

Bødekaske

Formål

Formålet med projektet var at kunne lave en hjemmeside, hvor en administrativ bruger kan give forskellige bøder til spillerne på et bestemt hold, hvis de har gjort noget forkert. På vores hjemmeside har man mulighed for at oprette en specifik bøde, og tilgive bøden til den spiller som skal have den. Vi valgte at lave en front-end del til begge projekter, for bedre at kunne fremvise funktionaliteterne for de forskellige API's.

API Integration

Vi bruger 3 forskellige API'er. To af de API'er er integreret til vores hjemmeside, og Machine Learnings API har vi lavet på en separat hjemmeside. Det har vi gjort fordi ML-API'en ikke passede ind i vores hovedprojekt. De to andre API'er som er integreret til hjemmesiden viser direkte livescore for alle kampene der bliver spillet. API nummer to viser alle kommende fodboldkampene i hele verden. Vores Machine Learning API analysere billedet som vi uploader til vores hjemmeside. Til det projekt har vi brugt Google Visions API.

Hvordan virker vores API?

Vi har brugt <https://live-score-api.com/>'s officielle dokumentation til at implementere deres API's i vores hjemmeside.

```
/**
 * Gets the live scores
 *
 * @param array $params filter parameters
 * @return array with live scores data
 */
public function getLivescores($params = [])
{
    $url = $this->_buildUrl('scores/live.json', $params);
    $data = $this->_makeRequest($url);
    return $data['match'];
}

/**
 * Gets upcoming games
 *
 * @param array $params filter parameters
 * @return array with live scores data
 */
public function getUpcomingGames($params = [])
{
    $url = $this->_buildUrl('fixtures/matches.json', $params);
    $data = $this->_makeRequest($url);
    return $data['fixtures'];
}
```

For at vi fremvise kommende kampene og Live Scores har vi lavet 2 forskellige public functions. Disse to funktioner kalder vi i deres separate filer (upcoming.php og livescore.php).

```
<?php
ini_set('display_errors', 'Off');
error_reporting(E_ALL);

require_once 'LiveScoreApi.class.php';

$LiveScoreApi = new LiveScoreApi($API_KEY, $API_SECRET, $servername, $username, $password, $dbname);
$fixtures = $LiveScoreApi->getUpcomingGames();

include 'includes/left.php'; ?>

<?php foreach ($fixtures as $_fixtures) { ?>
    <div class="card bg-info">
        <div class="row text-dark">
            <div class="col-md-2 time-box">
                <?= $_fixtures['time'] ?>
            </div>
            <div class="col-md-4 team-name">
                <?= $_fixtures['home_name'] ?>
            </div>
            <div class="col-md-2">
                v
            </div>
            <div class="col-md-4 team-name">
                <?= $_fixtures['away_name'] ?>
            </div>
        </div>
    </div>
</body>
```

Her kalder vi på kommende kampene. Vi require først vores LiveScore class som indeholder vores GetUpcomingFunctions. Siden det er en public function har vi mulighed for at kalde på den i forskellige filer. Vi laver en foreach løkke for at kunne udskrive data som API indeholder (time, home_name og away_name).

