9th Industrial Research Workshop and National Seminar
Peran Penelitian dan Inovasi di Era Industri 4.0 Dalam Mewujudkan Pembangunan Berkelanjutan Menuju Kemandirian Bangsa
POLBAN
IRONS

Analisa Penerapan *MEAN Stack* Dalam Pengembangan Web Berbasis Deklaratif

Ghifari Munawar

Jurusan Teknik Komputer dan Informatika, Politeknik Negeri Bandung, Bandung 40012
E-mail : ghifari.munawar@polban.ac.id

ABSTRAK

Pengembangan web berbasis deklaratif telah menjadi cita-cita sejak awal dikembangkannya aplikasi ini. Berbagai teknologi telah diupayakan untuk menyederhanakan *effort* pengembangan website, diantaranya ialah dengan berkembangnya teknologi *javascript* serta teknologi pendukung lainnya seperti AJAX, DOM, JSON, dlsb. Banyaknya framework *javascript* yang berkembang saat ini merupakan upaya untuk mengurangi ketergantungan terhadap teknologi *server side*, dimana implementasinya akan lebih banyak menggunakan teknologi *client side* mulai dari pengembangan *backend* hingga *frontend* [1]. Pendekatan secara deklaratif lebih memperhatikan aspek *what* (permasalahan apa yang ingin diselesaikan) oleh sistem, dibandingkan dengan aspek *how* (bagaimana menyelesaikan permasalahan) pada sistem [3]. Salah satu upaya untuk mengembangkan web secara deklaratif adalah dengan menggunakan framework *javascript*. Saat ini ada banyak framework *javascript* yang berkembang, namun yang popular dalam mengimplementasikan *full stack development* adalah *MEAN* (*MongoDB*, *Express*, *Angular*, dan *Node JS*). *MEAN* merupakan kombinasi dari beberapa framework *javascript* yang dapat mencakup seluruh teknologi yang dibutuhkan dalam mengembangkan aplikasi web mulai dari pengembangan *backend* (*Node JS*, *MongoDB*, *Express*) sampai dengan pengembangan *frontend* (*Angular JS*) [6]. Penelitian ini bertujuan untuk menganalisa penerapan *MEAN* dalam pengembangan web deklaratif. Berdasarkan implementasi yang telah dilakukan, framework *MEAN* dapat mendukung pengembangan web berbasis deklaratif.

Kata Kunci
MEAN Stack, Web Deklaratif, MongoDB, Express, Angular, Node JS

1. PENDAHULUAN

Pengembangan aplikasi web berbasis deklaratif telah menjadi cita-cita sejak awal dikembangkannya aplikasi ini. Berbagai teknologi telah diupayakan untuk menyederhanakan *effort* pengembangan website, diantaranya ialah dengan berkembangnya teknologi *javascript* serta teknologi pendukung lainnya seperti AJAX, DOM, JSON, dlsb. Banyaknya framework *javascript* yang berkembang saat ini merupakan upaya untuk mengurangi ketergantungan terhadap teknologi *server side* (seperti PHP, ASP, JAVA, SQL, dll), dimana implementasinya akan lebih banyak menggunakan teknologi *client side* (seperti *html*, *javascript*, JSON, dll) mulai dari pengembangan *backend* hingga *frontend* [1]. Salah satu kendala yang saat ini masih dihadapi dalam pengembangan web deklaratif adalah implementasi dari *business logic* aplikasi yang pada umumnya masih dilakukan secara imperatif [1].

Pengembangan web tradisional umumnya membagi arsitektektur kedalam 3 (tiga) layer, diantaranya : (1) *presentation layer*, (2) *business logic*, dan (3) *data management layer* [2]. Kesulitan yang dihadapi oleh pengembang web adalah perlunya penguasaan contoh, pada *presentation layer*, umumnya menggunakan pemrograman *client* seperti *javascript*, *html*, dan *css*. Pada *business logic layer* menggunakan bahasa pemrograman *server* seperti PHP, ASP, C#, Java, dll. Sedangkan pada *data management layer* menggunakan bahasa pemrograman SQL. Seorang pengembang web umumnya jarang menguasai teknologi tersebut sekaligus dikarenakan masing-masing layer menerapkan teknologi yang berbeda. Sehingga perlu ada alternatif lain agar lebih memudahkan *effort* pengembangannya.

