Pembuatan Web Application untuk Mendukung Pelayanan Asisten Tutor di Universitas Kristen Petra

Kevin Fidelis, Adi Wibowo, Justinus Andjarwirawan
Program Studi Teknik Informatika, Fakultas Teknologi Industri, Universitas Kristen Petra
Jln. Siwalankerto 121 – 131 Surabaya 60236
Telp. (031)-2983455, Fax. (031)-8417658
mmc.kevinf@gmail.com, adiw@petra.ac.id, justin@petra.ac.id

ABSTRAK

Asisten tutor adalah sebutan bagi mahasiswa yang melayani mahasiswa baru dalam kelompok-kelompok kecil, dalam sebuah program pembinaan mahasiswa baru yang dikenal sebagai *Ethics Enrichment*. Pada saat ini, setiap alat pembantu yang dipakai asisten tutor dicetak dengan menggunakan kertas. Dibandingkan dengan jumlah asisten tutor, pencetakan dengan kertas ini adalah pemborosan dan tidak terlalu praktis, khususnya bila melihat perkembangan teknologi yang sebenarnya dapat memudahkan pelayanan asisten tutor.

Web application dibuat dengan tujuan untuk membantu pelayanan asisten tutor melalui sebuah sistem yang dapat diakses di berbagai perangkat dan dapat memenuhi kebutuhan-kebutuhan dari asisten tutor dalam melakukan pelayanan. Beberapa fitur yang terdapat di dalam *web application* antara lain akses materi, penilaian, absensi online, dan interaksi dengan mahasiswa baru seperti perjanjian pertemuan dan sharing. *Web application* juga menggunakan teknologi JSON Web Token untuk melakukan *login*, dan ada juga notifikasi melalui *email* jika ada penambahan konten tertentu di dalam *web application*.

Pengujian yang dilakukan menghasilkan kesimpulan bahwa *web application* yang dibuat dapat diakses dan dijalankan dengan baik pada berbagai jenis perangkat. Survei yang dilakukan juga menunjukkan bahwa *web application* menjawab kebutuhan di dalam menyediakan sistem yang mendukung pelayanan asisten tutor di Universitas Kristen Petra, dengan 59% responden menyatakan bahwa web application bermanfaat dan 53% responden menyatakan bahwa web application secara keseluruhan baik.

Kata Kunci: Aplikasi Web, Kerangka Kerja, JavaScript, PHP, AngularJS, Laravel, *JSON Web Token*, *REST*

ABSTRACT

Tutor assistant is a term for students who serve new students in small groups, in a coaching program for new students known as Ethics Enrichment. Currently, every support tools that have been used by tutor assistants were printed. In comparison with the number of tutor assistants, the printing of support tools is a waste and not really practical, especially when compared to the advancement of technologies which actually can ease the service of tutor assistants.

Web application was made with the purpose of helping tutor assistant service by providing a system that can be accessed across devices and can fulfill the needs of tutor assistants while doing their services. Some features in web application include material access, grade assignment, online attendance form and interactions with new students via sharing or meeting agreement form. Web application also use JSON Web Token to provide login service and email notifications if any content addition occur in web application.

After doing some testing, it can be concluded that web application can be accessed nicely across devices. Survey also shows that web application answers the needs in providing a system that support the service of tutor assistants in Petra Christian University, with 59% of respondents agree that the web application is useful and 53% of respondents agree that web application as a whole has been good.

Keywords: *Web Application, Framework, JavaScript, PHP, AngularJS, Laravel, JSON Web Token, REST*

1. PENDAHULUAN

Asisten Tutor (Astor) adalah sebutan bagi mahasiswa yang melayani mahasiswa baru dalam kelompok-kelompok kecil, dalam sebuah program pembinaan mahasiwa baru yang dikenal sebagai *Ethics Enrichment* (EE). Program *Ethics Enrichment* dinaungi oleh Tim Petra Sinergi, dan bertujuan untuk memuridkan mahasiswa baru, sehingga para mahasiswa baru tersebut dapat mengerti mengenai makna dan tujuan hidupnya di hadapan Tuhan selama berkuliah di Universitas Kristen Petra.

