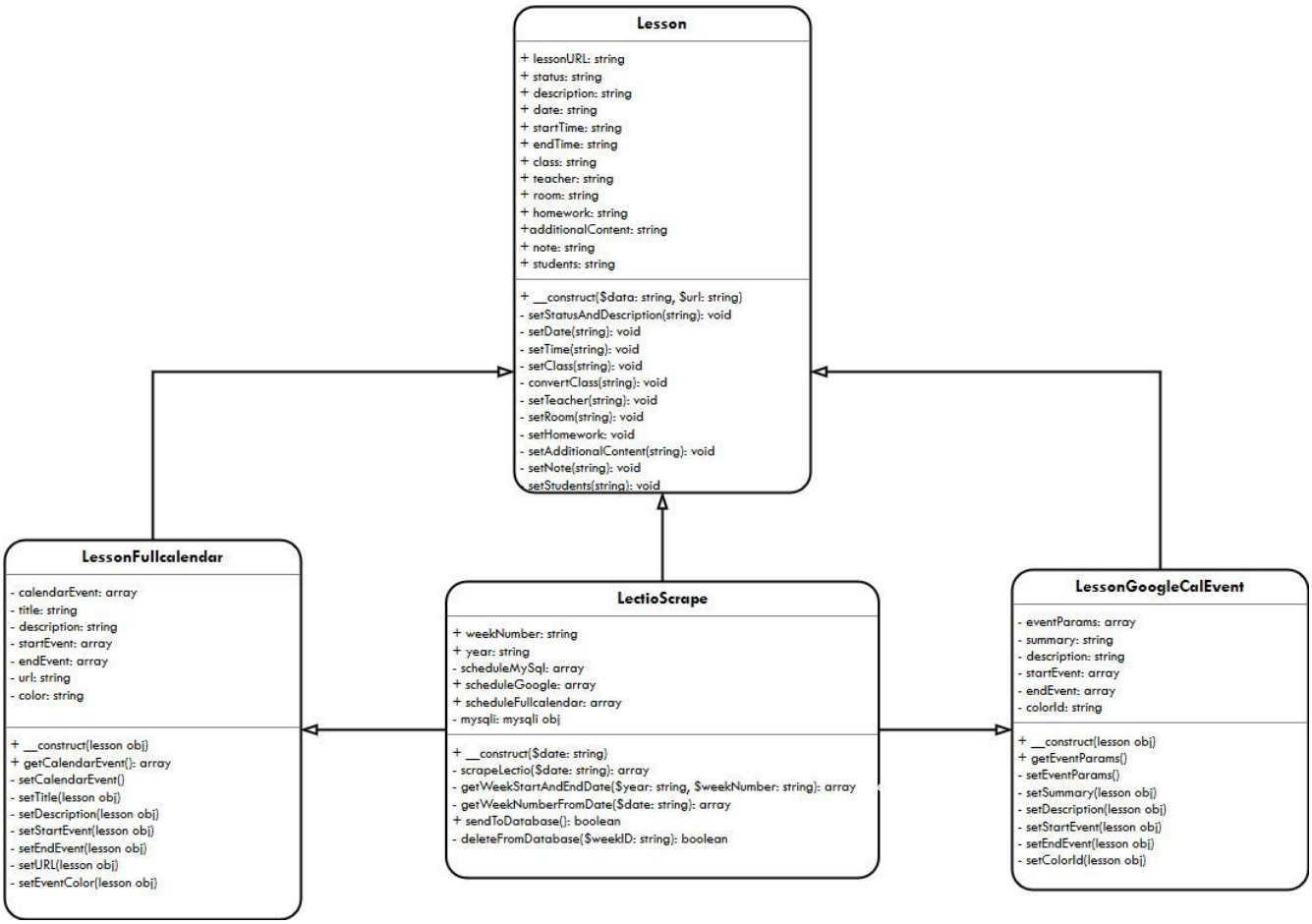


*Figur 10 - Liste over projektmappen på WampServeren. Filerne jeg selv har lavet er markeret med **rød**. Mappernes indhold er kort beskrevet med **blå**. Klasser er markeret **grøn**.*

Klasser

Jeg har forsøgt så vidt muligt at følge *DRY*-princippet, som står for *Don't Repeat Yourself* og handler om at skrive koden på en måde, så der ikke er noget af koden, der bliver gentaget. Dette har jeg forsøgt at opnå ved at lave en række klasser, der både gør det nemmere at videreprogrammere og forstå programmet. Hver klasse har jeg gemt i et PHP-dokument for sig selv, og jeg har tilføjet *.class* til filnavnet. På Figur 10 har jeg markeret klasserne med **grøn**.

Jeg har på Figur 11 opstillet et klassediagram, hvor man kan se klassens navn, properties og metoder. På figuren kan man også se deres afhængighed af hinanden.



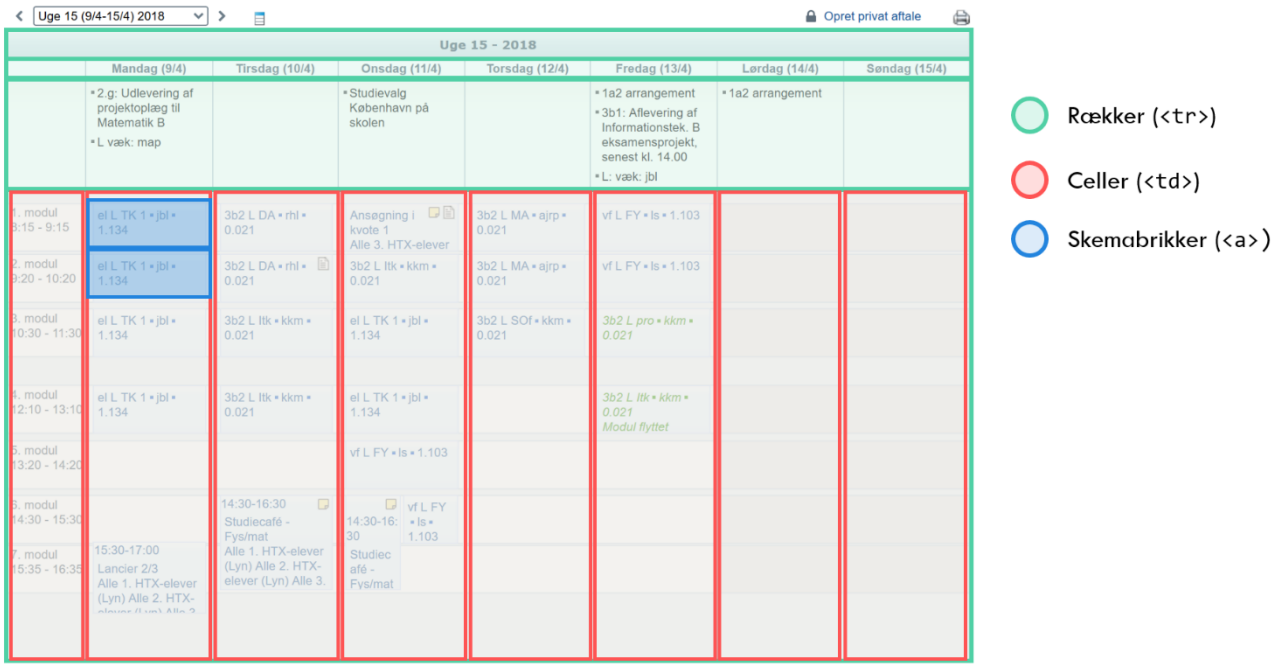
Figur 11 - Klassediagram over programmet (kun de klasser jeg selv har programmeret). LectioScrape er afhængig af Lesson, LessonFullcalendar og LessonGoogleCalEvent, hvilket er vist med pilene. LessonGoogleCalEvent og LessonFullcalendar er afhængige af Lesson

Lesson

Lesson-klassen definerer en skemabrik, som indeholder informationer om tidspunkt, fag lærer osv. Lesson-klassens construct-metode (metoden der konstruerer objektet) tager to inputparametre - data og url. De to inputparametre er dem, som jeg scraper fra hjemmesiden.

Ved at kigge grundigt på HTML-strukturen for Lectios skemaside er jeg kommet frem til hvordan skemaet skal scrapes fra hjemmesiden. HTML-koden kan ses ved at anvende browserens *web inspector* - i Firefox kan *inspectoren* findes under udviklerværktøjer.

Jeg er kommet frem til, at siden består af en tabel, hvor hver dag i skemaet befinder sig i en af tabellens celler. Jeg har forsøgt at illustrere dette på Figur 12. Her har jeg markeret tabellens rækker og tabellens celler. I hver celle befinder der sig et `<div>` -tag, hvori skemabrikkerne er repræsenteret som link-tags (`<a>`).



Figur 12 - Figur over Lectioskemaets HTML-struktur

Hver af disse `<a>`-tags har en `href`-attribut, som indeholder en URL til skemabrikkens side. Samtidig har hver skemabrik også en `data`-attribut. I HTML bliver `data`-attributten brugt til at indlejre

brugerdefineret data i HTML-koden [11]. På Lectios skemaside hedder den `data`-attribut, de anvender, `data-additionalinfo`. Denne attribut indeholder alle de informationer om skemabrikken, som jeg skal bruge.

Nedenstående ses et eksempel på en skemabrik fra Lectio:

```
    <a
href="/lectio/681/aktivitet/aktivitetforside2.aspx?absid=26447729185&amp;prevurl=SkemaNy.aspx%3fty
pe%3delev%26elevid%3d14742506655&amp;elevid=14742506655" class="s2skemabrik s2bgbox s2withlink
lec-context-menu-instance" style="position:absolute; top:9.55em;
left:0.55em;height:3.82em;width:9.09em;word-wrap:break-word;" data-additionalinfo="10/4-2018 10:30
til 11:30 Hold: 3b2 L Itk Lærer: Kristian Krabbe Møller (kkm) Lokale: 0.021">

        <div class="s2skemabrikInnerContainer">

            <div class="s2skemabrikcontent">

                <span data-lectiocontextcard="HE16681476804">3b2 L Itk</span><img alt=""
src="/lectio/img/transparent.png" class="s2seperator"><span data-
lectiocontextcard="T14748965700">kkm</span><img alt="" src="/lectio/img/transparent.png"
class="s2seperator">0.021</div>

        </div>

    </a>
```

Lesson-klassen tager skemabrikkens url som input, som kan findes i `href` -attributten, samt skemadataene, som findes i `data-additionalinfo`.

Dataene, der findes i `data-additionalinfo` er en relativt rodet tekststreng. Dataene står i samme rækkefølge og på samme måde på hver skemabrik, hvis de eksisterer, men for at jeg kan præsentere skemaet på en pæn måde samt uploade det til en database og Googles kalendarsystem, bliver jeg nødt til at organisere indholdet af strengen. Til dette bruger jeg *regular expressions* eller *regex*. *Regex* bruges til at søge efter mønstre i tekststrenge. Jeg skriver *regex'es* med syntaksen fra Perl 5, som jeg beskrev tidligere.

Jeg har observeret, at datoen altid optræder i formatet `dd/mm/yyyy`. Årstallet indeholder altid 4 cifre, dagen kan indeholde 2 eller 1 ciffer, og det samme kan måneden. For at søge efter dette mønster ved

[11] W3Schools, "HTML Global data-* Attributes".

hjælp af regex i PHP anvender jeg `preg_match()`, som er en funktion, der udfører *regular expressions* i PHP. Nedenstående ses mit *regex* til at finde datoen for skemabrikken.

```
preg_match('/(\d\d|\d)\/(\d\d|\d)-(\d\d\d\d)/', $data, $matches)
```

Jeg har på samme måde undersøgt mønstre for alle de andre skemainformationer og udarbejdet regular expressions undervejs. Jeg har benyttet værktøjet på https://regex101.com/, som kan bruges til nemt og overskueligt at teste regular expressions uden at man skal eksekvere et testdokument på WampServeren.