Dalam pengembangan web modern, dikenal dengan istilah *declarative web development*, dimana pengembangan aplikasi web dapat dilakukan secara deklaratif dengan mengutamakan pendekatan *what* (permasalahan apa yang ingin selesaikan oleh sistem), dibandingkan dengan pendekatan *how* (bagaimana menyelesaikan suatu permasalahan pada sistem) [3]. Untuk mengimplementasikan hal tersebut salah satunya adalah menggunakan framework *javascript*. Saat ini ada banyak framework yang berkembang seperti *Knockout JS*, *Vue JS*, *Ember JS*, *React JS*, dll. Namun yang cukup populer dalam

MEAN (*MongoDB*, *Express*, *Angular*, dan *Node JS*). MEAN merupakan kombinasi dari beberapa framework *javascript* yang dapat mencakup seluruh teknologi yang dibutuhkan dalam mengembangkan aplikasi web mulai dari pengembangan *backend* (*Node JS*, *MongoDB*, *Express*) hingga pengembangan *frontend* (*Angular JS*) [6].

Framework tersebut dapat menyederhanakan *effort* dalam pengembangan web, dengan memberikan dukungan teknologi yang besar pada sisi *client*, baik untuk desain tampilan, proses logika, akses ke database, *routing*, dlsb hanya dengan bahasa *javascript*. Hal ini akan menguntungkan bagi para pengembang web, karena menyederhanakan proses pengembangan web dengan tidak lagi bergantung pada teknologi *server side*. Penelitian ini bertujuan untuk menganalisa penerapan *MEAN Stack* dalam mendukung pengembangan website secara deklaratif.

2. DASAR TEORI

2.1 Pengembangan Web Deklaratif

Pemrograman deklaratif merupakan suatu pendekatan dalam membuat struktur dan elemen dari aplikasi dengan mengekspresikan apa (*what*) yang sistem harus lakukan untuk menyelesaikan permasalahan, dibandingkan bagaimana (*how*) mengimplementasikan logika program untuk menyelesaikan permasalahan pada sistem [3]. Pengembangan web deklaratif pada umumnya mengarah kepada penggunaan teknologi *client side* seperti *html*, *javascript*, *css*, dan *xml*. Pada pengembangan tradisional, teknologi ini hanya digunakan untuk mengembangkan *presentation layer* / *user inteface*. Berkembangnya framework *javascript* memberikan keuntungan dalam pendekatan ini, dimana teknologi *client side* juga dapat diperluas dengan fungsionalitas yang dimiliki oleh teknologi *server side*. Sehingga pengembangan pada layer *business logic* dan *data management* juga dapat dilakukan secara deklaratif [1].

2.2 Framework Javascript

Javascript merupakan sebuah *scripting language* yang diturunkan dari standar *ECMA Script* [4] yang mendukung beberapa paradigma pemrograman seperti pemrograman imperatif, dan pemrograman deklaratif. Framework *javascript* merupakan sekumpulan *library javascript* yang bertujuan untuk mempermudah pengembangan website. Eksositem *javascript* dalam mendukung pengembangan aplikasi web dari waktu ke waktu semakin menunjukan eksistensinya, termasuk untuk mendukung pengembangan aplikasi web skala besar, pembuatan *tools*, *problem solving*, dan pengimplementasian sebelumnya [5]. Beberapa framework *javascript* yang saat ini berkembang dapat dibagi kedalam berbagai kategori berikut:

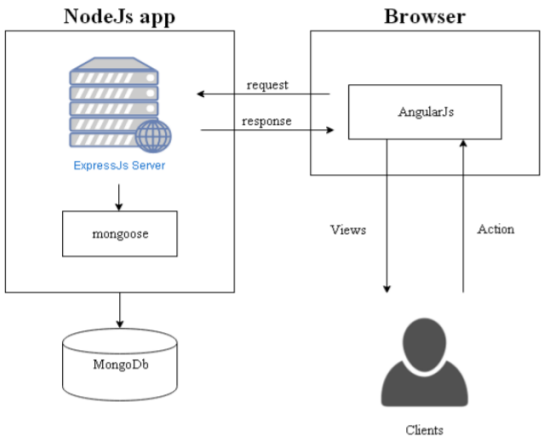
Tabel 1. Kategori framework *javascript* [5]

Framework Type	Framewok Example
Desktop Like	GWT Ext JS
MVC with HTML templates, including data binding	Backbone Ampersand Knockout JS CanJS Ember
HTML extension, data binding	Angular
Web Component Like	Polymer
Plain Library	Jquery
Virtual DOM Renderer (including flux architecture)	React Redux
CSS and Javascript utilities for UI	Bootstrap

Evaluasi framework *javascript* dapat dilakukan dengan mempertimbangkan beberapa faktor seperti kemudahan untuk dipahami, popularitas, jumlah pengguna, fleksibilitas, paradigma data, penggunaan pada skala *enterprise*, dll [5].

2.3 MEAN

Istilah MEAN mulai diperkenalkan pada tahun 2013 dan merupakan kombinasi dari beberapa framework yang *open source* untuk membangun aplikasi web secara dinamis menggunakan *javascript*. *MEAN* dapat mencakup seluruh teknologi yang dibutuhkan dalam mengembangkan aplikasi web mulai dari pengembangan *backend* sampai dengan pengembangan *frontend* [6]. Komponen MEAN, yaitu *MongoDb*, *Express*, *Angular*, dan *Node JS*. Gambar 1 menunjukan *workflow* pada *MEAN*, dimana lingkup *frontend* menggunakan framework *angular*, sedangkan lingkup *backend* menggunakan *mongodb*, *express*, dan *node js*.



Gambar 1. *Workflow of MEAN Stack* [7]

2.3.1. MongoDb

MongoDb merupakan tipe database *NoSQL* yang berorientasi pada dokumen. Berbeda dengan database *relational* pada umumnya, *mongodb* menyimpan data dalam format *binary encoded json* (BSON). *MongoDb* digunakan pada lingkungan *javascript* pada sisi *server* untuk menampung data melalui fungsionalitas yang disediakan oleh *node js*. *Mongoose* merupakan *package* yang dimiliki oleh *node* untuk mengoperasikan *create retrieve update delete* (CRUD) pada *mongodb*.

2.3.2. Express

Express merupakan framework *server side* yang dibangun dalam lingkup *node js*. Framework ini digunakan untuk mengelola *request* dari *client* ke *server* termasuk untuk pengelolaan *routing* dan operasi HTTP (`PUT`, `GET`, `POST`, dll). *Express* dapat dikatakan sebagai suatu *middleware* yang bertanggungjawab dalam mengelola siklus *request-response* dan menjamin tidak ada suatu *request* yang tertinggal (dibiarkan menggantung) [7].

2.3.3. Angular

Angular merupakan framework *javascript* yang dikembangkan oleh Google dan digunakan dalam menangani seluruh aplikasi *client side* dan interaksinya. Secara spesifik framework ini mengembangkan *single page application* (SPA) yang memuat seluruh halaman website kedalam satu halaman (page) saja. *SPA* memiliki beberapa keunggulan, diantaranya :

- Tidak ada *page refresh*
- *User experience* yang lebih baik
- Kemampuan untuk bekerja secara *offline*

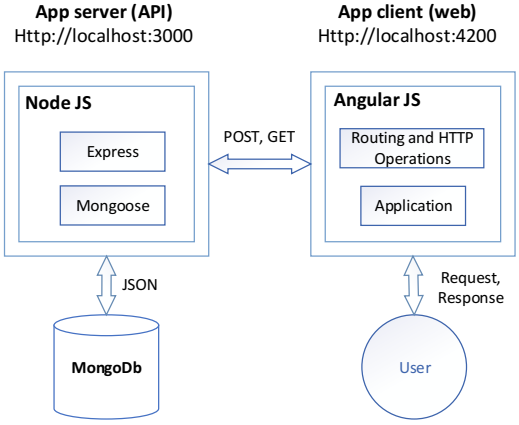
Angular menerapkan pemrograman secara deklaratif dimana fungsionalitas framework dapat disisipkan secara langsung pada kode *html* melalui *directives*. `Ng-` yang digunakan untuk *binding* data dan manipulasi DOM.