Pada setiap semester gasal, asisten tutor akan mengikuti penjelasan materi pada saat *briefing*. *Briefing* dilaksanakan pada pukul 8.30 hingga pukul 10.00 setiap hari Jumat. Setiap Asisten Tutor diberi sebuah map yang berisi informasi-informasi, alat pembantu yang diperlukan dalam menyampaikan materi, serta absensi ketika memasuki ruangan *briefing*. Tentu saja, informasi-informasi serta absensi dicetak dengan kertas. Pencetakan informasi dan absensi dengan kertas bukanlah suatu langkah yang baik, mempertimbangkan kampanye "*Go Green*" yang dikampanyekan oleh *Ministerium Movement* yang bekerja sama dengan pihak universitas, dan jumlah kertas yang perlu dicetak sesuai dengan jumlah Asisten Tutor yang lebih dari 100 orang. Selain itu, dengan perkembangan teknologi yang semakin canggih seiring perkembangan zaman, diperlukan sesuatu yang lebih praktis dan efektif bagi Asisten Tutor dalam menjalankan pelayanannya.

Melihat realita ini, maka perlu dibuat sebuah solusi yang dapat membantu pelayanan Asisten Tutor dalam melihat bahan pertemuan, melakukan absensi kehadiran, serta melakukan penilaian terhadap mahasiswa baru. Solusi tersebut harus dapat diakses dengan mudah melalui *smartphone* maupun *tablet*, sehingga benar-benar dapat digunakan dengan praktis oleh para Asisten Tutor. Solusi tersebut juga haruslah sebuah solusi yang dapat diakses dengan menggunakan perangkat apapun, karena perlu disadari bahwa tidak semua Asisten Tutor menggunakan perangkat dengan sistem operasi yang sama ataupun dapat

langsung terhubung dengan internet melalui *smartphone* yang mereka miliki. Karena itu, dengan mempertimbangkan akses melalui perangkat lintas *platform*, solusi tersebut dapat dihadirkan dengan membuat sebuah *web application* yang *mobile-friendly*, sehingga dapat diakses dengan mudah melalui semua *smartphone* dan dapat pula diakses melalui komputer.

2. LANDASAN TEORI

2.1 Web Application

Web application yang juga bisa disebut sebagai *rich internet application*, memiliki fungsi yang lebih dari sekedar menampilkan konten kepada pengguna. *Web application* dikembangkan dengan maksud untuk memungkinkan interaksi pengguna, dan bukan hanya menampilkan konten. [9]

Dalam halnya dengan pengembangan aplikasi *mobile*, menggunakan web application untuk membuat aplikasi *cross-platform* memiliki berbagai manfaat [5]:

• Tidak ada biaya yang harus dikeluarkan, karena tersedianya banyak *framework web application* yang dapat di-*download* dengan gratis.

• Semua sistem operasi *mobile* dapat membuka web application, hanya dengan syarat adanya *browser* yang dapat dipakai untuk membuka internet.

• Tidak perlu melakukan instalasi aplikasi pada perangkat yang ingin membuka *web application*, sehingga tidak perlu menggunakan ruang yang ada pada perangkat untuk menampung aplikasi yang bersangkutan.

• Perkembangan teknologi bahasa pemrograman web mengalami perkembangan terus-menerus dan internet akan selalu menjadi landasan bagi perkembangan teknologi masa depan, sehingga melakukan pengembangan aplikasi dengan *web application* bersifat *future-proof*.

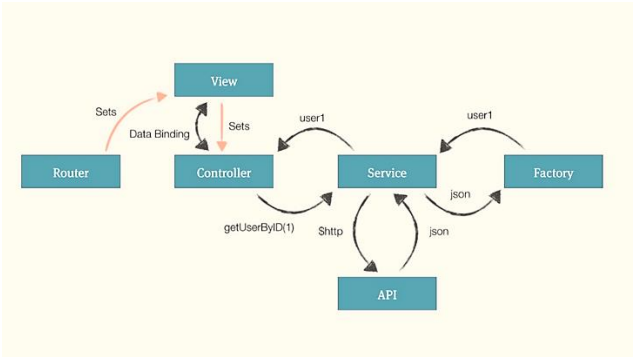
• Pengguna hanya perlu mengetahui alamat dari *web application* dan pengguna akan selalu dapat melihat perkembangan terbaru dari *web application* tersebut.

• Ada banyak tool yang dapat digunakan untuk membangun dan mengubah *web application*.