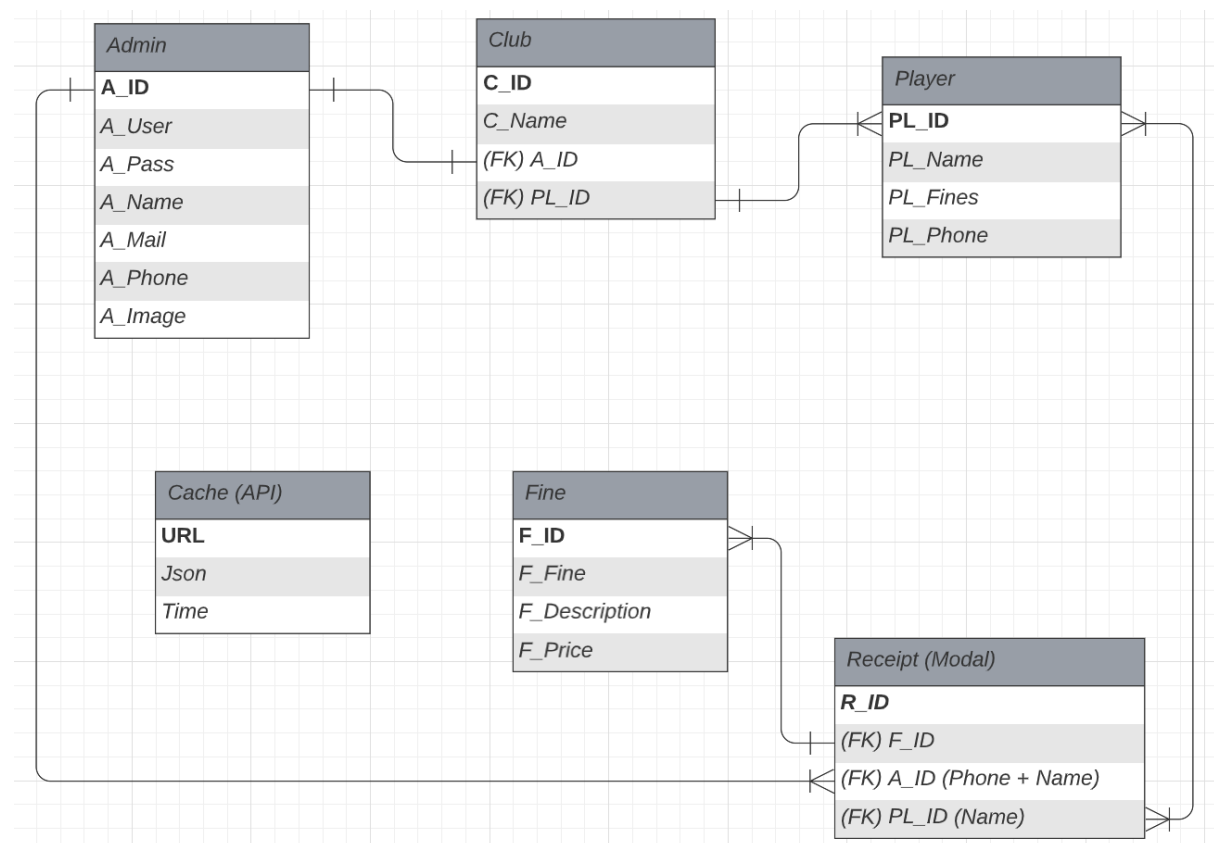
Posten Bødekasse				Bøder	Bødetyper	Spillere	LiveScore	Kommende Kampe
14:20:00	Al Rayyan SC	v	Al Ahli Doha SC					
15:30:00	Klub 04	v	Honka Akademia					
16:00:00	Al Mokawloon Al Arab	v	Pharco FC					

Sådan ser kommende kampene ud på vores hjemmeside.

Samme princip gælder for LiveScore siden

GO-JEK Liga 1	08:30			
FT	Persebaya Surabaya	1 - 0	PSIS	
GO-JEK Liga 1	08:30			
FT	Persib Bandung	2 - 3	Bali United Pusam	