2.3.4. Node JS

Node merupakan komponen yang paling penting dalam *MEAN*. *Node* menyediakan lingkungan *webserver* dengan performa tinggi dan ringan [8], serta ideal untuk membangun *webservice* API. *Node* memungkinkan pengembang untuk membuat *web server* dan *networking tools* menggunakan *javascript* dan sekumpulan modul yang mengelola berbagai fungsi utama. Modul tersebut menyediakan *file system* I/O, *networking*, *binary data*, kriptografi, *data stream*, dll. Saat ini *node* telah digunakan oleh beberapa perusahaan besar seperti : IBM, LinkedIn, Paypal, Yahoo, Walmart, dll [9].

3. IMPLEMENTASI & ANALISA PENERAPAN

Implementasi dilakukan dengan membangun sebuah website katalog produk. Website yang dibangun memiliki operasi dasar CRUD, dengan beberapa fungsional seperti menampilkan daftar produk, melihat detail produk, menambah produk, menghapus produk, melakukan pencarian, dll. Website dikembangkan dengan arsitektur *RESTful* API, dimana implementasinya dilakukan secara terpisah antara *app-server* dengan *app-client* nya sehingga masing-masing dapat memiliki alamat *server* dan nomor *port* yang berbeda. Arsitektur sistem dapat dilihat pada Gambar 2.



Gambar 2. Arsitektur Sistem

Spesifikasi yang digunakan dalam implementasinya adalah sebagai berikut :

- *Node* versi 8.11.2
- *Mongodb* versi 3.6.5
- *Angular* 5
- *Express* versi 4.16.3
- *Mongoose* versi 5.1.6

Integrated development environment (IDE) yang digunakan adalah *Visual Studio Code* versi 1.23.1.

3.1 Implementasi Database

Skema tabel dibuat dalam file JS (*javascript*), diimplementasikan 4 model, diantaranya :

- *category*
- *product*
- *catalog*
- *contactUs*

Masing-masing skema dibuat struktur datanya dengan format JSON. Berikut ini adalah kode *javascript* untuk mendefinisikan skema *Catalog* :

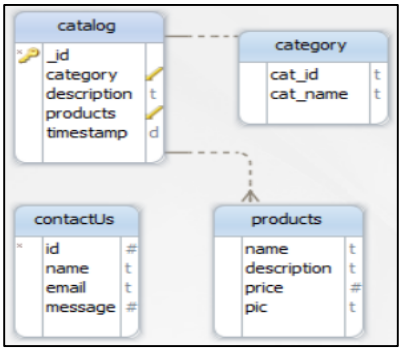
```
var mongoose=require('mongoose');

var catalogSchema =
mongoose.Schema({
    category : categorySchema,
    description: String,
    products: [productSchema]}, {
    timestamps: true});

module.exports =
mongoose.model('Catalog',
catalogSchema);
```

Gambar 3. Skema *Catalog*

Node js menggunakan *mongoose* sebagai komponen *object document model* (ODM) untuk mengelola pembuatan skema database pada *mongodb*. Dapat dikatakan bahwa komponen tersebut mendukung untuk pengembangan deklaratif, dimana teknis pemrogramannya lebih banyak menggunakan fungsi-fungsi yang telah disediakan oleh komponen. Sebagai contoh fungsi `mongoose.Schema` yang digunakan untuk mendefinisikan skema tabel database, serta fungsi `mongoose.Model` sebagai *type casting* (*mapping*) antara objek database dengan modelnya.