2.2 AngularJS

AngularJS adalah sebuah *framework* struktural yang ditujukan untuk pengembangan *web application* yang dinamis. AngularJS memungkinkan penggunaan HTML sebagai bahasa *template* dan memperluas *syntax* HTML untuk mengekspresikan komponen aplikasi yang dibangun dengan jelas dan ringkas. AngularJS berupaya untuk meminimalisir ketidakcocokan antara HTML yang dokumen sentris dengan kebutuhan dalam pengembangan aplikasi dengan membuat susunan HTML yang baru. AngularJS menangani semua hubungan antara DOM dan AJAX dengan menyusunnya sebagai struktur yang didefinisikan dengan baik. AngularJS juga menyederhanakan pengembangan aplikasi dengan menyediakan tingkat abstraksi yang lebih tinggi bagi pengembang. [3]

Ada beberapa konsep dasar yang dipakai dalam pengembangan aplikasi AngularJS, antara lain adalah : *client-side templates, model view controller, data binding, dependency injection, directives*. [4]

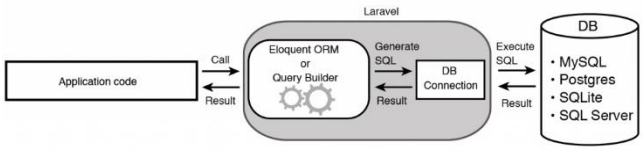


Gambar 1. Cara Kerja AngularJS [6]

2.3 Laravel

Laravel adalah sebuah *framework* PHP yang ditulis Taylor Otwell, dengan tujuan untuk membantu pengembang dalam membuat *web application* dengan *syntax* yang sederhana, elegan, ekspresif, dan menyenangkan. Pembuatan *web application* dengan Laravel dapat menghasilkan *web application* dengan *syntax* yang ekspresif dan elegan. "Dengan Laravel, tugas-tugas umum developer dapat dikurangi pada sebagian besar proyek-proyek web seperti *routing*, *session* dan *caching*." [7]

Laravel sebagai *framework* berbasis PHP umumnya digunakan pengembang bersama dengan sebuah *database* seperti MySQL atau PostgreSQL. Karena itu, sebelum melakukan pengembangan *web application* dengan Laravel, pengembang perlu melakukan instalasi PHP dengan versi 5.4 atau lebih baru dan *database* yang didukung oleh Laravel (MySQL, PostgreSQL, SQLite, dan Microsoft SQL Server). [2]



Gambar 2. Laravel dan *Database* [10]

2.4 REST

REST (*REpresentational State Transfer*) adalah suatu gaya arsitektur dan suatu pendekatan untuk melakukan komunikasi yang sering dipakai dalam pengembangan *web service*. Pemakaian REST lebih sering dipilih dibandingkan SOAP (*Simple Object Access Protocol*) karena REST tidak membutuhkan terlalu banyak *bandwidth* seperti halnya SOAP. Arsitektur yang bersifat terpisah dari REST, disertai pemakaian data komunikasi yang lebih ringan antara produsen dan konsumen membuat REST menjadi gaya pengembangan yang populer dalam membangun API berbasis *cloud*. *Web service* yang memakai arsitektur REST disebut sebagai RESTful API atau REST API. [8]

2.5 JSON Web Token

JSON Web Token adalah sebuah standar terbuka yang mendefinisikan cara yang tersusun dan mandiri untuk mengamankan informasi pengiriman antara beberapa pihak sebagai sebuah obyek JSON. Informasi ini dapat diverifikasi dan dipercayai karena ditandatangani secara *digital*. JSON Web Token memiliki kegunaan dalam hal melakukan autentikasi dan pertukaran informasi. JSON Web Token terdiri atas tiga bagian yang dipisahkan dengan tanda titik (.), yaitu :

- *Header* : terdiri atas dua bagian, yaitu tipe token dan algoritma *hashing* yang digunakan.
- *Payload* : mengandung *claim*, yaitu pernyataan mengenai sebuah entitas dan metadata tambahan.
- *Signature* : dipakai untuk memverifikasi bahwa pengirim JWT adalah seperti yang terkandung dalam token dan untuk memastikan bahwa pesan tidak diubah selama pemakaian JSON Web Token.