Det skal nævnes, at det ofte anses for dårlig praksis at skrive et program, der udelukkende er afhængigt af regular expressions [12]. Jeg har dog valgt at anvende det alligevel, da jeg ser det som den nemmeste løsning at implementere. Jeg gør også brug af PHP Simple HTML DOM Parser, så jeg undgår at skulle udføre *regular expressions* på hele HTML-siden.

LessonGoogleCalEvent

LessonGoogleCalEvent definerer ligesom Lesson-klassen en skemabrik. Forskellen mellem de to er, at LessonGoogleCalEvent konsturerer en skemabrik, som er klar til at blive uploadet til Google kalender. Klassen er afhængig af Lesson, da den kun kan konstrueres, hvis den modtager et Lesson-objekt som inputparameter. Denne afhængighed har jeg illustreret med en pil på Figur 11.

Denne klasse tager så at sige bare et Lesson-objekt og formaterer det, så det har det rigtige format til Google kalender.

LessonFullcalendar

LessonFullcalendar virker på samme måde som LessonGoogleCalEvent. Her konstrueres blot en skemabrik i det format, som Fullcalendar skal bruge. LessonFullcalendar er også afhængig af Lesson-klassen, da denne også skal bruge et Lesson-objekt.

LectioScrape

LectioScrape-klassen er den, man bruger, når man skal scrape skemaet. Når man konstruerer et objekt fra denne klasse giver man den en datostreng som inputparameter. Denne datostreng vil blive konverteret til et ugenummer, og skemaet for denne uge vil blive scrapet. Et Lesson-objekt vil blive

[12] Scrapers, *Webbot Spider and Screen Scrapers*, 47; Turland, *php|architect's Guide to Web Scraping with PHP*, 143.

oprettet med den scrapede data, og med dette Lesson-objekt vil der blive oprettet et LessonGoogleCalEvent-objekt og et LessonFullcalendar-objekt. Dette gøres for hver skemabrik på den hentede uge, og disse skemabrik-objekter gemmes i arrays. Da LectioScrape opretter både Lesson-, LessonFullcalendar- og LessonGoogleCalEvent-objekter er denne altså afhængig af de tre andre klasser. Jeg har ikke lavet nogen form for nedarvning fra mine klasser, men afhængighed er der stadig imellem dem.

Databasen

LectioScrape indeholder også en metode, der gør, at man kan uploade skemaet til MySql-databasen. For at uploade til databasen gør jeg brug af MySql Improved Extension, som er en del af PHP[13]. Jeg opretter et såkaldt mysqli-objekt, som indeholder informationer om databasen - adresse, brugernavn, kodeord og databasenavnet.

```
$mysqli = new mysqli("localhost","LectioDB","password","lectio");
```

Når mysqli-objektet er oprettet forbereder jeg min kommando til databasen med `prepare`-metoden, som er en del mysqli-klassen.

```
$stmt = $mysqli->prepare("INSERT INTO skema(ID,Week, Status, Description, Date, StartTime,
EndTime, Class, Teacher, Room, Homework, Note) VALUES (NULL,?,?,?,?,?,?,?,?,?,?,?)");
```

Spørgsmålstegnene i `prepare` -metoden repræsenterer alle variablene. Her kan man så bruge `bind_param` til at binde værdier til disse variable. Denne metode anvender man for at undgå SQL-injections, som jeg har skrevet mere om i afsnittet Sikkerhed.

```
$stmt->bind_param("sssssssssss",$weekID,$lesson->status,$lesson->description,$lesson-
>date,$lesson->startTime,$lesson->endTime,$lesson->class,$lesson->teacher,$lesson->room,$lesson-
>homework,$lesson->note);
```

[13] PHP, “MySQL Improved Extension”.

Strengen bestående af s'er fortæller, hvilken type hver af variablerne, som jeg binder til min forberedte kommando, har. I mit tilfælde er de alle strings.

Til sidst kan den forberedte kommando eksekveres.

```
$stmt->execute();
```

Før jeg uploader skemadataene til databasen. Sletter jeg allerede eksisterende data for den uge, jeg uploader. Dette gøres på samme måde med forberedte kommandoer.

```
$mysqli->prepare("DELETE FROM `skema` WHERE `Week`= ?");
```

Hvis jeg skulle bruge databasen til noget, ville det ikke give meget mening at slette alle dataene før upload af nye data. En af årsagerne til, at jeg har anvendt en MySql-database er dels at vise, at jeg besidder viden om databaser og dels at åbne op for muligheden for videreudvikling. Noget, man kunne bruge databasen til, er at undersøge, om der er sket ændringer på Lectio siden sidste tjek. Dette ville kræve et system lidt i stil med Git, som tjekker ændringer.

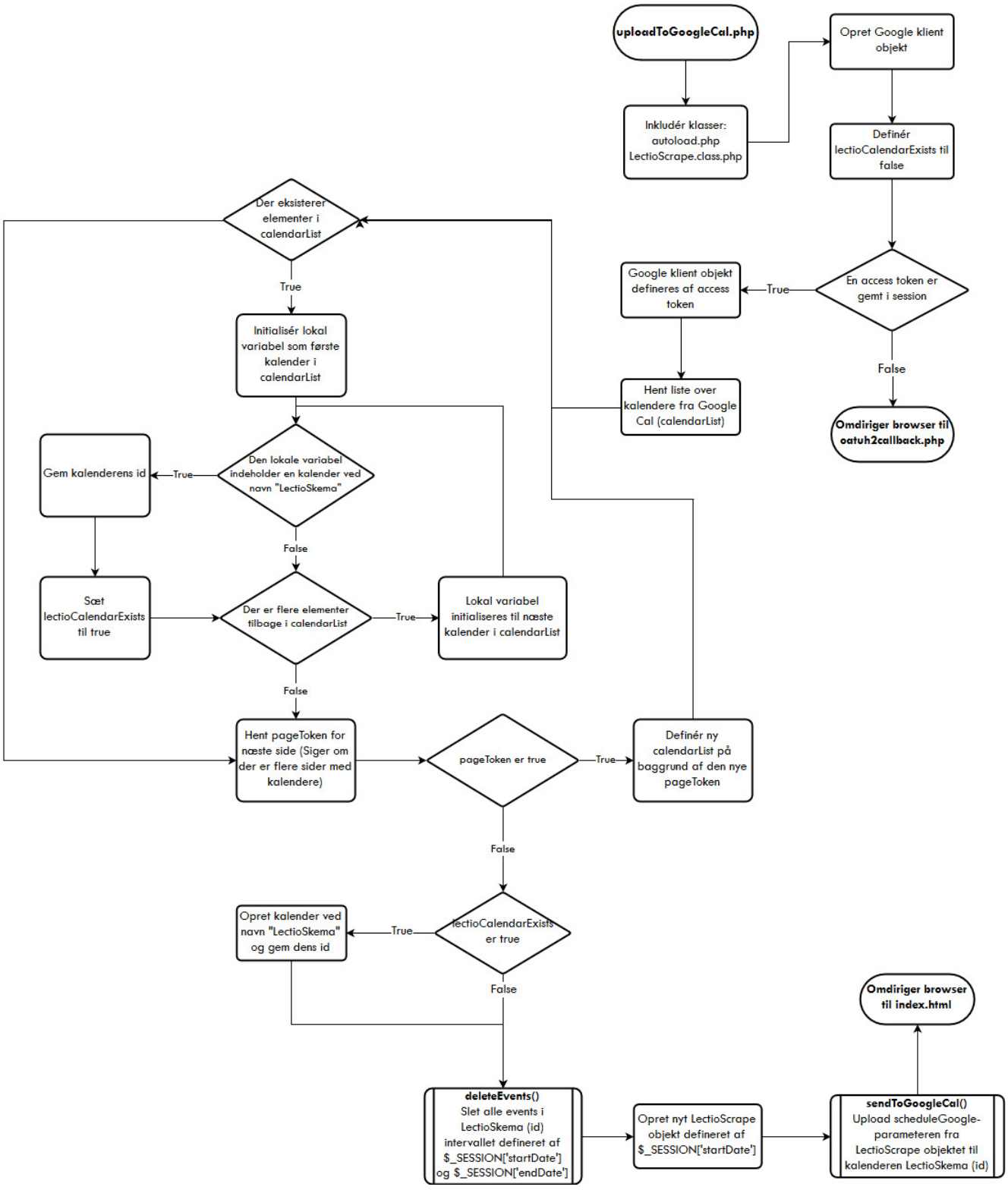
Koden til oprettelse af tabeller i databasen ses i Bilag. Databasen er oprettet ved hjælp af phpMyAdmin, som er et værktøj, der følger med WampServer. Jeg har oprettet en ny bruger ved navn *LectioDB*, som kun har adgang til *lectio*-databasen. Dette har jeg gjort for at sikre, at jeg ikke kan ødelægge noget uden for denne database.

Flowchart

Ud over klasser består mit program også af dokumenter med anden kode. Jeg vil ikke redegøre for det hele, men fokusere på dele af det. Jeg vil især fokusere på `uploadToGoogleCal.php`. Dette script sørger for at uploade skemaet til Google kalender. Jeg har på Figur 13 oprettet et flowchart over koden i `uploadToGoogleCal.php`.

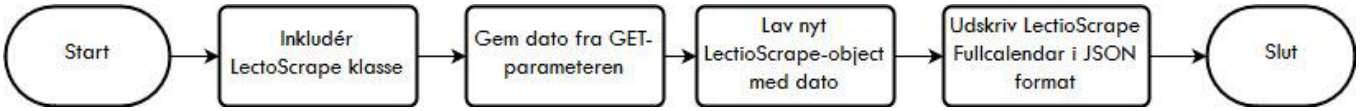
Først inkluderes klasser fra andre dokumenter og et Google Client-objekt defineres. Hvis en *access token* er gemt i sesionen, vil Client-objektet defineres af denne, hvis ikke vil brugeren blive sendt over

til oauth2callback.php, så en *access token* kan hentes hos Google. Såfremt en *access token* er gemt, og Client-objektet er opdateret, vil en liste over kalendere fra Google Calendar hentes. For hver kalender i listen tjekkes om kalenderens navn er lig ”LectioSkema”. Hvis der ikke eksisterer en kalender med dette navn, vil den oprettes. Herefter vil alle events i den valgte uge, som er defineret af `$_SESSION['startDate']` og `$_SESSION['endDate']`, blive slettet ved hjælp af en funktion, jeg ikke vil komme nærmere ind på. Når ugen er slettet fra kalenderen, vil den opdaterede uge uploades til kalenderen. Det er ikke det mest effektive at slette hele ugen for så at oprette en ny, når der formentlig kun har været få ændringer siden sidst skemaet blev uploadet. Her ville en form for versionshistorik udarbejdet på baggrund af databasen være optimal.