E/R Diagram

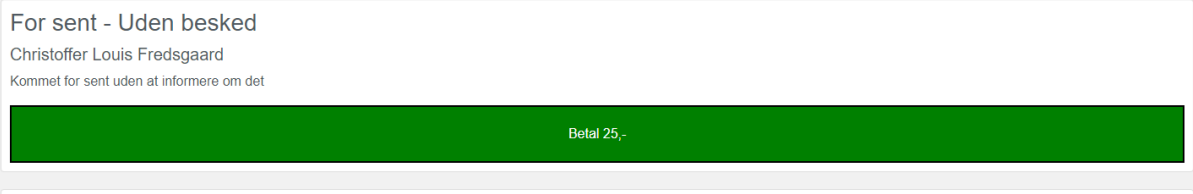


Vores E/R diagram består af 5 hoved tabeller og 1 tabel for sig selv som bliver brugt som cache for vores API del på hjemmesiden. Inden vi skulle lave vores database, startede vi selvfølgelig med at lave et E/R diagram, for at få et godt overblik over hvordan databasen skulle sættes op. Måden vi lavede diagrammet på, var med fremgangsmåden som beskrevet herunder:

1. Vi har en klub (Club), som skal være styret af en admin (Admin)
2. Derudover skal der selvfølgelig også være nogle spillere i denne klub, som kan få en bøde (Player)
3. Så skal der også være nogle bøder som kan gives til de forskellige spillere. Bøderne bliver lavet af Admin og lagt ind i (Fine), og når bøden skal gives bliver der oprettet en (Receipt), som bruger ID'erne fra de forskellige tabellerne til at definere en bøde.

Nu kommer vi så til den enkelte tabel. Den enkelte tabel, (Cache), bliver brugt til vores API som får live score og de kommende kampe. Det som tabellen gør, er at gemme det nyeste kørte kald, og bruger det samme kald igen til de kommende kald, hvis det er det samme som søges efter. Det bliver gjort for at mindste responstiden, og mængden af kald som bliver lavet. Den bliver resat ved et hard refresh af siden, eller hvis siden bliver lukket ned og åbnet igen.

Når en bøde bliver tildelt en spiller, er den data som bliver fremvist, lavet med et inner join. Inner joins kombinerer data fra to tabeller, når der er matchende værdier i et felt, der er fælles for begge tabeller. I dette tilfælde er det, F_ID, A_ID og PL_ID.



For sent - Uden besked
Christoffer Louis Fredsgaard
Kommet for sent uden at informere om det

Betalt 25,-

Hvorfor bruge et ERD?

ER-diagrammer angiver, hvilke data vi vil gemme: enhederne og deres attributter. De viser også, hvordan enheder forholder sig til andre enheder. En anden fordel ved ERD'er er, at de repræsenterer dataene på en grafisk måde. Dette gør det overskueligt og nemt forståeligt for folk udefra, samt når man skal opsætte sin database.

Hovedprojektet - Bødekasse

For at kunne forstå hvad vores projekt går ud på, skal man vide hvad en bødekasse er, og hvad vores bødekasse indebærer.

En bødekasse, er en online kasse, som indsamler penge når visse regler bliver brudt. Denne bødekasse er sat op således, at det kun er admin som kan uddele bøder, oprette spillere og oprette bøderne som kan uddeles.

Opret Spiller <input type="text" value="Fuldt navn"/> <input type="text" value="Telefonnummer"/> <input type="button" value="Opret Spiller"/>	Opret bøde <input type="text" value="Bøde Navn"/> <input type="text" value="Bøde Beskrivelse"/> <input type="text" value="Bøde Pris"/> <input type="button" value="Opret Bøde"/>	Giv bøde <input type="text" value="Mikkel Hansen Skræp"/> <input type="text" value="Rødt Kort - Brok"/> <input type="button" value="Giv bøde"/>
---	---	---

Spillerne, når de får tildelt en bøde, kan gå ind på forsiden af vores hjemmeside, hvor der kommer et lille blogpost, og betale bøden derigennem. Siden er primært egnet til telefon, og derfor kommer der et 'telefon-link' frem, som redirecter til en mobilepay bødekasse, hvor de kun skal skrive beløbet ind og så swipe, når det er der bliver trykket på betal knappen.