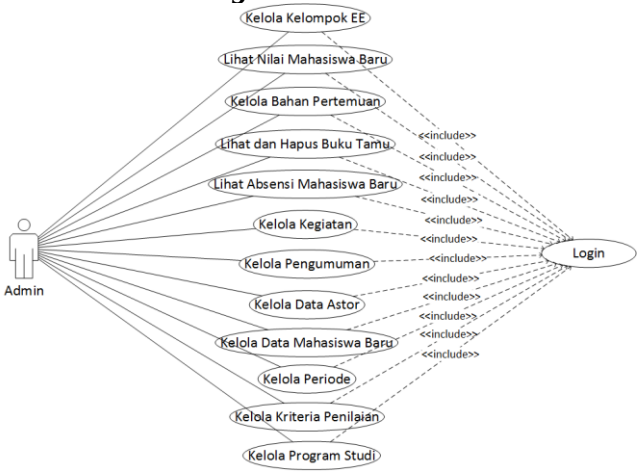
Sehingga, struktur JSON Web Token umumnya terlihat seperti "xxxxx.yyyyy.zzzzz", di mana "xxxx" adalah *header*, "yyyyy" adalah *payload*, dan "zzzzz" adalah *signature*. [1]

3. DESAIN SISTEM

3.1 Desain Sistem Aplikasi

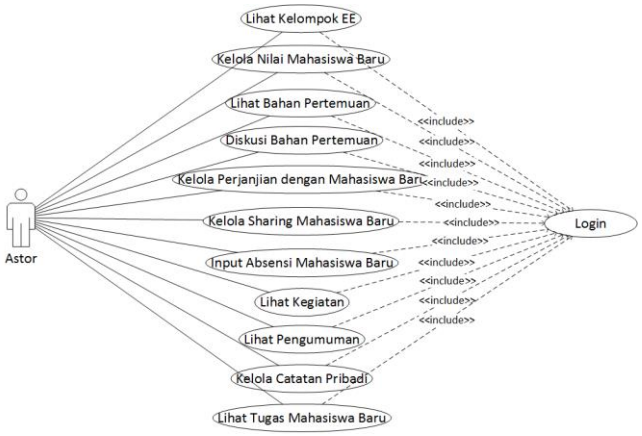
Dalam perancangannya, *web application* dibagi menjadi 3 bagian menurut perannya, yaitu administrator, asisten tutor, dan mahasiswa baru. Ketiga peran ini masing-masing memiliki hak akses yang berbeda-beda, karena itu *web application* juga memberikan fitur yang berbeda-beda bagi setiap pengguna sesuai perannya masing-masing. Data-data yang diakses maupun dikelola melalui *web application* disimpan dalam *database* MySQL. Sudah dijelaskan sebelumnya bahwa *web application* juga bersifat *mobile-friendly*, karena itu *web application* juga dapat diakses melalui *gadget* yang dimiliki pengguna dengan nyaman.

3.2 Use Case Diagram



Gambar 3. *Use Case Diagram* Administrator

Gambar 3 menunjukkan *use case diagram* bagi sisi administrator. Administrator adalah pihak yang paling bertanggung jawab dalam mengelola konten dari *web application*, karena itu banyak interaksi yang dapat dilakukan oleh administrator di dalam sistem, termasuk beberapa interaksi yang hanya dapat dilakukan oleh administrator saja. Contoh interaksi yang hanya dapat dilakukan oleh administrator mencakup mengelola kelompok *Ethics Enrichment*, mengelola bahan pertemuan, mengelola kegiatan, dan mengelola pengumuman.