Figur 13 - Flowchart over uploadToGoogleCal.php

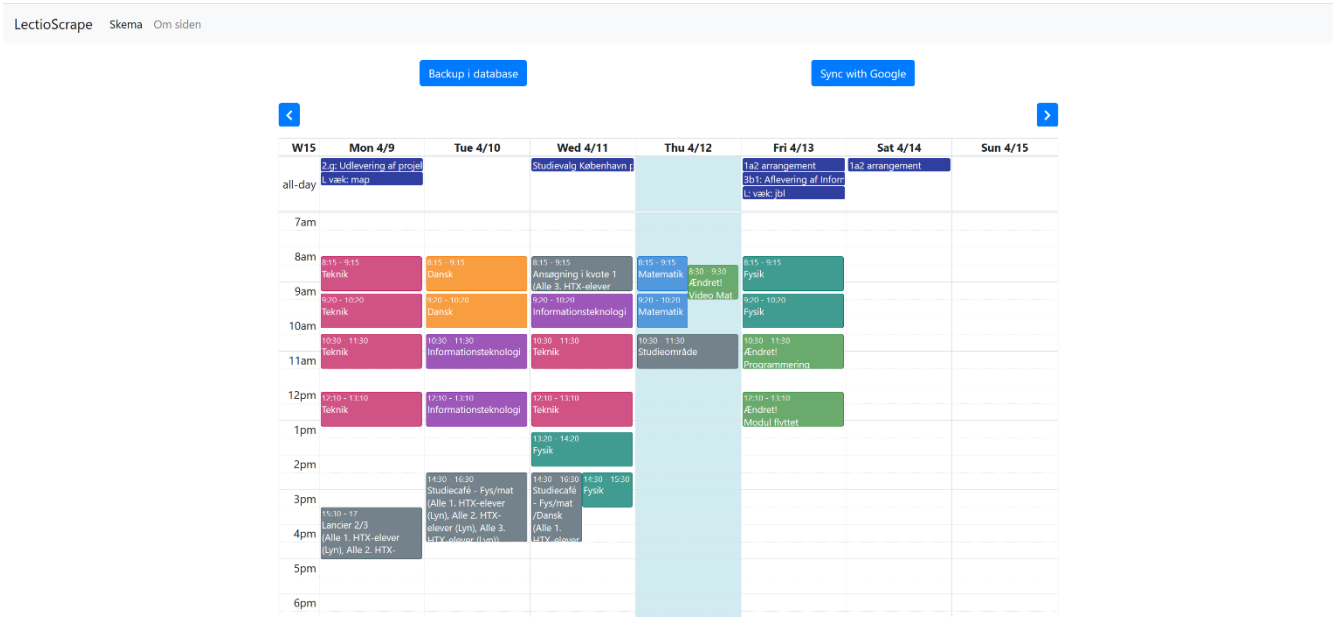
For at illustrere effektiviteten af mine klasser, har jeg også udarbejdet et flowchart over koden fra fullcalendarFeed.php. Flochartet ses på Figur 14, og der er kun meget få blokke, da LectioScrape gør det meste af arbejdet.



Figur 14 - Flowchart over fullcalendarFeed.php

Layout

Layoutet af brugergrænsefladen er helt central, når man udarbejder IT-løsninger til folk uden teknisk baggrund. Jeg har fokuseret på at lave et så minimalistisk design som muligt, så der ikke er nogen forstyrrende elementer på siden, der kan afskrække brugeren fra at bruge applikationen. På Figur 15 ses et skærmbillede af forsiden på min applikation.



Figur 15 - Skærmbillede af applikationen

Jeg har, som nævnt tidligere, benyttet Fullcalendar til at lave et pænt kalender-layout, der samtidig opdaterer automatisk, når jeg går videre til næste uge. For at gøre skemaet mere intuitivt har jeg forsøgt at farvekode alle timerne. Jeg har benytte farverne fra Googles Material Design [14], da farverne er meget fyldige og formentlig frembringer en følelse af genkendelse hos brugeren. De har formentlig set farverne før i andet software. Jeg har brugt 700-farverne på trods af at Google anbefaler at bruge 500 som primær farver. Dette har jeg gjort fordi de mørkere farver gør, at det ikke er så overvældende, når jeg bruger så mange forskellige farver. Google anbefaler heller ikke ligefrem at blande alle farverne fra deres pallette, men da jeg gerne vil have kontrast imellem skemabrikkerne har jeg gjort det alligevel.

Navigationsbjælken er grå, så den ikke står så meget i kontrast til baggrunden - jeg vil gerne have, at det er skemaet, der er i fokus.

Knapperne er blå, som man typisk ser ved klikbare links. Der skal ikke være tvivl om, at man kan klikke på knapperne.

Sikkerhed

Når man bygger et stykke software er det vigtigt at tage højde for sikkerheden. Jeg har nedenstående beskrevet nogle sikkerhedsrisici ved mit program.

DDoS

Først vil jeg pointere, at web-scraping kommer med en række risici, hvis det ikke bliver gjort orentligt. Når man arbejder med web-scraping og programmerer en bot, der er i stand til at hente data fra en side, vil den principielt også være i stand til at sende så mange forespørgsler, at systemadministratorerne kan se det som et forsøg på at hacke deres system [15]. At lægge et system ned ved at sende så mange forespørgsler, at serveren ikke kan følge med, kaldes for DDoS (Distributed Denial of Service). DDoS straffes med bøde eller fængsel ifølge straffelovens §293, stk 2 [16]. De fleste servere vil blot blokere min IP-adresse, hvis jeg sender for mange forespørgsler, men jeg kan risikere at komme ud i juridiske problemer, og jeg bliver derfor nødt til at være forsigtig, når jeg scraper data.

[14] Google, “Material Design - Color”.
[15] Scrapers, *Webbot Spider and Screen Scrapers*, 3.
[16] “Hacking og DDoS”.

SQL-injections

Når man arbejder med databaser, skal man tage højde for SQL-injections. SQL-injections er tilføjelse af potentielt *farlig* kode til databasesystemet. Kode kan tilføjes til databasen gennem inputfelter på en hjemmeside. Har man et inputfelt på sin side, hvor man indtaster brugernavn, er det sandsynligt at serveren sender følgende kode til databasen.

```
SELECT * FROM Users WHERE Username = brugernavn;
```

Databasen vil her *spytte* alle rækker ud, hvor brugernavnet er lig det indtastede brugernavn. En ondsindet person kunne finde på at skrive følgende ind i indtastningsfeltet:

```
brugernavn OR 1=1;
```

Nu vil serveren lige pludselig sende følgende linje til databasen:

```
SELECT * FROM Users WHERE Username = brugernavn OR 1=1;
```

Og da 1 altid er lig 1, vil databasen returnere alle rækker fra databasen, og man vil på denne måde kunne få fat i både brugernavn og passwords, som man ikke skulle have adgang til.

Jeg har ikke nogen inputfelter på min side, og risikoen for SQL-injections er altså ikke noget, jeg bør bekymre mig om. En af mine idéer til udbygning er, at man kan anvende skemaet, som er gemt i databasen til noget nyttigt. Her vil et loginsystem nok være fordelagtigt, og derfor er SQL-injections stadig relevant. Man kan sikre sig mod injections ved at anvende såkaldte prepared statements, som forbereder en kommando og først indsætter variable bagefter. Variablens type defineres, når den indsættes i den forberedte kommando, og hvis den fx er en string, vil den blive betragtet som en string og ikke MySQL-kode.

Google API

Der er også risiko for misbrug af Googles API. Systemet burde som sådan være sikkert, da oauth-nøglerne ikke vil blive tilladt udveksling over en ikke-krypteret forbindelse. Dette erfarede, da jeg testede programmet før tilføjelse af CA-certifikaterne. Den største risiko ved systemet er nok misbrug af min API-nøgle. Det skal siges, at jeg i API-klienten har linket til min server som callback, hvilket kan mindske misbrug. I mit tilfælde fører callback-linket til `localhost`, hvilket principielt også kan

åbnes på andre computere, og dermed kan nøglen godt misbruges. Normalt ville callback føre tilbage til en server, der har et unikt domæne, som kun jeg har adgang til. Ifølge Google er det god praksis ikke at give API-nøglen til andre eller inkludere den i koden, hvor den kan ses. De anbefaler også, at man opdaterer den med jævne mellemrum [17].

Hvad er tilladt?

Når man arbejder med web *scraping* er det relevant at stille spørgsmålstegn ved, hvem der ejer hvilken data, og hvad der er i orden at *scrape*. Lectios skemaer er offentligt tilgængelige og principielt kan jeg manuelt gå ind og hente alle de data, som jeg henter med mit program. Jeg ser ikke noget moralsk forkert i at *scrape* skemaet i skolesammenhæng, men jeg vil gerne understrege, at jeg er klar over, at der kan være juridiske begrænsninger i forhold til, hvad der er tilladt. Selvom databaser er offentligt tilgængelige, kan de godt være beskyttet af ophavsret, hvilket jeg kan risikere at krænke, hvis jeg *scraper* databasen [18]. Der er ikke noget, der tyder på, at vilkårene for Lectios offentlige database er åbne, og derfor kan Lectio principielt anklage mig for brud på ophavsretten, ifølge It-advokaten Martin Haller Grønbæk, som har udtalt sig til Version2 i forbindelse med en sag omhandlende scraping af Tingbogen [19]. Det skal nævnes, at en anklage ikke nødvendigvis betyder, at man juridisk set har krænket nogens ophavsret.

Så vidt jeg kan komme frem til, er der ingen klausuler på Lectios hjemmeside, der begrænser min brug af deres side. Og hvem har egentlig ophavsretten? Lectio ejer databasen, men ejer skolen ikke selve dataene? Dette spørgsmål er ikke til at svare på uden kendskab til de aftaler, skolen har indgået med platformen. Jeg har ikke indgået nogen aftale ved brug af Lectio, så jeg gør i hvert fald ikke noget, jeg har skrevet under på, at jeg ikke vil gøre.

Uden at have nogen juridiske forudsætninger for at tage stilling til spørgsmålet om, hvorvidt jeg må *scrape* data fra Lectios side, vil jeg mene, at undersøgelse og afprøvning web scraping til

[17] Google, “Best practices for securely using API keys”.
[18] Lundström, “It-advokat om Tingbogen-kopi: Det kan give bøder at scrape offentlige data”.
[19] Lundström.

uddannelsesformål, er i orden. Mit program kan kun *scrape* én uge af gangen, hvilket betyder, at jeg ikke belaster Lectios servere synderligt, da jeg kun tilgår siden få gange.

Demonstration og test af produktet

Som nævnt har jeg undervejs testet min kode. Jeg har fokuseret på at lave små ændringer, der kan testes. Jeg har arbejdet meget med Arrays og Lister i PHP, og for at teste indholdet af disse har jeg gjort brug af `var_dump()`, som kan bruges til at printe indholdet samt strukturen af arrays i PHP.

Den endelige test af produktet har blot gået ud på at scrolle igennem ugerne på mit skema for at se, om skemaet bliver hentet, som det skal. Jeg har forsøgt at uploade skemaet til Google kalender - både til tomme uger og til uger, der allerede indeholder et skema. Test af databaseupload har foregået på samme måde.

Generelt virker mit produkt som det skal. Det skal dog nævnes, at der af og til dukker nye strukturer og skemainformationer op, som jeg ikke har taget højde for. Jeg har forsøgt at tage højde for så meget som muligt, men uden indblik i Lectios fulde databasestruktur er det svært at lave en perfekt tilpasning. Her kræves der løbende tilpasning, som tingene dukker op. Af denne grund har jeg også lavet klassestrukturen, som jeg forklarede tidligere. Det er nemmere at videreudvikle på.

Mit program er meget afhængig af både Lectios system og Google Calendars API. Googles API vil formentlig forblive tilgængelig. Der er dog risiko for at systemet ændrer sig eller at adgangen bliver begrænset efter Facebook-skandalen, hvor udviklere har misbrugt adgangen og høstet data for 87 mio. brugere [20].

Der er også risiko for at Lectio ændrer deres system. Der træder bl.a. nye regler i kraft angående behandling af EU-borgeres data til maj måned 2018 [21]. Lectio har tidligere fået kritik af deres offentlige

[20] Cadwalladr og Graham-Harrison, "Revealed: 50 million Facebook profiles harvested for Cambridge Analytica in major data breach"; Badshah, "Facebook to contact 87 million users affected by data breach".
[21] Burgess, "What is GDPR? The need-to-know guide".

skemaer [22], og man kunne forestille sig, at de på et tidspunkt bliver påbudt at gemme skemadataene bag login. Hvis de ændrer deres system, vil det resultere i, at mit program stopper med at virke.