Gambar 4. Desain database

Gambar 4 adalah desain database yang dihasilkan

3.2 Implementasi *App-server* (API)

Berikut adalah daftar *endpoint* API yang dikembangkan :

Tabel 2. Daftar *endpoint* API

URI	Method
`/api/category`	`POST`
`/api/category`	`GET`
`/api/category/:id`	`GET`
`/api/product`	`POST`
`/api/product`	`GET`
`/api/product/:id`	`GET`
`/api/catalog`	`POST`
`/api/category/:id/product`	`GET`
`/api/contactus`	`POST`

Routing pada *endpoint* diimplementasikan menggunakan `Router` di framework *express js*. Objek `Router` tersebut telah menyediakan operasi *http* baik untuk method `POST`, `GET`, `PUT`, dan `DELETE`. Berikut ini adalah kode *javascript* untuk mengimplementasikan *endpoint* `/api/contactus` yang menggunakan *router* dalam memetakan method `POST` :

```
var express = require('express');
var router = express.Router();
var Contact =
require('../models/ContactUs.js')
;

router.post('/api/contactus',
function(req, res, next) {
  Contact.create(req.body,
function (err, post) {
    if (err) return next(err);
    res.json(post);
  });
});
```

Gambar 5. Implementasi `/api/contactus`

Dapat dilihat pada Gambar 5, untuk memasukkan data (`INSERT`) ke dalam skema c*ontactUs* tidak menggunakan *native query* sebagaimana umumnya dilakukan pada pengembangan tradisional, namun dengan memanggil fungsi `create` dengan parameter berupa *request body* berdasarkan hasil inputan user. Hal ini disebabkan implementasinya telah menggunakan *mongoose* sebagai *object document model* (ODM) pada *mongodb*, sehingga operasi CRUD dapat dengan mudah dilakukan dengan memanggil masing-masing fungsinya. Sebagai contoh lain pada Gambar 5 adalah implementasi *endpoint* `/api/category/:id` yang akan mengembalikan hasil pencarian kategori

Fungsi yang dipanggil pada model adalah `findById` dengan parameter berupa *request parameter* berdasarkan `id` yang dicari.

```
router.get('/api/category/:id',
function(req, res, next) {

Category.findById(req.params.id,
function (err, post) {
    if (err) return next(err);
    res.json(post);
  });
});
```

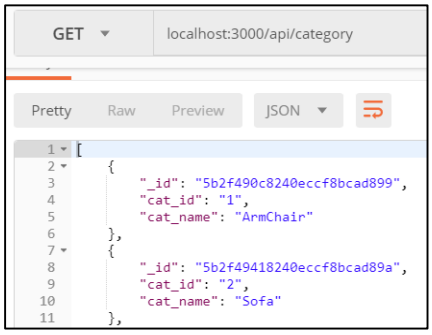
Gambar 6. Implementasi `/api/category/:id`

Sesuai rancangan arsitektur, *app-server* dikonfigurasikan pada *port* 3000.

```
var app = express();
app.listen('3000', function () {
    console.log("Server running
at http://localhost:3000");
});
```

Gambar 7. Konfigurasi *port* pada *app-server*

Fungsionalitas API yang dikembangkan kemudian diuji menggunakan *Postman*.



Gambar 8. Uji fungsionalitas API pada *Postman*

3.3 Implementasi *App-client* (Web)

Pada implementasi *app-client*, framework yang digunakan adalah *angular js*. Untuk memetakan (*route*) seluruh *request* API ke *app-server*, maka dibuatkan konfigurasi *proxy*-nya terlebih dahulu :

```
"/api/*":
{
    "target" :
"http://localhost:3000",
    "secure" : false,
    "logLevel" : "debug",
    "changeOrigin" : true
}
```

Gambar 9. Konfigurasi *Proxy Server*

Angular menerapkan *SPA*, halaman utama diimplementasikan pada *index.html* sebagai *container*, dengan beberapa halaman lain yang didefinisikan dalam bentuk komponen. Komponen diimpor dari `@angular/core` dan diidentifikasi menggunakan `selector`. Beberapa komponen yang dibangun diantaranya :