Gambar 4. *Use Case Diagram* Asisten Tutor

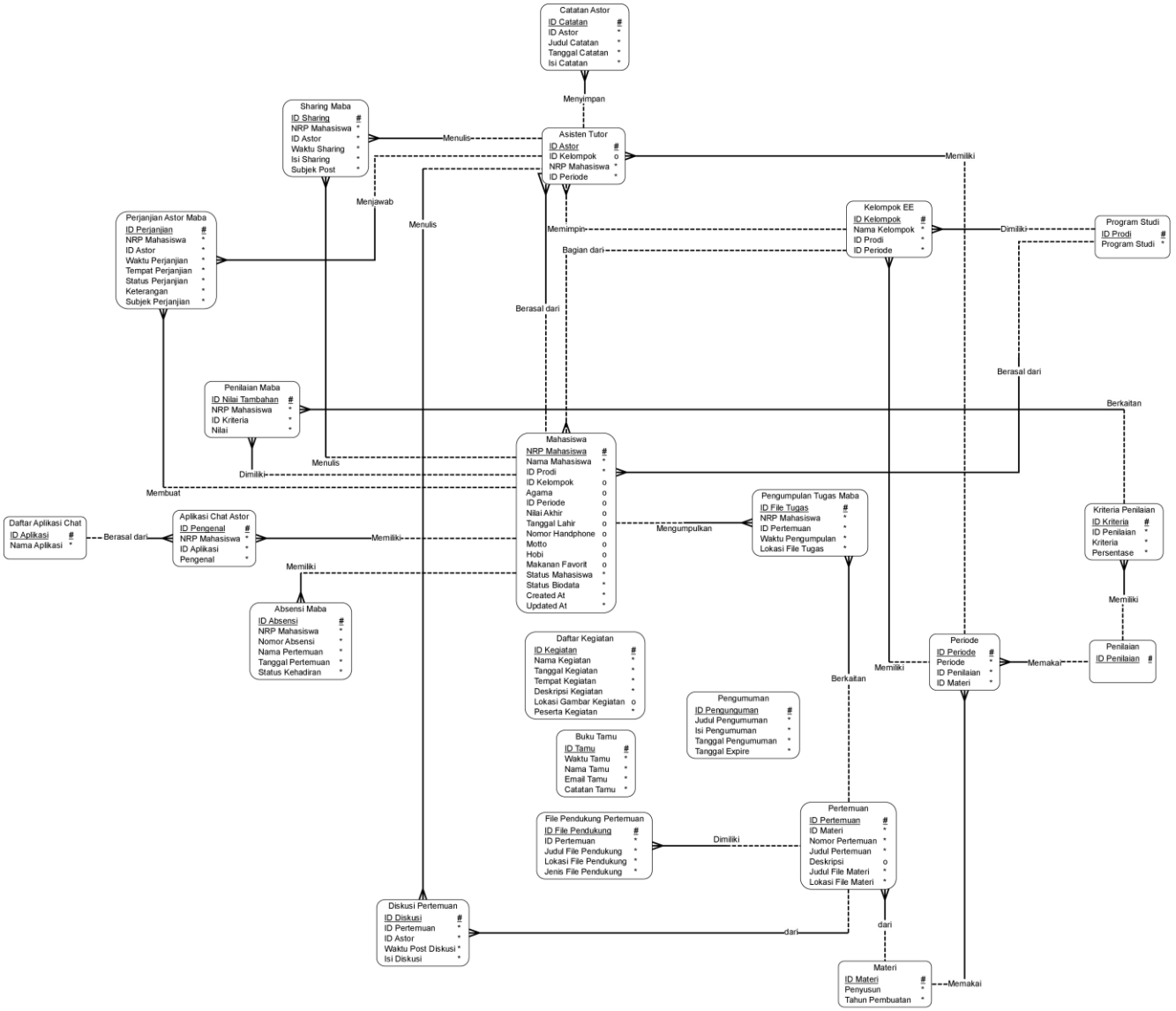
Gambar 4 menunjukkan *use case diagram* bagi sisi asisten tutor. Asisten tutor adalah pihak kedua yang paling bertanggung jawab dalam sistem, dikarenakan adanya beberapa interaksi yang juga hanya dapat dilakukan oleh asisten tutor dalam hubungannya dengan konten dari sistem. Beberapa interaksi yang hanya dapat dilakukan oleh asisten tutor di antaranya mengelola nilai mahasiswa baru, menjawab perjanjian dengan mahasiswa baru, input absensi mahasiswa baru, dan mengelola catatan pribadi.

Gambar 5. *Use Case Diagram* Mahasiswa Baru dan Tamu

Gambar 5 menunjukkan *use case diagram* bagi sisi mahasiswa baru dan tamu. Interaksi-interaksi yang dapat dilakukan oleh mahasiswa baru dan tamu tidaklah sebanyak administrator maupun asisten tutor. Input yang perlu dilakukan oleh mahasiswa baru pun tidak banyak, mahasiswa baru hanya dapat melakukan input dalam hal mengadakan perjanjian dengan astor, menulis sharing mahasiswa baru, dan mengumpulkan tugas mahasiswa baru. Sementara tamu bahkan hanya dapat menulis buku tamu saja. Akan tetapi, bila administrator, asisten tutor, dan mahasiswa baru harus melakukan login terlebih dahulu sebelum melakukan interaksi dengan sistem, maka tamu tidak perlu melakukan login.