Da mit produkt er så afhængigt af andre tjenester har jeg optaget en video, som demonstrerer, at produktet virker efter hensigten i skrivende stund. Videoen kan findes i det vedlagte zip-dokument og kan også findes på:

```
https://1drv.ms/v/s!As2AgsiGzUmog4ZdPwUJQP1z1XXb9w
```

Vurdering og perspektivering

Det er lykkedes mig at scrape Lectios offentlige skema samt uploade det til en database og Googles kalendersystem. Produktet virker efter hensigten, og jeg har med mit program demonstrereret web-scrapings potentiale. Der findes utallige mængder af data på internettet, som ikke nødvendigvis er organiseret. Ved hjælp af web-scraping kan denne data få *nyt liv* og ny funktionalitet kan tilføjes. Mange selskaber tilbyder API-adgang til deres databser, og det er ikke altid klart, om et program scraper data eller har fået adgang til en database gennem en API. Som sagt kan scraping befinde sig i en gråzone mht. hvad der er lovligt, da ejerne for det meste ikke har givet lov til at scrape deres side. På trods af det juridiske spørgsmålstegn man kan stille ved web-scraping, kan man ikke komme uden om, at det er en metode, der kan gavne mange mennesker. Det kan også misbruges og gøre meget skade. Ufører man et par hurtige søgninger på *web scraping*, *data aggregation* og *big data*, vil man hurtigt finde ud af, at indsamling af data er en kæmpe industri, der kun fortsætter med at vokse.

Forslag til forbedring og videreudvikling

På trods af, at kravene til produktet er opfyldt, er der mange andre funktioner, man kunne tilføje til produktet. Jeg har nedenstående oplistet funktioner, som man kunne udbygge programmet med:

- Mulighed for at vælge skema selv. Mit program er sat op til at scrape mit skema. Man kunne tilføje en funktion, så man selv kan indtaste sit Elev-id og får scrapet sit skema.

[22] Svansø, "Oplysninger om danske skoleelever flyder frit på populær lektieplatform".

- Muligheden for at uploade mere end én uge af gangen til Google kalender. Denne funktion ville være relativt simpel at tilføje, men ville øge loadingtiden betydeligt. En mere effektiv *scraping*-metode vil derfor muligvis være nødvendig.
- Tilføjelse af feedback ved loading af skema og upload til database og Google kalender. Der mangler en funktion, som fortæller brugeren at siden arbejder, så man ikke tror, at den er frosset.
- Mulighed for selv at sammensætte skemalayoutet: Brugeren skal have mulighed for selv at vælge hvilke farver der skal tilknyttes timerne på skemaet.
- Databasen kunne anvendes til at føre en opdateringshistorik, samt backup til når Lectio er nede. Opdateringshistorikken kan bruges til at se, hvornår der er sket ændringer på skemaet, og om der er aflysninger.
- Mulighed for e-mail eller sms-notifikation ved aflyste timer. Her skal man gøre brug af opdateringshistorikken og en tjeneste, der kan sende sms'er til brugeren. Her kunne man bl.a. bruge den betalte service Twilio.com, som har PHP-understøttelse [23].

Dette er blot idéer til videreudvikling. Der findes utallige muligheder for at anvende skemadataene. Man kan også vælge at scrape mere data fra Lectio som opgaver, beskeder osv.

[23] Twilio, "Twilio".

Kilder

Værktøjer

Nedenstående har jeg samlet en liste over de værktøjer, jeg har benyttet til udviklingen af mit produkt samt udarbejdelse af rapporten. Jeg har linket til værktøjerne.

Rapporten

- MS Office: Word og PowerPoint (https://www.office.com/)
- Draw.io (https://www.draw.io/)
- Devicons 2.0 (https://github.com/konpa/devicon/ og https://konpa.github.io/devicon/)
- Mendeley (https://www.mendeley.com/)

Produktet

- Brackets (http://brackets.io/)
- GitHub (https://github.com/)
- GitHubs desktopklient (https://desktop.github.com/)
- WampServer (http://www.wampserver.com/en/)
- jQuery (https://jquery.com/)
- Bootstrap (https://getbootstrap.com/)
- Fullcalendar (https://fullcalendar.io/)
- Moment.js (https://momentjs.com/)
- Fontawesome (https://fontawesome.com/)
- PHP Simple HTML DOM Parser (http://simplehtmldom.sourceforge.net/)
- Google API Client Library (https://github.com/google/google-api-php-client og https://developers.google.com/api-client-library/php/)
- Composer (https://getcomposer.org/)
- regular expressions 101 (https://regex101.com/)

Litteraturliste

Nedenstående ses min litteraturliste:

Badshah, Nadeem. “Facebook to contact 87 million users affected by data breach”. *The Guardian*, 8.

april 2018. https://www.theguardian.com/technology/2018/apr/08/facebook-to-contact-the-87-million-users-affected-by-data-breach.

Bootstrap. “License FAQs”. Bootstrap. Set 13. april 2018. https://getbootstrap.com/docs/4.0/about/license/.

Burgess, Matt. “What is GDPR? The need-to-know guide”. WIRED UK. Set 13. april 2018. https://www.wired.co.uk/article/what-is-gdpr-uk-eu-legislation-compliance-summary-fines-2018.

Cadwalladr, Carole, og Emma Graham-Harrison. “Revealed: 50 million Facebook profiles harvested for Cambridge Analytica in major data breach”. *The Guardian*, 17. marts 2018. https://www.theguardian.com/news/2018/mar/17/cambridge-analytica-facebook-influence-us-election.

Chacon, Scott, og Ben Straub. *Pro Git*. Set 13. april 2018. https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control.

Chen, S.C. “PHP Simple HTML DOM Parser”. Sourceforge. Set 13. april 2018. http://simplehtmldom.sourceforge.net/.

cURL. “CA certificates extracted from Mozilla”. Set 16. april 2018. https://curl.haxx.se/docs/caextract.html.

D. Hardt Ed. “The OAuth 2.0 Authorization Framework”. *Internet Engineering Task Force (IETF)*, 2012. https://tools.ietf.org/html/rfc6749.

Font Awesome. “License”. Set 13. april 2018. https://fontawesome.com/license.

Google. “Best practices for securely using API keys”. Cloud Platform Console Help. Set 14. april 2018. https://support.google.com/cloud/answer/6310037?hl=en.

———. “Material Design - Color”. Material.io. Set 13. april 2018. https://material.io/guidelines/style/color.html#color-themes.

“Google API PHP Client Library”. *Github repository*. Set 13. april 2018. https://github.com/google/google-api-php-client.

"Hacking og DDoS", 15. januar 2018. https://www.politi.dk/da/borgerservice/anmeldelser/hacking/.

Lundström, Eías Christian. "It-advokat om Tingbogen-kopi: Det kan give bøder at scrape offentlige data". *Version2*, 28. januar 2016. https://www.version2.dk/artikel/it-advokat-tingbogen-kopi-kan-give-boeder-at-scrape-offentlige-data-570672.

Mozilla. "CA certdata". Set 16. april 2018. https://hg.mozilla.org/releases/mozilla-release/raw-file/default/security/nss/lib/ckfw/builtins/certdata.txt.

PHP. "MySQL Improved Extension". php.net. Set 13. april 2018. https://secure.php.net/manual/en/book.mysqli.php.

Scrapers, Screen. *Webbot Spider and Screen Scrapers*, 2007.

Svansø, Vibeke Lyngklip. "Oplysninger om danske skoleelever flyder frit på populær lektieplatform". *Berlingske Business*, 11. april 2017. https://www.business.dk/digital/oplysninger-om-danske-skoleelever-flyder-frit-paa-populaer-lektieplatform.

Turland, Matthew. *php|architect's Guide to Web Scraping with PHP*. MarcoTabini & Associates, Inc., 2010.

Twilio. "Twilio". Set 13. april 2018. https://www.twilio.com/.

W3Schools. "HTML Global data-* Attributes". w3schools.com. Set 13. april 2018. https://www.w3schools.com/tags/att_global_data.asp.

Bilag

Koden

Nedenstående har jeg vedhæftet al den kode, som jeg selv har udarbejdet. Jeg har ikke inkluderet koden fra bibliotekerne, jeg har anvendt. Min kode kan også findes på Github.

```
https://github.com/AndreasLF/LectioScrape
```

Jeg har kommenteret al min kode på engelsk, da engelsk er sproget, man for det meste anvender i erhvervslivet, samtidig med at mange udtryk ikke findes på dansk.

Opsætning af database

Følgende kode, er den jeg bruger til at oprette min database samt tabellen i databasen. Jeg har oprettet en ny bruger på min server, som kun har adgang til at ændre i LectioDB-databasen. Dette er ikke inkluderet i nedenstående kode. Det skal gøres manuelt på serveren.

```
-- phpMyAdmin SQL Dump
-- version 4.6.4
-- https://www.phpmyadmin.net/
--
-- Host: 127.0.0.1
-- Generation Time: Apr 13, 2018 at 09:55 AM
-- Server version: 5.7.14
-- PHP Version: 5.6.25

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET AUTOCOMMIT = 0;
START TRANSACTION;
SET time_zone = "+00:00";

--
-- Database: `lectio`
--
CREATE DATABASE IF NOT EXISTS `lectio` DEFAULT CHARACTER SET latin1 COLLATE latin1_swedish_ci;
USE `lectio`;

-- --------------------------------------------------------

--
-- Table structure for table `skema`
```

```
--

DROP TABLE IF EXISTS `skema`;
CREATE TABLE `skema` (
  `ID` int(11) NOT NULL,
  `Week` int(8) DEFAULT NULL,
  `Status` varchar(10) DEFAULT NULL,
  `Description` varchar(1200) DEFAULT NULL,
  `Date` date DEFAULT NULL,
  `StartTime` time DEFAULT NULL,
  `EndTime` time DEFAULT NULL,
  `Class` varchar(1200) DEFAULT NULL,
  `Teacher` varchar(1200) DEFAULT NULL,
  `Room` varchar(1200) DEFAULT NULL,
  `Homework` varchar(1200) DEFAULT NULL,
  `Note` varchar(1200) DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

--
-- Indexes for dumped tables
--

--
-- Indexes for table `skema`
--
ALTER TABLE `skema`
  ADD PRIMARY KEY (`ID`);

--
-- AUTO_INCREMENT for dumped tables
--

--
-- AUTO_INCREMENT for table `skema`
--
ALTER TABLE `skema`
  MODIFY `ID` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=3458;COMMIT;
```