Posten Bødekasse
For sent - Uden besked Christoffer Louis Fredsgaard Kommet for sent uden at informere om det <input type="text" value="Betal 25,-"/>
For sent - Besked skrevet Mikkel Hansen Skræp Spiller kommet for sent, men har informeret om det <input type="text" value="Betal 15,-"/>
For sent - Uden besked David Szelmeczi Kommet for sent uden at informere om det <input type="text" value="Betal 25,-"/>

Derudover, så skal bøderne som bliver oprettet og bøderne som bliver givet, holdes separat. Måden vi har løst det på, er at en admin kan på admin.php siden, oprette en spiller, oprette og en bøde og så kan han give en bøde. Hver oprettelse går ind på hver sin del af databasen. Når bøden skal gives, bliver den sat ind på Receipt, på dansk: kvittering. Der bliver redirected til forsiden, som looper igennem tabellen Receipt når siden bliver loadet.

De Query's vi har lavet, bliver alle sammen kaldt i en stored procedure, for at gøre koden mere overskuelig. For at lave et Call til en stored procedure i MySQL, bruger man 'Call Procedure_Name'

Vores SQL Query's ser ud således:

Insert

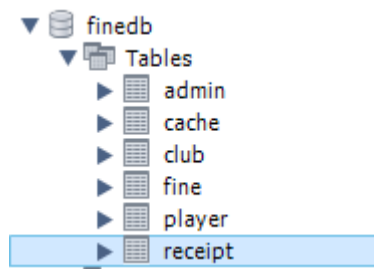
```
INSERT INTO finedb.player (PL_Name, PL_Phone)  
VALUES ('$name', $number)"
```

Select

```
SELECT  
    Receipt.R_ID as 'ID',  
    Admin.A_Name as 'Admin Name',  
    Admin.A_Phone as 'Admin Phone',  
    Fine.F_Fine as 'Fine',  
    Fine.F_Price as 'Price',  
    Fine.F_Description as 'Description',  
    Player.PL_Name as 'Player Name'  
  
FROM Receipt  
    INNER JOIN Fine  
    ON Receipt.F_ID=Fine.F_ID  
    INNER JOIN Admin  
    ON Receipt.A_ID=Admin.A_ID  
    INNER JOIN Player  
    ON Receipt.PL_ID=Player.PL_ID;
```

Databasen

Databasen er opsat ud fra vores ERD, som er opsat med det vi mener er de rigtige relationer og tabeller i forhold til hvad projektet skal indeholde og skal kunne. Som beskrevet tidligere i rapporten, så består databasen af 5 hovedtabeller og 1 enkelt tabel for sig selv, til cache fra vores API.



I oprettelsesprocessen, er der opsat primary keys på ID'erne i tabellerne. Samt foreign keys på de kolonner / felter som skal joines med andre tabeller i databasen. Vores receipt tabel er udelukkende lavet af ID'er af dataen fra de forskellige tabeller, de bliver derefter læselige / forståelige ved hjælp af joins, hvor de rigtige rækker bliver selected.

	R_ID	A_ID	F_ID	PL_ID
▶	24	2	114	1
	25	2	115	2
	26	2	116	1
	27	2	115	5
*	NULL	NULL	NULL	NULL

Underprojekt - Google Vision API

Vores Google Vision API projekt kunne ikke implementeres i vores hovedprojekt, derfor har vi valgt at lave et separat front-end til den del. Projektet handler om at Google Vision skal beskrive, hvad billedet indeholder. Hjemmesiden har forskellige menuer for at se flere forskellige informationer om billedet. Hjemmesiden kan udregne via Machine Learning, fortælle hvor mange % den er sikker på at den ser X. Man kan upload alle slags billeder man har lyst til, og den tager imod både JPG og PNG filer, så der er fri lege i hvilket af de to formater man vil uploade et billede.