3.3 Entity Relationship Diagram

Ada beberapa relasi antar *entity* yang terdapat di dalam *database*. Relasi-relasi tersebut dapat dilihat pada Gambar 6.



1 / 1

Gambar 6. ***Entity Relationship Diagram***

4. IMPLEMENTASI SISTEM

Web application dibuat dengan AngularJS sebagai *framework* yang membantu pengembangan *front-end* dan Laravel sebagai *framework* yang membantu pengembangan *back-end*. *Database* yang digunakan adalah MySQL. *Web application* juga menggunakan JSON Web Token dalam melakukan autentikasi dan sisi *back-end* berperan sebagai penyedia layanan dengan memakai konsep arsitektur REST.

4.1 Pengaturan Awal

Sebelum memulai proses implementasi program dengan menggunakan Laravel dan AngularJS, hal yang perlu dilakukan terlebih dahulu adalah melakukan *download framework* Laravel. Dalam hal ini, *download framework* Laravel akan dilakukan melalui aplikasi Composer. Penggunaan aplikasi Composer membantu dalam melakukan instalasi *dependencies* yang diperlukan oleh Laravel secara otomatis. Selain itu, penggunaan aplikasi Composer juga memungkinkan pengembangan lebih lanjut dari aplikasi Laravel yang telah dibuat, dengan menambahkan fungsionalitas dari Laravel melalui *plugin* dengan cara yang mudah.

Setelah selesai melakukan *download framework* Laravel, maka langkah berikutnya yang perlu dilakukan adalah melakukan konfigurasi dari file '.env' yang terdapat pada folder utama proyek Laravel. File tersebut menyimpan konfigurasi-konfigurasi yang diperlukan oleh Laravel, salah satunya untuk melakukan koneksi dengan *database*.

4.2 Pemrograman AngularJS dan Laravel

AngularJS berperan sebagai *framework front-end*, sementara Laravel berperan sebagai *framework back-end* yang menyediakan layanan dengan memakai konsep arsitektur REST. Dengan begitu, sisi *front-end* memiliki peran dalam menerima input yang dilakukan pengguna dan meneruskan proses tersebut menuju *back-end*, ataupun *back-end* menyediakan data yang diminta oleh *front-end* dan *front-end* menampilkan data tersebut. Berikut adalah contoh dari penggunaan AngularJS pada sisi *front-end* dan Laravel pada sisi *back-end* dalam melakukan proses *login.*

Proses *login* adalah proses memasukkan *username* dan *password*, yang kemudian diperiksa di dalam server. Bila *login* berhasil, pengguna akan diarahkan masuk ke dalam halaman *home* sesuai dengan peran masing-masing.

Ketika pengguna memasukkan *username* dan *password*, maka data akan dikirim ke controller Login di sisi Laravel. $http kemudian menerima respons dari *request* yang sudah dikirimkan. Bila mendapat respons berupa token (menunjukkan bahwa request berhasil dan *valid*), maka token tersebut akan diperiksa untuk menentukan peran dari subject yang dimaksud dari token yang diterima. Jika subject bernama 'petrasinergi', maka akan diarahkan ke halaman administrator, tetapi jika bukan, maka *subject* akan diperiksa dengan mengirimkan *request* $http untuk memeriksa apakah *subject* adalah asisten tutor atau bukan. Jika *subject* adalah asisten tutor, maka akan diarahkan ke halaman astor, tetapi jika bukan, maka *subject* akan kembali diperiksa apakah *subject* adalah mahasiswa atau bukan. Pesan *error* akan dikeluarkan bila pengguna bukan administrator, asisten tutor, atau mahasiswa yang aktif.

```
$http({
    method: 'POST',
    url: API_URL + "login",
    data: $httpParamSerializerJQLike($scope.login),
    headers: {'Content-Type': 'application/x-www-
form-urlencoded'}
}).success(function(e) {
    var base64Url = e.split('.')[1];
    var base64 = base64Url.replace('-',
'+').replace('_', '/');
    var subject =
JSON.parse($window.atob(base64)).sub;
    if(subject == "petrasinergi"){
        localStorageService.set('tps', e, 180);
        window.location= "/admin";
    }
    else{
        $http.get(API_URL + "astor/login/" +
subject).success(function(response) {
            $scope.astor = response;
            if($scope.astor != ''){
                localStorageService.set('tps', e,
180);
                window.location="/astor";
            }
            else{
                $http.get(API_URL + "mahasiswa/" +
subject).success(function(response) {
                    $scope.maba = response;
                    if($scope.maba != ''){

localStorageService.set('tps', e, 180);
                        window.location="/maba";
                    }
                    else{
                        alert('Tidak ada mahasiswa
dalam database! Pastikan anda adalah astor atau
mahasiswa baru.');
                    }
                });
            }
        });
    }
}).error(function() {
    alert('Maaf, data login tidak benar. Silahkan
diperiksa kembali.');
});
```