index.html

```
<!DOCTYPE HTML>

<html>

<head>
    <title>LectioScrape</title>

    <!--Charset-->
```

```
    <meta charset="UTF-8" />

    <!--Jquery-->
    <script src='lib/jquery.min.js'></script>
    <!--Bootstrap CSS-->
    <link rel="stylesheet" href="Bootstrap/css/bootstrap.css">
    <!--Bootstrap JS-->
    <script src="Bootstrap/js/bootstrap.js"></script>
    <!--Fontawesome-->
    <link href="https://use.fontawesome.com/releases/v5.0.8/css/all.css" rel="stylesheet">
    <!--Fullcalendar CSS-->
    <link rel='stylesheet' href='fullcalendar/fullcalendar.css' />
    <!--Moment.js-->
    <script src='lib/moment.min.js'></script>
    <!--Fullcalendar JS-->
    <script src='fullcalendar/fullcalendar.js'></script>


    <script src="script.js"></script>

</head>

<body>


    <!--Nav bar-->
    <nav class="navbar navbar-expand-lg navbar-light bg-light">
        <a class="navbar-brand" href="#">LectioScrape</a>
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle
navigation">
        <span class="navbar-toggler-icon"></span>
    </button>
        <div class="collapse navbar-collapse" id="navbarNav">
            <ul class="navbar-nav">
                <li class="nav-item active">
                    <a class="nav-link" href="#">Skema <span class="sr-only">(current)</span></a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="#">Om siden</a>
                </li>
            </ul>
        </div>
    </nav>
    <br>


    <!--Container-->
    <div class="container">
        <!--Container-->
        <div class="container">
```

```
            <!--Row with buttons-->
            <div class="row">
                <div class="col-6 text-center">
                    <button type="button" class="btn btn-primary" id="databaseButton">Backup i
database</button>
                </div>
                <div class="col-6 text-center">
                    <button type="button" class="btn btn-primary" id="googleButton">Sync with
Google</button>
                </div>
            </div>
        </div>

        <!--Full calendar modal. Displays when an event is clicked-->
        <div id="fullCalendarModal" class="modal" tabindex="-1" role="dialog">
            <div class="modal-dialog" role="document">
                <div class="modal-content">
                    <div class="modal-header">
                        <h5 id="eventModalTitle" class="modal-title">Modal title</h5>
                        <button type="button" class="close" data-dismiss="modal" aria-
label="Close">
                            <span aria-hidden="true">&times;</span>
                        </button>
                    </div>
                    <div id="eventModalBody" class="modal-body">
                    </div>
                    <div class="modal-footer">
                        <a id="eventModalLectioButtonLink" href="" target="_blank">     <button
type="button" class="btn btn-primary">Lectio</button></a>
                        <button type="button" class="btn btn-secondary" data-
dismiss="modal">Luk</button>
                    </div>
                </div>
            </div>
        </div>

        <br>

        <!--Fullcalendar-->
        <div id='calendar'></div>
    </div>
</body>

</html>
```

script.js

```
$(document).ready(function() {
```

```
            //When the page is ready the calendar will be initialized
            $('#calendar').fullCalendar({
                //Links to the json feed
                events: 'fullcalendarFeed.php',
                editable: false,
                header: {
                    left: 'prev',
                    center: '',
                    right: 'next'
                },
                //Only view one week at a time
                defaultView: 'agendaWeek',
                minTime: '07:00:00',
                maxTime: '19:00:00',
                themeSystem: 'bootstrap4',
                displayEventTime: true,
                weekNumbers: true,
                timeFormat: 'H(:mm)',
                slotEventOverlap: false,
                eventOverlap: false,
                //Set event click listener. When an event on the calendar is clicked, this will
trigger
                eventClick: function(event, jsEvent, view) {
                    //Sets modal information
                    $('#eventModalTitle').html(event.title);
                    $('#eventModalBody').html(event.description);
                    $('#eventModalLectioButtonLink').prop("href",event.lessonURL);
                    //Opens the modal
                    $('#fullCalendarModal').modal();
                },
                firstDay: 1
            })


            //When googleButton is clicked
            $("#googleButton").click(function(){

                //Gets the current view from the calendar
                var view = $('#calendar').fullCalendar('getView');

                //Formats the start date in the view with Moment
                var startDate = view.start.format();

                //Formats the end date in the view with Moment
                var endDate = view.end.format();

                //Open uploadToGoogleCal.php and pass startDate and endDate via $_GET
                window.location.replace("setDatesSession.php?startDate=" + startDate + "&endDate="
+ endDate );

            });
```

```
            //When databaseButton is clicked
            $("#databaseButton").click(function(){

                //Gets the current view from the calendar
                var view = $('#calendar').fullCalendar('getView');

                //Formats the start date in the view with Moment
                var startDate = view.start.format();

                //Formats the end date in the view with Moment
                var endDate = view.end.format();

                //Open sendScheduleToDatabase.php and pass startDate and endDate via $_GET
                window.location.replace("sendScheduleToDatabase.php?startDate=" + startDate +
"&endDate=" + endDate );

            });


        });
```

Lesson.class.php

```
<?php
/**
* Lesson
* Creates a lesson object containing information about the lesson
*
* @author Andreas Fiehn
*/
class Lesson{
    /** @var string $lessonURL*/
    public $lessonURL;
    /** @var string $status*/
    public $status;
    /** @var string $description*/
    public $description;
    /** @var string $date*/
    public $date;
    /** @var string $startTime*/
    public $startTime;
    /** @var string $endTime*/
    public $endTime;
    /** @var string $class*/
    public $class;
    /** @var string $teacher*/
    public $teacher;
    /** @var string $room*/
```

```
    public $room;
    /** @var string $room*/
    public $homework;
    /** @var string $additionalContent*/
    public $additionalContent;
    /** @var string $note*/
    public $note;
    /** @var string $students*/
    public $students;


    /**
    * This method constructs the object from the lesson class
    * @param String $data contains the lesson data as a string. The lesson data is fetched from
Lectio and is the value of the data-additionalinfo attribute.
    * @param String $url is the lessons web page url
    */
    function __construct($data,$url){

        //The url is set
        $this->lessonURL = $url;

        //Sets every property by refering to methods
        $this->setStatusAndDescription($data);
        $this->setDate($data);
        $this->setTime($data);
        $this->setClass($data);
        $this->setTeacher($data);
        $this->setRoom($data);
        $this->setHomework($data);
        $this->setAdditionalContent($data);
        $this->setNote($data);
        $this->setStudents($data);
    }


    /**
    * This method sets the lesson's status and description, by performing regular expressions
    * @param String $data contains the lesson data as a string. The lesson data is fetched from
Lectio and is the content of the data-additionalinfo attribute.
    */
    private function setStatusAndDescription($data){
        //Checks if the lesson has been changed by performing a regex
        if(preg_match('/^Ændret!/',$data)){
            //Sets the status
            $this->status = 'Ændret';

            //Checks if there is a string between "Ændret!" and the date. Here the lesson
descritption will show up if there is any
            if(preg_match('/^Ændret!\s(.*?)\s((\d\d|\d)\/(\d\d|\d)-(\d\d\d\d))/', $data,
$matches)){
```

```
                //The description is set
                $this->description = $matches[1];
             }

        }
        //Checks if the lesson has been cancelled by performing a regex
        else if(preg_match('/^Aflyst!/', $data)){
            //Sets the status
            $this->status = 'Aflyst';

            //If there is a string between "Aflyst!" and the date. Here the lesson descritption
will show up if there is any.
            if(preg_match('/^Aflyst!\s(.*?)\s((\d\d|\d)\/(\d\d|\d)-(\d\d\d\d))/', $data,
$matches)){
                //Sets the description
                $this->description = $matches[1];
            }
        }
        //Else if there is a string before the date and no status message. Here the lesson
description will show up if there is any.
        else if(preg_match('/(.*?)\s((\d\d|\d)\/(\d\d|\d)-(\d\d\d\d))/', $data, $matches)){
            //Status is null
            $this->status = NULL;
            //Sets description
            $this->description = $matches[1];
        }
        else{
            //Status is null
            $this->status = NULL;
            //Description is null
            $this->description = NULL;
        }
    }


    /**
    * By performing a regex the date of the lesson is retrieved and and saved.
    * @param String $data contains the lesson data as a string. The lesson data is fetched from
Lectio and is the content of the data-additionalinfo attribute.
    */
    private function setDate($data){
        //Perform a regex on $data. Searches for a date pattern
        if(preg_match('/(\d\d|\d)\/(\d\d|\d)-(\d\d\d\d)/', $data, $matches)){


            //If the retrieved day only contains 1 digit a '0' is added to ensure, that the format
complies with MySQL's date format.
            if(strlen($matches[1])==1){
                $dd = "0".$matches[1];
            }
            else{
```

```
                $dd = $matches[1];
            }


            //If the retrieved month only contains 1 digit a '0' is added to ensure, that the
format complies MySQL's date format.
            if(strlen($matches[2])==1){
                $mm = "0".$matches[2];
            }
            else{
                $mm = $matches[2];
            }

            //The year is defined
            $yyyy = $matches[3];

            //The date property is updated to contain the date in the correct format
            $this->date = $yyyy."-".$mm."-".$dd;
        }
        else{
            //If no date exists the date property will be null
            $this->date = NULL;
        }
    }


    /**
    * This method saves the start and end time for the lesson
    * @param String $data contains the lesson data as a string. The lesson data is fetched from
Lectio and is the content of the data-additionalinfo attribute.
    */
    private function setTime($data){
        if(preg_match('/(\d\d:\d\d)\stil\s(\d\d:\d\d)/', $data, $matches)){

            //Splits the matches at every ':'. The result is returned as an array.
            $startTimeArray = preg_split("/[:]+/",$matches[1]);
            $endTimeArray = preg_split("/[:]+/",$matches[2]);

            //Concatenates the results and adds '00' to the end to make sure the format is correct
            $this->startTime = $startTimeArray[0].":".$startTimeArray[1].":"."00";
            $this->endTime= $endTimeArray[0].":".$endTimeArray[1].":"."00";
        }
        else{
            $this->startTime = NULL;
            $this->endTime = NULL;
        }
    }

    /**
    * Saves the class information on the lesson
```

```
    * @param String $data contains the lesson data as a string. The lesson data is fetched from
Lectio and is the content of the data-additionalinfo attribute.
    */
    private function setClass($data){
        //Searches for the content between 'Hold: ' and ' Lærer'. The '?' makes it non-greedy,
which means it will search for the shortest string between the two.
        if(preg_match('/Hold:\s(.*?)\sLærer/', $data, $matches)){
            //Converts the class and sets it as the class property
            $this->class = $this->convertClass($matches[1]);
        }
        else{
            $this->class = NULL;
        }
    }


    /**
    * Converts the classId to a class name
    * @param String $classId is the class id
    * @return String
    */
    private function convertClass($classId){
        //Checks for known class codes and returns a string corresponding to the code
        if(preg_match('/DA/', $classId)){
            return "Dansk";
        }
        else if(preg_match('/MA|Ma/', $classId)){
            return "Matematik";
        }
        else if(preg_match('/FY|Fy/', $classId)){
            return "Fysik";
        }
        else if(preg_match('/TK/', $classId)){
            return "Teknik";
        }
        else if(preg_match('/EN|En/', $classId)){
            return "Engelsk";
        }
        else if(preg_match('/KE|Ke/', $classId)){
            return "Kemi";
        }
        else if(preg_match('/Itk/', $classId)){
            return "Informationsteknologi";
        }
        else if(preg_match('/pro/', $classId)){
            return "Programmering";
        }
        else if(preg_match('/Ti/', $classId)){
            return "Teknologi";
        }
        else if(preg_match('/th/', $classId)){
```

```
            return "Teknologihistorie";
        }
        else if(preg_match('/Sa/', $classId)){
            return "Samfundsfag";
        }
        else if(preg_match('/SO|SOf/', $classId)){
            return "Studieområde";
        }
        else if(preg_match('/Andenaktiv/', $classId)){
            return "Anden aktivitet";
        }
        else{
            //If the code was not recognized the classId will be returned unchanged
            return $classId;
        }

    }


    /**
    * Saves the teacher on the lesson
    * @param String $data contains the lesson data as a string. The lesson data is fetched from
Lectio and is the content of the data-additionalinfo attribute.
    */
    private function setTeacher($data){
        //If only one teacher is on the schedule, his name will be shown following by his initials
        if(preg_match('/Lærer:\s(.*?)\s\((.*?)\)\sLokale/', $data, $matches)){
            //Sets the teacher property
            $this->teacher = $matches[1];
        }
        //If more than one teacher is on the schedule only the initials will be shown seperated by
commas. I search for all the content between 'Lærere: ' and ' Lokale'
        else if(preg_match('/Lærere:\s(.*?)\sLokale/', $data, $matches)){
            //Sets the teacher property
            $this->teacher = $matches[1];
        }
        else{
            //Sets the teacher property
            $this->teacher = NULL;
        }
    }


    /**
    * Saves the room information on the lesson
    * @param String $data contains the lesson data as a string. The lesson data is fetched from
Lectio and is the content of the data-additionalinfo attribute.
    */
    private function setRoom($data){

        //Gets all the content after 'Lokale: ' or 'Lokaler: '
```

```
        if(preg_match('/Lokale\S+\s(.*)/', $data, $matches)){

            $this->room = $matches[1];

            //Sorts the string if there is a note, homework, additionalContent or elever
            if(preg_match('/(.*)\sElever:/', $matches[1], $matchesSorted)){
                $this->room = $matchesSorted[1];
            }
            if(preg_match('/(.*?)\sNote/', $matches[1],$matchesSorted)){
                $this->room = $matchesSorted[1];
            }
            if(preg_match('/(.*)\sØvrigt\sindhold:/', $matches[1],$matchesSorted)){
                $this->room = $matches[1];
            }
            if(preg_match('/(.*?)\sLektier/', $matches[1],$matchesSorted)){
                $this->room = $matchesSorted[1];
            }
        }
        else {
            $this->room = NULL;
        }
    }


    /**
    * Saves the homework on the lesson
    * @param String $data contains the lesson data as a string. The lesson data is fetched from
Lectio and is the content of the data-additionalinfo atribute.
    */
    private function setHomework($data) {
        //Gets all the content after "Lektier: "
        if(preg_match('/Lektier:\s(.*)/', $data, $matches)){
            $this->homework = $matches[1];

            if(preg_match('/(.*)\sElever:/', $matches[1], $matchesSorted)){
                $this->homework = $matchesSorted[1];
            }
            if(preg_match('/(.*?)\sNote/', $matches[1],$matchesSorted)){
                $this->homework = $matchesSorted[1];
            }
            if(preg_match('/(.*)\sØvrigt\sindhold:/', $matches[1],$matchesSorted)){
                $this->homework = $matches[1];
            }

        }
        else {
            $this->homework = NULL;
        }
    }
```

```
    /**
    * Saves the additional content for the lesson
    * @param String $data contains the lesson data as a string. The lesson data is fetched from
Lectio and is the content of the data-additionalinfo attribute.
    */
    private function setAdditionalContent($data){
        //Gets all the content after "Øvrigt indhold: "
        if(preg_match('/Øvrigt\sindhold:\s(.*)/', $data, $matches)){
            $this->additionalContent = $matches[1];

             if(preg_match('/(.*)\sElever:/', $matches[1], $matchesSorted)){
                $this->additionalContent = $matchesSorted[1];
            }
            if(preg_match('/(.*?)\sNote/', $matches[1],$matchesSorted)){
                $this->additionalContent = $matchesSorted[1];
            }

        }
        else {
            $this->additionalContent = NULL;
        }
    }


    /**
    * Saves the notes on the lesson
    * @param String $data contains the lesson data as a string. The lesson data is fetched from
Lectio and is the content of the data-additionalinfo attribute.
    */
    private function setNote($data){
        //Gets all content after "Note: "
        if(preg_match('/Note:\s(.*)/', $data, $matches)){
            $this->note = $matches[1];
            if(preg_match('/(.*)\sElever:/', $matches[1], $matchesSorted)){
                $this->note = $matchesSorted[1];
            }
        }
        else{
            $this->note = NULL;
        }
    }

    /**
    * Saves the students on the lesson
    * @param String $data contains the lesson data as a string. The lesson data is fetched from
Lectio and is the content of the data-additionalinfo class.
    */
    private function setStudents($data){
        //Gets all content after "Elever: "
        if(preg_match('/Elever:\s(.*)/', $data, $matches)){
```

```
            $this->students = $matches[1];
        }
        else{
            $this->students = NULL;
        }
    }
}
?>
```