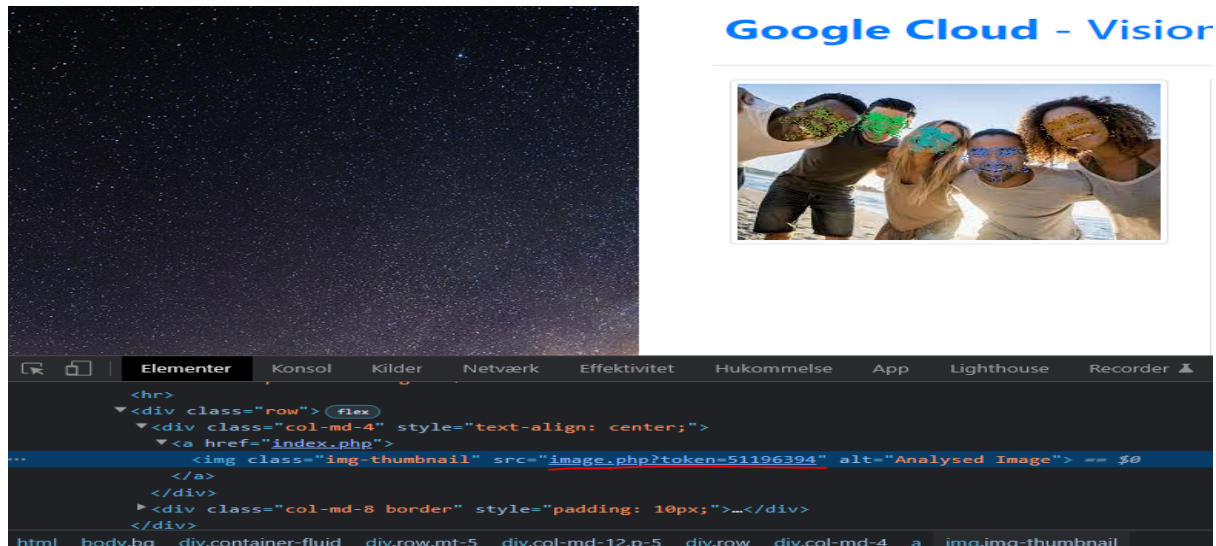
Kodestykke

```
$familyPhotoResource = fopen($_FILES['image']['tmp_name'], 'r');

$image = $vision->image(
    $familyPhotoResource,
    [
        'FACE_DETECTION',
        'LABEL_DETECTION',
        'IMAGE_PROPERTIES',
        'WEB_DETECTION',
        'LANDMARK_DETECTION',
        'LOGO_DETECTION'
    ]
);
$result = $vision->annotate($image);
```

I denne del af koden, er der et billede som lige er blevet uploadet. Vi bruger den integreret del af API'en til at fortælle den hvad den skal kigge efter, og hvis den eksempelvis finder et ansigt på siden vil den komme med data for hvad den tror, at den ser.

```
<div class="row">
  <div class="col-md-4" style="text-align: center;">
    <a href="index.php">
      " alt="Analysed Image">
    </a>
  </div>
</div>
```