Gambar 7. *Login* dari Sisi AngularJS

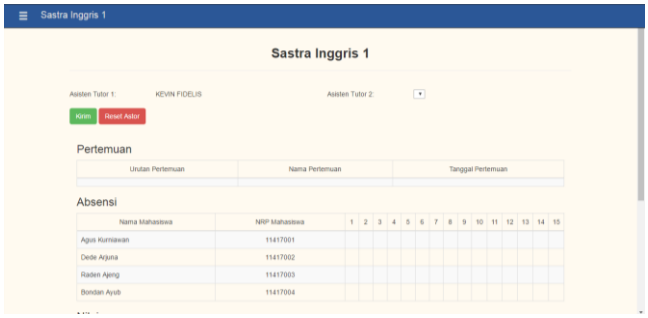
Controller 'Login' di sisi Laravel menerima request yang telah dikirimkan dari sisi Javascript, kemudian memeriksa apakah *username* yang diterima berasal dari administrator. Jika *username* yang diterima adalah *username* administrator, maka controller akan mencocokkan *username* dan *password* dengan server Gmail. Jika kombinasi *username* dan *password* cocok, maka token akan dibuat dan dikirim. Tetapi jika tidak cocok, *controller* akan mengirimkan pesan *error* sebagai respons kepada Javascript. Jika *username* adalah asisten tutor atau mahasiswa baru, maka *controller* akan memeriksa *username* dan *password* dengan server Universitas Kristen Petra dan kemudian mencocokkan data *username* dengan tabel mahasiswa. Jika kombinasi *username* dan *password* cocok serta *username* yang bersangkutan didapati sebagai mahasiswa aktif, *controller* akan membuat dan mengirimkan token. Sebaliknya, jika kombinasi *username* dan *password* salah atau mahasiswa yang bersangkutan bukan mahasiswa aktif, *controller* akan mengirimkan pesan *error*.

```
public function login(Request $request) {
    $username = $request->input('username');
    $password = $request->input('password');
    if ($username == "petrasinergi@gmail.com" ||
$username == "petrasinergi")
    {
        $mail =
imap_open('{imap.gmail.com:993/imap/ssl/novalidate-
cert}', $username, $password);

        $at_limit = explode("@", $username);
        $user = $at_limit[0];
        $customClaims = ['sub' => $user];
        $payload = JWTFactory::make($customClaims);
        $token = JWTAuth::encode($payload);

        imap_close($mail);
        return $token;
    }
    else {
        $mail =
imap_open('{john.petra.ac.id:110/pop3/novalidate-
cert}', $username, $password);

        $at_limit = explode("@", $request-
>input('username'));
        $user = str_replace('m', '', $at_limit[0]);


if(MahasiswaModel::where('status_mahasiswa','=',1)-
>find($user)) {
            $customClaims = ['sub' => $user];
            $payload =
JWTFactory::make($customClaims);
            $token = JWTAuth::encode($payload);
        }
        else{
            return false;
        }

        imap_close($mail);
        return $token;

    }
}
```