LessonFullcalendar.class.php

```
<?php

/**
* LessonFullcalendar
* This creates an object ready for fullcalendar
*
* @author Andreas Fiehn
*/
class LessonFullcalendar{

    /** @var array containing calendar event in Fullcalendar format*/
    private $calendarEvent;

    /** @var string */
    private $title;

    /** @var string */
    private $description;

    /** @var array */
    private $startEvent;

    /** @var array */
    private $endEvent;

    /** @var string */
    private $url;

    /** @var string */
    private $color;


    /**
    * Constructs the Event object ready for Fullcalendar
    * @param $lessonObject is the object from which the calendar event is created
    */
    function __construct($lessonObject){
        //Sets the calendar parameters
```

```
        $this->setTitle($lessonObject);
        $this->setStartEvent($lessonObject);
        $this->setDescription($lessonObject);
        $this->setEventColor($lessonObject);
        $this->setStartEvent($lessonObject);
        $this->setEndEvent($lessonObject);
        $this->setURL($lessonObject);

        //Creates the calendar event
        $this->setCalendarEvent();
    }


    /**
    * This gets the fullcalendar event
    *
    * @return array
    */
    public function getCalendarEvent(){
        return $this->calendarEvent;
    }


    /**
    * This sets the calendarEvent property
    */
    private function setCalendarEvent(){
        $this->calendarEvent = array(
            'title'=>$this->title,
            'allday'=>$this->startEvent['allDay'],
            'description'=>$this->description,
            'color'=>$this->color,
            'textColor'=>'#ffffff',
            'start'=>$this->startEvent['start'],
            'end'=>$this->endEvent['end'],
            'lessonURL'=>$this->url
        );
    }


    /**
    * Sets the lesson's summary
    * @param $lessonObject is the lesson object created from the Lesson class
    */
    private function setTitle($lessonObject){
        //If lessonObject contains a description and class
        if($lessonObject->description && $lessonObject->class){
            //If lessonObject contains a status
            if($lessonObject->status){
                //The title will contain the status, the description and the class in parantheses
```

```
                $this->title = $lessonObject->status. "!\n". $lessonObject->description .
"\n(".$lessonObject->class.")";
            }
            else{
                //The title will contain the description and class in parentheses
                 $this->title = $lessonObject->description . "\n(".$lessonObject->class.")";
            }
        }
        //Else if lessonObject only contains a class
        else if($lessonObject->class){
            //Checks if lessonObject contains a status
            if($lessonObject->status){
                //The title will contain staus and class
                $this->title = $lessonObject->status. "!\n". $lessonObject->class;
            }
            else{
                //The lesson will only contain the class
                $this->title = $lessonObject->class;
            }
        }
        else{
            //Checks if lessonObject contains status
            if($lessonObject->status){
                //The title will contain status and description
                $this->title = $lessonObject->status. "!\n". $lessonObject->description;
            }
            else{
                //The title will contain only the description
                $this->title = $lessonObject->description;
            }
        }
    }

    /**
    * Set the lesson description
    * @param $lessonObject is the lesson object created from the Lesson class
    */
    private function setDescription($lessonObject){

        //An array is created
        $descriptionArray = array();

        //Checks for prperties in the lessonObject. If a property exists it will be added to the
description array
        if($lessonObject->status){
            $descriptionArray[] = $lessonObject->status;
        }

        if($lessonObject->description){
            $descriptionArray[] = $lessonObject->description;
        }
```

```
        if($lessonObject->teacher){
            $descriptionArray[] = "<b>Lærer: </b>" . $lessonObject->teacher;
        }

        if($lessonObject->room){
            $descriptionArray[] = "<b>Lokale: </b>" . $lessonObject->room;
        }

        if($lessonObject->homework){
            $descriptionArray[] = "<b>Lektier: </b>" . $lessonObject->homework;
        }

         if($lessonObject->additionalContent){
            $descriptionArray[] = "<b>Øvrigt indhold: </b>" . $lessonObject->additionalContent;
        }

        if($lessonObject->note){
            $descriptionArray[] = "<b>Note: </b>" . $lessonObject->note;
        }

        if($lessonObject->students){
            $descriptionArray[] = "<b>Elever: </b>" . $lessonObject->students;
        }


        //Description string is created
        $descriptionString = "";

        //For each element in the descriptionArray the element will be added to the
descriptionString
        foreach($descriptionArray as $desc){
            $descriptionString = $descriptionString . $desc . "<br>";

        }

        //The description property is set to contain the description string
        $this->description = $descriptionString;
    }

    /**
    * Set the start time and timezone of the lesson
    * @param $lessonObject is the lesson object created from the Lesson class
    */
    private function setStartEvent($lessonObject){

        //If startTime exists the event wont last all day
        if($lessonObject->startTime){
            $this->startEvent = array(
                'start' => $lessonObject->date.'T'.$lessonObject->startTime,
                'allDay' => false
```

```
                );
        }
        else {
            $this->startEvent = array(
                'start' => $lessonObject->date,
                'allDay' => true
                );
        }

    }

    /**
    * Sets the start time and timezone of the lesson
    * @param $lessonObject is the lesson object created from the Lesson class
    */
    private function setEndEvent($lessonObject){

        //If endTime exists the event wont last all day
        if($lessonObject->endTime){
            $this->endEvent = array(
                'end' => $lessonObject->date.'T'.$lessonObject->endTime,
                'allDay' => false
                );
        }
        else {
            $this->endEvent = array(
                'end' => $lessonObject->date,
                'allDay' => true
                );
        }
    }


    /**
    * Sets the lesson URL
    * @param $lessonObject is the lesson object created from the Lesson class
    */
    private function setURL($lessonObject){
        $this->url = $lessonObject->lessonURL;
    }

    /**
    * Sets the event color
    * @param $lessonObject is the lesson object created from the Lesson class
    */
    private function setEventColor($lessonObject){
        //Defines an array of colors (hex)
        $colorsArray = array(
            'red'=>'#d32f2f',
            'pink'=>'#c2185b',
            'purple'=>'#7b1fa2',
```

```
            'deep purple'=>'#512da8',
            'indigo'=>'#303f9f',
            'blue'=>'#1976d2',
            'light blue'=>'#0288d1',
            'cyan'=>'#0097a7',
            'teal'=>'#00796b',
            'green'=>'#388e3c',
            'light green'=>'#689f38',
            'lime'=>'#afb42b',
            'yellow'=>'#fbc02d',
            'amber'=>'#ffa000',
            'orange'=>'#f57c00',
            'deep orange'=>'#e64a19',
            'brown'=>'#5d4037',
            'gray'=>'#616161',
            'blue gray'=>'#455a64'
        );

        //If the lesson is changed or cancelled the color will be set
        if($lessonObject->status == 'Ændret'){
            $this->color = $colorsArray['green'];
        }
        else if($lessonObject->status == 'Aflyst'){
            $this->color = $colorsArray['red'];
        }
        else{
            //Checks if the event is an all day event and sets the color
            if($lessonObject->startTime == NULL && $lessonObject->endTime == NULL){
                $this->color = $colorsArray['indigo'];
            }
            //Checks the class of the lesson and sets a corresponding color
            else if($lessonObject->class == "Matematik"){
                $this->color = $colorsArray['blue'];
            }
            else if($lessonObject->class == "Dansk"){
                $this->color = $colorsArray['orange'];
            }
            else if($lessonObject->class == "Fysik"){
                $this->color = $colorsArray['teal'];
            }
            else if($lessonObject->class == "Kemi"){
                $this->color = $colorsArray['light green'];
            }
            else if($lessonObject->class == "Samfundsfag"){
                $this->color = $colorsArray['gray'];
            }
            else if($lessonObject->class == "Informationsteknologi"){
                $this->color = $colorsArray['purple'];
            }
            else if($lessonObject->class == "Programmering"){
                $this->color = $colorsArray['purple'];
```

```
            }
            else if($lessonObject->class == "Teknik"){
                $this->color = $colorsArray['pink'];
            }
            else{
                $this->color = $colorsArray['blue gray'];
            }
        }
    }
}

?>
```