- *header*
- *home*
- *product*
  - *product-list*
  - *product-detail*
- *contact-us*
- *page-not-found*

Berikut ini adalah kode untuk membangun komponen *header* :

```
import { Component } from
'@angular/core';
...
@Component({
  selector: 'app-header',
  templateUrl:
'./header.component.html',
  styleUrls:
['./header.component.css']
})
```

Gambar 10. Komponen *header*

Komponen pada *angular* bersifat *reusable*, ketika telah dibangun, maka dapat digunakan ulang sesuai dengan kebutuhan. Pemanggilan komponen berupa tag sesuai dengan identifikasi pada `selector`-nya. Komponen *header* dapat dipanggil dengan menuliskan tag `<app-header>`.

Validasi pada form dapat dilakukan secara deklaratif menggunakan *directives*. *Directives* disisipkan langsung pada kode *html* sebagai atribut. Gambar 11 menunjukan penggunaan `ngIf` untuk memvalidasi input form.

```
...
<span
*ngIf="!contactForm.get('username
').valid &&
contactForm.get('username').touch
ed" class="help-block">
...
```

Gambar 11. *Directives* `*ngIf` untuk validasi form

Selain itu untuk mengakses (*looping*) item data pada *array* menggunakan *directives* `ngFor`.

```
<div class="col-md-4 mb-5"
*ngFor="let product of
productList">
  <h4 class="card-title">
{{product.name}} </h4>
  <p class="card-text">
     {{product.description}
  </p>
...
</div>
```

Gambar 12. *Directives* *ngFor untuk *looping* item pada *array*

Dalam memetakan operasi *http* (POST dan GET) ke *endpoint* API, *angular* menggunakan Http yang diimpor dari @*angular/http*.

```
import { Http } from
'@angular/http';
...
this.http.post("/api/contactus",
{
    name:  username.value,
    email: useremail.value,
    message : usermessage.value,
     { headers: headers })
     .subscribe((response:
Response) => {
     const data =
response.json();},
(error) => console.log(error))
```

Gambar 13. Operasi POST ke *endpoint* API

Angular menerapkan navigasi halaman pada sisi *client* menggunakan Routes yang diimpor dari @*angular/router*. *Service* ini memungkinkan setiap *request* halaman dari *user* untuk di-*mapping* sesuai dengan komponennya.

```
import { Routes} from
'@angular/router';
...
const appRoutes: Routes = [
  {path: '', redirectTo: '/home',
pathMatch: 'full' },
  {path: 'home', component:
HomeComponent },
  {path: 'contactus' , component:
ContactUsComponent },
  {path: 'not-found' , component:
PageNotFoundComponent },
  ...
  {path: '**', redirectTo: '/not-
found' }
]
```

Gambar 14. *Routing* komponen pada *angular*

3.4. Analisa Penerapan

MEAN Stack menggunakan *javascript* sebagai dasar pemrogramannya. Fungsionalitas yang dimiliki oleh *MEAN*, memungkinkan *javascript* untuk digunakan pada berbagai *layer*-nya, baik untuk akses database, pembuatan *RESTful* API, pengelolaan *routing*, hingga implementasi *single page application* (SPA). Format umum yang digunakan sebagai media pertukaran data antar layernya adalah *JSON*. *JSON* bersifat *native*, dan ringan yang dapat dengan mudah dikonsumsi oleh *javascript*.

Node js memiliki komponen yang digunakan untuk mengakses database, yakni *mongoose*. *Mongoose* digunakan sebagai *object document model* (ODM) yang memetakan objek pada database *mongodb* dengan model *javascript*-nya, serta menyediakan fungsi CRUD. Komponen *node* lain yang digunakan yaitu *express* untuk membuat *RESTful* API, dan mengelola operasi *http* (POST, GET, dll) pada sisi *server*. Disamping itu *angular* digunakan untuk mengelola *routing* pada sisi *client* agar sesuai dengan karakteristik *single page application* (SPA).