På de to billeder herover, ser vi den process som sker, når et billede bliver analyseret på vores side. Hvis billede som skal analyseres, IKKE, har indeholder ansigt, så laver et random genereret navn.jpg, men hvis billedet indeholder et ansigt, så skal den i stedet lave navnet om til image.php?token=0000000. I stedet for nullerne får den en random generet int, som set lige herunder.

```
27 if ($result) {
28     $imagetoken = random_int(1111111, 999999999); //Generet en random INT som bliver billedets jp
29     move_uploaded_file($_FILES['image']['tmp_name'], __DIR__ . '/feed/' . $imagetoken . ".jpg");
30 } else {
31     header("location: index.php");
32     die();
33 }
```

```
session_start();
/*
Linje 8-10: Modtager image token fra check.php, hvis billedet er et ansigt, kører den sessionen 'faces' hvor den generer en src="image.php?token=0000000000"
Linje 12+: Hvis der er et ansigt, kører den foreach loopet på linje18. Hvor der defineres at den skal løbe x gange.
*/
$imagetoken = $_GET['token'];
$faces = $_SESSION['faces'][$imagetoken];
$image = imagecreatefromjpeg("feed/" . $imagetoken . ".jpg");

foreach ($faces as $key => $face) {
    $face = json_decode($face);
    $faceColorR = $_SESSION['faces'][$key][0];
    $faceColorG = $_SESSION['faces'][$key][1];
    $faceColorB = $_SESSION['faces'][$key][2];

    foreach ($face as $part) {
        for ($j = 0; $j < 5; $j++) {
            imagepixel($image, round($part->position->x), round($part->position->y), imagecolorallocate($image, $faceColorR, $faceColorG, $faceColorB));
            imagepixel($image, round($part->position->x - random_int(1, 3)), round($part->position->y + random_int(1, 3)), imagecolorallocate($image, $faceColorR, $faceColorG, $faceColorB));
            imagepixel($image, round($part->position->x + random_int(1, 3)), round($part->position->y - random_int(1, 3)), imagecolorallocate($image, $faceColorR, $faceColorG, $faceColorB));
            imagepixel($image, round($part->position->x - random_int(1, 3)), round($part->position->y - random_int(1, 3)), imagecolorallocate($image, $faceColorR, $faceColorG, $faceColorB));
            imagepixel($image, round($part->position->x + random_int(1, 3)), round($part->position->y + random_int(1, 3)), imagecolorallocate($image, $faceColorR, $faceColorG, $faceColorB));
        }
    }
}
```

I den her kodestykke kan man se at vores program sætter forskellige farvet firkanter på ansigtet. Det har vi valgt at tilføje, for bedre at visualisere, hvad API'en kigger på når den vil finde ud af hvad deres ansigtet udtrykker. På linje 18 i image.php hvor vi bruger echo er det til at fremvise tallet 1. Hvis der er flere ansigter vil der står Face 2, Face 3 osv..

```
<div class="row">
  <div class="col-6">
    <strong>Happy</strong>
  </div>
  <div class="col-6">
    <strong><?php echo $face->info()['joyLikelihood'] ?></strong>
  </div>
</div>
<div class="row">
  <div class="col-6">
    <strong>Sorrow</strong>
  </div>
  <div class="col-6">
    <strong><?php echo $face->info()['sorrowLikelihood'] ?></strong>
  </div>
</div>
<div class="row">
  <div class="col-6">
    <strong>Angry</strong>
  </div>
  <div class="col-6">
    <strong><?php echo $face->info()['angerLikelihood'] ?></strong>
  </div>
</div>
```

Her i vores HTML printer vi Google Visions API informationer ude. joyLikeliHood og de andre er indbygget i API'ens hukommelse.

1. Face 1

Happy	VERY_LIKELY
Sorrow	VERY_UNLIKELY
Angry	VERY_UNLIKELY
Surprised	VERY_UNLIKELY
Blurred	VERY_UNLIKELY
Headwear	VERY_UNLIKELY

Det ses sådan her ud på hjemmesiden.

API - Doc & Key's

Live Score API Officielle implemtering:

<https://live-score-api.com/documentation/tutorials>

Google Vision: <https://cloud.google.com/vision>

Live score: https://live-score-api.com/documentation/reference/6/getting_livescores

Kommende fodboldkampene:

<https://live-score-api.com/documentation/reference/13/getting-scheduled-games>

LiveScore:

API_KEY =

WsGzwgwinKE8u3Nx

API_SECRET =

Fp5Nm0KFRKPFVuoRW5qtlXMIHqRcuEK9

POSTMAN CALL =

<https://livescore-api.com/api-client/fixtures/matches.json?&key=WsGzwgwinKE8u3Nx&secret=Fp5Nm0KFRKPFVuoRW5qtlXMIHqRcuEK9>

Google Vision:

"private_key":