LessonGoogleCalEvent.class.php

```
<?php

/**
* LessonGoogleCalEvent
* This creates an object readable by the Google Calendar API from the Lesson object
*
* @author Andreas Fiehn
*/
class LessonGoogleCalEvent{
    /** @var array $eventParams */
    //These are the Google Calendar event parameters, which can be passed to the Google Calendar
API
    private $eventParams;

    /** @var string $sumamry */
    private $summary;
    /** @var string $description */
    private $description;
    /** @var array $startEvent */
    private $startEvent;
    /** @var array $endEvent */
    private $endEvent;
    /** @var string $colorId */
    private $colorId;


    /**
    * Constructs the Google event object
    * @param $lessonObject is the object from which the calendar event is created from
    */
    function __construct($lessonObject){
        //sets calendar parameters
        $this->setSummary($lessonObject);
        $this->setDescription($lessonObject);
        $this->setStartEvent($lessonObject);
```

```
        $this->setEndEvent($lessonObject);
        $this->setColorId($lessonObject);

        //Creates the eventParams array
        $this->setEventParams();
    }


    /**
    * This gets the google calendar event parameters
    *
    * @return array
    */
    public function getEventParams(){
        return $this->eventParams;
    }

    /**
    * Sets the google calendar event parameters
    */
    private function setEventParams(){
        $this->eventParams = array(
        'summary' => $this->summary,
        'description' => $this->description,
        'start' => $this->startEvent,
        'end' => $this->endEvent,
        'colorId' => $this->colorId
        );
    }

    /**
    * Returns the lesson's summary
    * @param $lessonObject is the lesson object created from the Lesson class
    * @return string
    */
    private function setSummary($lessonObject){
        //If lessonObject contains class
        if($lessonObject->class){
            //If a status exists
            if($lessonObject->status){
                $this->summary = $lessonObject->status . "! " . $lessonObject->class;
            }
            else{
                $this->summary = $lessonObject->class;
            }
        }
        else{
            if($lessonObject->status){
                $this->summary = $lessonObject->status . "! " . $lessonObject->description;
            }
            else{
```

```
                $this->summary = $lessonObject->description;
            }
        }


    }

    /**
    * Returns the lesson description
    * @param $lessonObject is the lesson object created from the Lesson class
    * @return string
    */
    private function setDescription($lessonObject){
        //Creates description array
        $descriptionArray = array();

        //If a lessonObject property exists it wil be added to the array
        if($lessonObject->status){
            $descriptionArray[] = $lessonObject->status;
        }

        if($lessonObject->description){
            $descriptionArray[] = $lessonObject->description;
        }

        if($lessonObject->teacher){
            $descriptionArray[] = "Lærer: " . $lessonObject->teacher;
        }

        if($lessonObject->room){
            $descriptionArray[] = "Lokale: " . $lessonObject->room;
        }

        if($lessonObject->homework){
            $descriptionArray[] = "Lektier: " . $lessonObject->homework;
        }

        if($lessonObject->note){
            $descriptionArray[] = "Note: " . $lessonObject->note;
        }

        //Description string is defined
        $descriptionString = "";

        //For each element in description array, the description is added to the string
        foreach($descriptionArray as $desc){
            $descriptionString = $descriptionString . $desc . "<br>";

        }

        //Sets the description property
```

```
        $this->description = $descriptionString;
    }

    /**
    * Returns the start time and timezone of the lesson
    * @param $lessonObject is the lesson object created from the Lesson class
    * @return array
    */
    private function setStartEvent($lessonObject){
        //If lessonObject does not have a startTime the event will last all day
        if($lessonObject->startTime){
            $this->startEvent = array(
            'dateTime' => $lessonObject->date.'T'.$lessonObject->startTime,
            'timeZone' => 'Europe/Copenhagen',
            );
        }
        else {
            $this->startEvent = array(
            'date' => $lessonObject->date,
            'timeZone' => 'Europe/Copenhagen',
            );
        }

    }

     /**
    *  Returns the start time and timezone of the lesson
    * @param $lessonObject is the lesson object created from the Lesson class
    * @return string
    */
    private function setEndEvent($lessonObject){
        //If lessonObject does not have an endTime the event will last all day
        if($lessonObject->endTime){
            $this->endEvent = array(
            'dateTime' => $lessonObject->date.'T'.$lessonObject->endTime,
            'timeZone' => 'Europe/Copenhagen',
            );
        }
        else {
            $this->endEvent = array(
            'date' => $lessonObject->date,
            'timeZone' => 'Europe/Copenhagen',
            );
        }
    }

    /**
    * Returns the color id for the event
    * @param $lessonObject is the lesson object created from the Lesson class
    * @return string
    */
```

```
    private function setColorId($lessonObject){
        //Defines the colors in an array (google calendar color id)
        $colorsArray = array(
        'blue' => '1',
        'green'=>'2',
        'purple'=>'3',
        'red'=>'4',
        'yellow'=>'5',
        'orange' => '6',
        'turquoise'=>'7',
        'gray'=>'8',
        'bold blue'=>'9',
        'bold green'=>'10',
        'bold red' => '11',
        );

        //Checks for information and sets the color corresponding to the information

        if($lessonObject->status == 'Ændret'){
            $this->colorId = $colorsArray['bold green'];
        }
        else if($lessonObject->status == 'Aflyst'){
            $this->colorId = $colorsArray['bold red'];

        }
        else{

            if($lessonObject->startTime == NULL && $lessonObject->endTime == NULL){
                $this->colorId = $colorsArray['bold blue'];
            }
            else if($lessonObject->class == "Matematik"){
                $this->colorId = $colorsArray['turquoise'];
            }
            else if($lessonObject->class == "Dansk"){
                $this->colorId = $colorsArray['orange'];
            }
            else if($lessonObject->class == "Fysik"){
                $this->colorId = $colorsArray['red'];
            }
            else if($lessonObject->class == "Kemi"){
                $this->colorId = $colorsArray['yellow'];
            }
            else if($lessonObject->class == "Samfundsfag"){
                $this->colorId = $colorsArray['gray'];
            }
            else if($lessonObject->class == "Informationsteknologi"){
                $this->colorId = $colorsArray['purple'];
            }
            else if($lessonObject->class == "Programmering"){
                $this->colorId = $colorsArray['purple'];
            }
```

```
            else if($lessonObject->class == "Teknik"){
                $this->colorId = $colorsArray['green'];
            }
            else{
                $this->colorId = $colorsArray['blue'];
            }

        }

    }

}


?>
```

LectioScrape.class.php

```
<?php

/**
* LectioScrape
* This creates a lectioScrape object containing the schedule for one week
*
* @author Andreas Fiehn
*/
class LectioScrape{

    /** @var string $weekNumber*/
    public $weekNumber;

    /** @var string $year*/
    public $year;

    /** @var array containing lesson objects */
    private $scheduleMySql;

    /** @var array containing lessonGoogleCalEvent objects */
    public $scheduleGoogle;

    /** @var array containing lessonFullcalendar objects */
    public $scheduleFullcalendar;

    /** @var mysqli object*/
    private $mysqli;



    /**
    * Constructs the LectioScrape object
```

```
    *
    * @param string $date is a date inside the week you want to scrape. The date is in ISO8601
format
    */
    function __construct($date){
                            //Includes simple_html_dom library
        require_once __DIR__.'/simple_html_dom.php';

        //Includes classes
        require_once __DIR__.'/Lesson.class.php';
        require_once __DIR__.'/LessonGoogleCalEvent.class.php';
        require_once __DIR__.'/LessonFullcalendar.class.php';

        //Sets the weekNumber and year
        $this->weekNumber = $this->getWeekNumberFromDate($date)['weekNumber'];
        $this->year = $this->getWeekNumberFromDate($date)['year'];

        //Scrapes the schedule
        $schedule = $this->scrapeLectio($date);

        //Sets the schedule parameters
        $this->scheduleMySql = $schedule['schedule'];
        $this->scheduleGoogle = $schedule['scheduleGoogle'];
        $this->scheduleFullcalendar = $schedule['scheduleFullcalendar'];
    }


    /**
    * This function scrapes the schedule for one week on lectio.dk
    *
    * @param string $date is a date inside the week you want to scrape. The date is in ISO8601
format
    *
    * @return array containing the schedule for one week in three formats - schedule (MySQL-
ready), scheduleGoogle (Google Cal-ready), scheduleFullcalendar (Fullcalendar-ready)
    */
    private function scrapeLectio($date){

        //gets the week number from the date and saves it in $date. $date is now an array
containing weekNumber and year
        $date = $this->getWeekNumberFromDate($date);

        //Sets the weekNumber and year properties
        $this->weekNumber = $date['weekNumber'];
        $this->year = $date['year'];

        //Sets the url's weekID, schoolID and studentID
        $weekID = $date['weekNumber'].$date['year'];
        $schoolID = "681";
        $studentID = "14742506655";
```

```
        //Creates the schedule url
        $lectioURL =
"http://www.lectio.dk/lectio/".$schoolID."/SkemaNy.aspx?type=elev&elevid=".$studentID."&week=".$we
ekID;



        //creates html-DOM from the URL (uses simple_html_dom library)
        $html = file_get_html($lectioURL);

        //Define schedule variables
        $schedule;
        $scheduleGoogle;
        $scheduleFullcalendar;

        //Loop foreach element in the html dom with the class s2skemabrik
        foreach($html->find('.s2skemabrik') as $element){
            //Gets the data-additionalinfo attribute from the element
            $data = $element->getAttribute('data-additionalinfo');

            //If a href attribute exists this is saved as the url. Else the lectioUrl will be the
url
            if($href = $element->href){
                $url = "https://www.lectio.dk" . $href;
            }
            else{
                $url = $lectioURL;
            }


            //Peforms a regular expression to check if the data-additionalinfo contains a date in
the desired format. If not it wont be processed
            if(!(preg_match('/(\d\d|\d)\/(\d\d|\d)-(\d\d\d\d)/', $data, $dateArr)==0)){

                //Lesson object are created
                $lesson = new Lesson($data,$url);
                $lessonGoogle = new LessonGoogleCalEvent($lesson);
                $lessonFullcalendar = new LessonFullcalendar($lesson);

                //The objects are added to the schedule arrays
                $schedule[] = $lesson;
                $scheduleGoogle[] = $lessonGoogle->getEventParams();
                $scheduleFullcalendar[] = $lessonFullcalendar->getCalendarEvent();
            }
        }

        //An array containing the three schedule arrays is returned
        return array('schedule' => $schedule,'scheduleGoogle' => $scheduleGoogle,
'scheduleFullcalendar' => $scheduleFullcalendar);
    }
```

```php
/**
* Gets the start and end date by specifying year and week number
*
* @param int $year is the year
* @param int $weekNumber is the weeknumber
*
* @return array containing weekStart and weekEnd
*/
private function getWeekStartAndEndDate($year,$weekNumber){

    //Creates a new dateTime object
    $dateTimeObject = new DateTime();

    //Sets the date by using the ISO 8601 standard, specifying year and week number
    $dateTimeObject->setISODate($year,$weekNumber);

    //Saves start date in an array
    $dateRangeArray['weekStart'] = $dateTimeObject->format('Y-m-d');

    //Add 6 days to the dateTime object
    $dateTimeObject->modify('+6 days');

    //Saves the new date in an array as the end date
     $dateRangeArray['weekEnd'] = $dateTimeObject->format('Y-m-d');

    //Returns the date range array
    return $dateRangeArray;
}



/**
* Gets the week number by specifying a date in the week
*
* @param string $date is the date
*
* @return array containing weekNumber and year
*/
private function getWeekNumberFromDate($date){

    //Creates ned datetime object from the date
    $dateTimeObject = new DateTime($date);
    //The weeknumber is saved
    $weekNumber = $dateTimeObject->format("W");
    //The year is saved
    $year = $dateTimeObject->format("Y");
    //Returns array containing week number and year
    return array('weekNumber' => $weekNumber,'year' => $year);
}
```

```
    /**
    * Sends the schedule to the MySQL database
    *
    * @return boolean true on succes, false on failure
    */
    public function sendToDatabase(){

        //Checks if the mysqli property is set
        if(!isset($this->mysqli)){
            //Creates a new mysqli obect
            $this->mysqli = new mysqli("localhost","LectioDB","password","lectio");
        }

        //Creates the weekID
        $weekID = $this->weekNumber.$this->year;

        //Deletes the week to prevent duplicates
        $r = $this->deleteFromDatabase($weekID);

        //If the week was not deleted from the database. False is returned
        if(!($r)){
            return false;
        }

        //Loops for each lesson in the scheduleMySql parameter
        foreach($this->scheduleMySql as $lesson){

            //Creates a prepared statement for the database
            $stmt = $this->mysqli->prepare("INSERT INTO skema(ID,Week, Status, Description, Date,
StartTime, EndTime, Class, Teacher, Room, Homework, Note) VALUES (NULL,?,?,?,?,?,?,?,?,?,?,?)");

            //If the prepared statement fails to be defined, false is returned
            if(!($stmt)){
                return false;
            }

            //Binds parameters to the prepared statement. Every parameter is of type String
            $result = $stmt->bind_param("sssssssssss",$weekID,$lesson->status,$lesson-
>description,$lesson->date,$lesson->startTime,$lesson->endTime,$lesson->class,$lesson-
>teacher,$lesson->room,$lesson->homework,$lesson->note);

            //If bind_param fails, false is returned
            if(!($result)){
                return false;
            }

            //Executes the prepared statement
            $result = $stmt->execute();
```

```
            //If execute fails, false is returned
            if(!($result)){
                return false;
            }

            //Closes the prepared statement
            $stmt->close();

        }

        //true is returned on succes
        return true;
    }


    /**
    * Deletes a week from the database
    *
    * @param string $weekID is the week to delete and year to delete
    *
    * @return boolean true on succes, false on failure
    */
    private function deleteFromDatabase($weekID){

        //Creates a prepared statement for the database
        $stmt = $this->mysqli->prepare("DELETE FROM `skema` WHERE `Week`= ?");


        //If the prepared statement fails to be defined, false is returned
        if(!($stmt)){
            return false;
        }

        //Binds parameters to the prepared statement. Every parameter is of type String
        $result = $stmt->bind_param("s",$weekID);

        //If bind_param fails, false is returned
        if(!($result)){
            return false;
        }


        //Executes the prepared statement. Returns a boolean - true on succes and false on
failure.
        $result = $stmt->execute();

        //If execute fails, false is returned
        if(!($result)){
            return false;
        }
```

```
        //Closes the prepared statement
        $stmt->close();

        //return true on succes
        return true;
    }
}

?>
```