Dapat dikatakan bahwa implementasi *MEAN stack* mendukung untuk pengembangan web berbasis deklaratif. Dimana salah satu aspeknya adalah lebih banyak memanfaatkan teknologi *client side* (dalam hal ini *javascript*), serta tidak lagi bergantung pada teknologi *server side* (seperti PHP, ASP, JAVA, SQL, dll) yang umumnya digunakan dalam pengembangan web tradisional. Disamping itu berbagai *service* yang disediakan oleh *MEAN* dapat menyederhanakan *effort* pengembangan webnya, dimana hal ini sesuai dengan tujuan pengembangan deklaratif, yakni lebih memperhatikan aspek *what* (apa yang dibutuhkan oleh sistem), dibandingkan aspek *how* (bagaimana menyelesaikan permasalahan pada sistem).

4. KESIMPULAN

Berdasarkan implementasi yang telah dilakukan, MEAN *Stack* (*MongoDb*, *Express*, *Angular*, dan *Node JS*) mendukung untuk pengembangan web berbasis deklaratif. Kombinasi framework ini menyediakan berbagai *service* yang dibutuhkan untuk membangun *backend* dan *frontend* secara deklaratif hanya dengan *javascript* sebagai dasar pemrogramannya. Banyaknya *service* yang telah tersedia dapat mengurangi *effort* pengembangannya, hal ini sesuai dengan tujuan pengembangan web deklaratif, yakni lebih menekankan pada aspek *what* (permasalahan apa yang ingin diselesaikan oleh sistem) dibandingkan aspek *how* (bagaimana menyelesaikan permasalahan pada sistem). Penelitian selanjutnya dapat mengukur implementasi

non-functional, seperti pengujian pada aspek *reliability*, *portability*, *performance*, *reusability*, dlsb.

5. UCAPAN TERIMA KASIH

Ucapan terimakasih disampaikan kepada segenap sivitas akademika Jurusan Teknik Komputer dan Informatika (JTK) dan UPPM Politeknik Negeri Bandung yang telah memberikan kesempatan dan bantuan baik materiil maupun non-materiil kepada penulis dalam melaksanakan penelitian ini.

DAFTAR PUSTAKA

[1] Lorenz, David H., dkk. *Application Embedding : A Language approach to Declarative Web Programming*. The Art, Science, and Engineering of Programming Vol 1, No 1. CC By 4.0. 2017.

[2] Vuorima, Petri, dkk. *Leveraging declarative languages in web application development*. Springer. 2015.

[3] McGinnis, Tyler. *Imperative vs Declarative Programming*. Available : https://tylermcginnis.com/ imperative-vs-declarative-programming/. 2016.

[4] Ecma International, *ECMAScript Language Specification (ECMA-262)*. Available : http://www.ecma-international.org/publications/files /ECMA-ST /Ecma-262.pdf. 2011.

[5] W. Roby, dkk. *Firefly : Embracing the future web technologies*. IPAC, Caltech, Pasadena, CA, USA 91125. 2017.

[6] Wikipedia. MEAN (*software bundle*). Available : https://en.wikipedia.org/wiki/MEAN_(software_bundle). Diakses pada : Januari 2018.

[7] Nirgudkar, Ninaad, dkk. *The MEAN Stack*. Volume : 04 Issue : 05. IRJET. 2017.

[8] I. K. Chaniotis, K.-I. D. Kyriakou, and N. D. Tselikas, *Is Node.js a viable option for building modern web applications? A performance evaluation study*, Computing, pp. 1–22, 2014. [Online]. Available: http://dx.doi.org/10.1007/s00607-014-0394-9.

[9] Github. *Projects, Applications, and Companies Using Node*. Available: https://github.com/joyent/node/wiki/Projects,-Applications,-and-Companies-Using-Node. 2015.

[10] Bretz, Adam, dkk. *Full Stack Javascript Development with MEAN*. Sitepoint. 2014.

[11] Codeproject. *Single Page Application using Angular JS Tutorial*. Available : https://www.codeproject.com/ Articles/1224654/Single-Page-Application-using-AngularJs-Tutorial. 2018.