"-----BEGIN PRIVATE

KEY-----\nMIIEvIBADANBgkqhkiG9w0BAQEFAASCBKkwggSIAGeAAoIBAQDdaTeOBKJhZFKq'n7BKzCQIY32hrZirNvNut8Li4E/wNA9opULAXQkS0yLtGEfeEOcqXxj3A8QnRkKqC'nBya67D+DDhCASdV2VjyFfnnw1nHSHYi5vO+wbDf6gYTiVBOcWKliJg6jIA2J/R7gn3lOozhsOaXUKV/O6ut2tNfrKmRd6NikEqpX0CRssuS2LaxT7yuyED22xRAZcVVgr'nO8AE4Gkp4nMOegcY9PISirJ138FmiYeGSz+0ZJBaTyklsvhh2A508vojKF0AojE/nioPni/F5RrOjVqV5yePWYkTVzFG4yGhK5X4p4g3kXpEdcOKU/MfhHhKGv2USLiCwFneDhDisUtAgMBAAECggEAHCu/fEvdpzufAw1PYcUzJzWDSqrwpkWxw8NdfOx6BOPpInFZmlFgiB+miR31v43LOHseDwyYRBpg7civfC90S13jFjaxc52DIFxeuC5IA60jRYnySFcLKM0szfVOQ4KBroGQwcd8EOmyVhR3fVgsREsQxyq3hqLscYF33x/QNjSsmMRnNMQUfJGkCEqWREbwWuZc7wx6/Brx+XFP5oRm6jTQNHdu68+RF9I0y8Y55MYWMJEIn9RQvGdT6L6Z/Z4v7ZRxsX1/6UQ3xxYZaSjy+vYjeJYvDopoQ1YxAnxvDzs87hzhHln/gPSU5D8zo6cYpokiCPy6DMAjxF0mV8+dLLk6uD+wQKBgQDzUGP6FtuWKRmJyVAnlnZNR9MQc2QBxdKImw5wRGNNg9bgYZ2xGb2PhfB7D4inlrGWJN3loXFnpKfDMDhr'n70GazzNPrUpW67kNnZJ8fZHVSTX4d4NnkgNwNqozk/0seQ2TroXgNludQAKoraAcxlnOPIHbHJLFUEX+X3ZOyK9/5a+oQKBgQD09HpkxDq3xiZhOYioOaQ0zEvf+YIT1wccInDmG0+n9tMK1ebV4JfzHcR0YLM0tEWFq97sCFM/c1rZIRRsdbu+UbruMHCCbJ2OrNnif5XsEaNTBqsSMtyuT9JpFTGOFFmeAtGP4wsoQQprJbg0B8xW8gZH3/Lbyrfn7EUUnuullqHa3DQKBgQCdqP+/c8RB39VIYRABFsLhhD1qY4qv+aWafmKAE4ny9Dlb2unlnkFimL+k3gHspibYZXcrbpsvQSEMPQUmuFtBp4c0b2ewLup6khe//ntZLOS MgbKyD'n900JXoQxdnUTiM8I9rlmE9Ods1cpwHSEVYzhHUBKCrGtOUsk0b7HYrBpQQKBgQCqInJV572OM2NPD1VCtfc3LG6WYFs2G9LbHQqSRuJgQKQWJvNjXZfWBYjA3gAh6a7ojInPY25/KGTwC7aMUL53P3SwwKqUlayONAOtBPYvaSfq1WRnXujviUrdkbQ/HEujRLInzr3ocdDIO9FXdEtEJtA4oUwazyBWYvHYvUgEPJ8wxQKBgQDARRoYGObuQlhg91H1n4mf+2S16k2QO9gneSE6sxc6yIV+dEKFrpow3WBLfBYMuy940ouQSW1x5pFebZBEInnv2lrEONG3/lcXlrSkkws2oQchuJIEQuHY4JNoGmAwamMyMtuPefzluLnUCjsTKhInuekO7KSGBM9aKNDYWgqMFwvBA==\n-----END PRIVATE KEY-----\n",