setDatesSession.php

```
<?php
/*This takes the start and end date passe with GET and saves it in the session*/

//Starts the session
session_start();

//Sets the startDate and endDate in session
$_SESSION['startDate'] = $_GET['startDate'];
$_SESSION['endDate'] = $_GET['endDate'];

//Redirects to uploadToGoogleCal.php
$redirect_uri = 'http://' . $_SERVER['HTTP_HOST'] . '/LectioScrape/uploadToGoogleCal.php';
header('Location: ' . filter_var($redirect_uri, FILTER_SANITIZE_URL));

?>
```

uploadToGoogleCal.php

```
<?php
session_start();

//Requires the Google Client Library
require_once __DIR__.'/vendor/autoload.php';

//Includes classes
require_once __DIR__.'/Lesson.class.php';
require_once __DIR__.'/LessonGoogleCalEvent.class.php';
require_once __DIR__.'/LectioScrape.class.php';



//Creates a new Google Client object
$client = new Google_Client();

//Sets the client authentification code from a json file
$client->setAuthConfig('client_secret.json');
```

```php
//Adds the Google calendar scope, that i want to acces in the API
$client->addScope(Google_Service_Calendar::CALENDAR);

$lectioCalendarExists = false;
$calendarId;

//Checks if the user's access token is stored in the session
if (isset($_SESSION['access_token']) && $_SESSION['access_token']) {


    //The access token is set in the client object
    $client->setAccessToken($_SESSION['access_token']);

    //Creates a Google_Service_Calendar object from the client object
    $service = new Google_Service_Calendar($client);

    //Gets a calendar list
    $calendarList = $service->calendarList->listCalendarList();

    //Checks if a calendar named 'LectioSkema' exists
    //Loops until a break occurs
    while(true) {
        foreach ($calendarList->getItems() as $calendarListEntry) {

            //If the calendarListEntry is LectioSkema
            if($calendarListEntry->getSummary()=="LectioSkema"){
                echo "LectioSkema already exists in Google Calendar <br>";
                $calendarId = $calendarListEntry->getId();
                $lectioCalendarExists = true;
            }

        }

        //Gets the next page token
        $pageToken = $calendarList->getNextPageToken();

        //If more calendarList pages exist ($pageToken == true)
        if ($pageToken) {
            //An optional parameter is set containing the page token
            $optParams = array('pageToken' => $pageToken);

            //A new calendarList is created with $optParams as input parameter
            //$optParams contains the next page token
            $calendarList = $service->calendarList->listCalendarList($optParams);
        }
        else {
            break;
        }
    }
```

```
    //If the calendar does not exist
    if(!$lectioCalendarExists){
        //Creates a new calendar objects
        $calendar = new Google_Service_Calendar_Calendar();

        //sets the calendar's summary
        $calendar->setSummary('LectioSkema');
        //Sets the calendar's time zone
        $calendar->setTimeZone('Europe/Copenhagen');

        //Creates the calendar
        $createdCalendar = $service->calendars->insert($calendar);

        //Echoes the calendar id for the new calendar
        $calendarId = $createdCalendar->getId();

        echo "LectioSkema created successfully";
    }


}
else {
    //If no auth token is stored in the SESSION variable, the browser gets redirected to
oauth2callback.php
    $redirect_uri = 'http://' . $_SERVER['HTTP_HOST'] . '/LectioScrape/oauth2callback.php';
    header('Location: ' . filter_var($redirect_uri, FILTER_SANITIZE_URL));
}

//Deletes the events from the week
deleteEvents($_SESSION['startDate'],$_SESSION['endDate'],$client,$service,$calendarId);

//New LectioScrape object is created
$schedule = new LectioScrape($_SESSION['startDate'].'T10:50:31');

//The schedule is sent to Google Cal
sendToGoogleCal($schedule->scheduleGoogle,$client,$service,$calendarId);

//Destroys the session
session_destroy();

$redirect_uri = 'http://' . $_SERVER['HTTP_HOST'] . '/LectioScrape/index.html';
header('Location: ' . filter_var($redirect_uri, FILTER_SANITIZE_URL));

/**
* Sends the schedule to Google calendar
*
* @param $scheduleList is a list of objects created from the LessonGoogleCalEvent class
* @param $client Google_Client object
* @param $service Google_Service_Calendar object
* @param $calendarId Google calendarId
*/
```

```php
function sendToGoogleCal($scheduleList,$client,$service,$calendarId){
    //Batch setup
    $client->setUseBatch(true);
    $batch = new Google_Http_Batch($client);

    foreach($scheduleList as $eventParams) {
        //Creates new event
        $event = new Google_Service_Calendar_Event($eventParams);

        //Inserts the event to the calendar
        $event = $service->events->insert($calendarId, $event);

        //Adds to the batch
        $batch->add($event);
    }

    //Executes batch
    $batch->execute();
}


/**
* Delete every event in the calendar between (including) $startDate and $endDate
*
* @param string $startDate is the start date in the following format: 'YYYY-MM-DD'
* @param string $endDate is the end date in the following format: 'YYYY-MM-DD'
* @param $client Google_Client object
* @param $service Google_Service_Calendar object
* @param $calendarId Google calendarId
*/
function deleteEvents($startDate,$endDate,$client,$service,$calendarId){

    //Specify minimum and maximum time to search for
    $timeMin = $startDate . 'T00:00:00+01:00';
    $timeMax = $endDate . 'T00:00:00+01:00';

    //List of items
    $eventItemsList;


    //Saves the timeMin and timeMax parameters in an array
    $optParams = array('timeMin' => $timeMin, 'timeMax'=>$timeMax);

    //creates the events list
    $events = $service->events->listEvents($calendarId,$optParams);

    //Add items to the list
    $eventItemsList[] = $events->getItems();

    //If more calendarList pages exist they will be added to the eventsItemsList array
    //Loops until a break occurs
```

```
    while(true){
        //Gets the next page token
        $pageToken = $events->getNextPageToken();

        if ($pageToken) {

            //The page token is added to the optParams
            $optParams = array('pageToken' => $pageToken,'timeMin' => $timeMin,
'timeMax'=>$timeMax);

            //A new list of events is created
            $events = $service->events->listEvents($calendarId, $optParams);

            //Add items to the list
            $eventItemsList[] = $events->getItems();

        }
        else {
            //Break free from while loop if no more pages exist
            break;
        }
    }

    //Batch setup
    $client->setUseBatch(true);
    $batch = new Google_Http_Batch($client);


    //Deletes every event in the events list

   foreach($eventItemsList as $eventItems){
        foreach ($eventItems as $event) {
            //Gets the event id
            $eventId = $event->getId();
            //Deletes the event
            $eventDeletion = $service->events->delete($calendarId, $eventId);

            //Add event deletion to the batch
            $batch->add($eventDeletion);
        }
   }

    //Batch execution
    $batch->execute();

}

?>
```

oauth2callback.php

```php
<?php
//Includes the google API
require_once __DIR__.'/vendor/autoload.php';

//Starts the session
session_start();

//Creates a new client object
$client = new Google_Client();
//Sets the Auth config file, which includes the client's secret key
$client->setAuthConfigFile('client_secret.json');
//Sets the redirect URI
$client->setRedirectUri('http://' . $_SERVER['HTTP_HOST'] . '/LectioScrape/oauth2callback.php');
//Adds the scope. I need access to the calendar
$client->addScope(Google_Service_Calendar::CALENDAR);


//If the $_GET variable does not contain a code from Google the browser is redirected to a login
page where they can grant access
if (! isset($_GET['code'])) {
    //The authentification URL is created
    $auth_url = $client->createAuthUrl();
    //The browser is redirected
    header('Location: ' . filter_var($auth_url, FILTER_SANITIZE_URL));
}
else {
    //If $_GET['code'] exists, an access token is created and stored in the session
    $client->authenticate($_GET['code']);
    $_SESSION['access_token'] = $client->getAccessToken();

    //The user gets redirected to the uploadToGoogleCal script
    $redirect_uri = 'http://' . $_SERVER['HTTP_HOST'] . '/LectioScrape/uploadToGoogleCal.php';
    header('Location: ' . filter_var($redirect_uri, FILTER_SANITIZE_URL));
}
```

sendScheduleToDatabase.php

```php
<?php
//Require the LectioScrape class
require_once __DIR__.'/LectioScrape.class.php';

//Creates a new LectioScrape object with the date passed from $_GET as input parameter
$schedule = new LectioScrape($_GET['startDate'].'T01:00:00');

//Sends the schedule to the database
$schedule->sendToDatabase();
```

```php
//Redirects to the front page
$redirect_uri = 'http://' . $_SERVER['HTTP_HOST'] . '/LectioScrape/index.html';
header('Location: ' . filter_var($redirect_uri, FILTER_SANITIZE_URL));

?>
```

fullcalendarFeed.php

```php
<?php
/*This script serves a Full calendar json feed*/

//Includes the LectioScrape class if not already included
require_once __DIR__.'/LectioScrape.class.php';

//Gets the start date passed with GET
$start = $_GET['start'];

//Creates a new LectioScrape object
$lectioSchedule = new LectioScrape($start);

//Sets the content type
header('Content-Type: application/json');

//Echoes the scheduleFullcalendar array as a JSON string
echo json_encode($lectioSchedule->scheduleFullcalendar);

?>
```