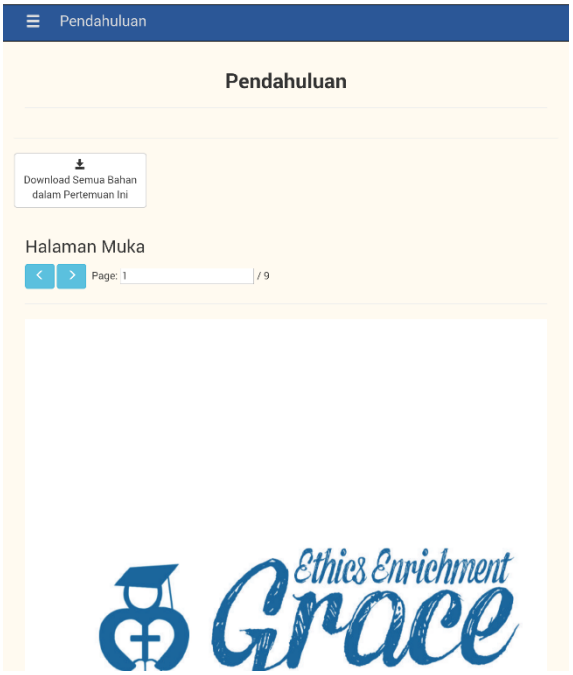
Gambar 8. *Login* dari Sisi Laravel

5. PENGUJIAN SISTEM

Penulis melakukan pengujian sistem dan membuktikan bahwa sistem dapat berjalan dengan baik. Sistem juga dapat diakses di berbagai jenis perangkat. Gambar 9, 10, dan 11 menunjukkan bagaimana *web application* diakses dengan menggunakan berbagai perangkat.



Gambar 9. Halaman Kelompok pada Sisi Administrator Dibuka dengan Komputer



Gambar 10. Halaman Bahan Pertemuan dari Sisi Asisten Tutor Dibuka dengan Tablet

Gambar 11. Input Perjanjian Pertemuan dengan Asisten Tutor dari Sisi Mahasiswa Baru Dibuka dengan Smartphone

6. KESIMPULAN DAN SARAN

6.1 Kesimpulan

- *Web application* terbagi atas bagian administrator, asisten tutor, mahasiswa baru, dan tamu dengan setiap

bagian memiliki fitur-fitur seperti manajemen data materi, data kelompok *Ethics Enrichment*, data penilaian, data asisten tutor, data mahasiswa baru.

- *Web application* bersifat responsif, dapat digunakan baik melalui komputer, *tablet*, maupun *smartphone*.
- Berdasarkan hasil survei yang dilakukan, didapatkan hasil yaitu 70% responden menyatakan bahwa *web application* mudah digunakan, 59% responden menyatakan bahwa *web application* bermanfaat, dan 53% responden menyatakan bahwa *web application* secara keseluruhan sudah baik. Karena itu, dapat disimpulkan bahwa *web application* menjawab kebutuhan dalam mendukung pelayanan asisten tutor di Universitas Kristen Petra.

6.2 Saran

- *Web application* dapat dikembangkan untuk juga mendukung *Ethics Enrichment Life* dan Mentor.
- Untuk pengembangan ke depan, *web application* dapat dikembangkan menjadi sebuah aplikasi *mobile* dengan menggunakan PhoneGap atau Appcelerator Titanium.

7. DAFTAR REFERENSI

[1] Auth0 Inc. n.d. Introduction to JSON Web Tokens. URI= https://jwt.io/introduction/

[2] Gilmore, W.J. 2015. Easy Laravel 5: A Hands On Introduction Using a Real-World Project, 1-33. URI= https://leanpub.com/easylaravel

[3] Google. 2016. AngularJS: Developer Guide: Introduction. URI=https://docs.angularjs.org/guide/introduction

[4] Green, B., Seshadri, S. 2013. AngularJS. URI=https://digitalrepo.files.wordpress.com/2015/07/brad-green-angularjs.pdf

[5] Heitkotter, H., Hanschke, S., Majchrzak, T.A. 2013. Evaluating Cross-Platform Development Approaches for Mobile Applications. URI= http://www3.nd.edu/~cpoellab/teaching/cse40814/crossplatform.pdf

[6] Hekman, A.J. 2014. AngularJS Services and Factories Done Right. URI= https://www.mutuallyhuman.com/blog/2014/05/08/angularjs-services-and-factories-done-right/

[7] Nugraha, T. n.d. Tutorial Dasar Laravel. URI=http://pondokprogrammer.com/wp-content/uploads/2014/12/tutorial_dasar_laravel.pdf

[8] Rouse, M. 2014. REST (representational state transfer). URI=http://searchsoa.techtarget.com/definition/REST

[9] Sharp, S. 2013. Web Site vs. Web Application… What's the Difference? URI=http://plexusweb.com/2013/04/web-site-vs-web-application-whats-the-difference/

[10] Surguy, M. 2014. Laravel: my first framework. Chapter 6 – Database Operations. URI=https://maxoffsky.com/code-blog/laravel-first-framework-chapter-6-database-operations/