KWIK KIAN GIE
SCHOOL OF BUSINESS
© Hak Cipta Milik IBI KKG(Institut Bisnis dan Informatika Kwik Kian Gie)
Hak Cipta Dilindungi Undang-Undang
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber:
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penyusunan laporan, penulisan kritik dan tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar IBIKKG.
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin IBIKKG.
Institut Bisnis dan Informatika Kwik Kian Gie

BAB II

LANDASAN TEORI

A. Analisis Perancangan Sistem

Menurut Jogiyanto Hartono (2005:129) Analisis sistem (system analysis) dapat didefenisikan sebagai penguraian dari suatu sistem informasi yang utuh ke dalam bagian komponennya dengan maksud untuk mengidentifikasikan dan mengevaluasi permasalahan, kesempatan, hambatan yang terjadi dan kebutuhan yang diharapkan sehingga dapat diusulkan perbaikannya.

Menurut Tata Sutabri (2001:30), Perancangan sistem merupakan tahapan setelah analisis pengembangan sistem yang menggambarkan bagaimana suatu sistem dibentuk dan mendefinisikan kebutuhan fungsionalnya.

Tujuan dari perancangan sistem adalah untuk merancang kebutuhan sistem yang diinginkan oleh pemakai serta membuat gambaran yang jelas. Penerapan database dalam sistem informasi disebut dengan database system. Menurut Jogiyanto Hartono (2005:217) sistem basis data sebagai suatu sistem informasi yang mengintegrasikan kumpulan dari data yang saling berhubungan satu dengan lainnya dan membuatnya tersedia untuk beberapa aplikasi yang bermacam-macam di dalam suatu organisasi.

B. Sistem Informasi

Menurut Marakas O'Brien(2013 : 6), Sistem informasi (IS) dapat berupa kombinasi terorganisir dari *brainware, hardware, software,* jaringan komunikasi, *data resource,* dan kebijakan dan prosedur yang menyimpan, mengambil, mengubah, dan menyebarkan informasi dalam sebuah organisasi.

C. Data

Menurut Marakas O'Brien(2013 : 23), Data merupakan material mentah sistem informasi. Data adalah fakta atau pengamatan baku.

Terdapat 2 jenis sumber data :

1. Data Primer, Menurut Umar (2003:56) merupakan data yang diperoleh langsung di lapangan oleh peneliti sebagai obyek penulisan.
2. Data Sekunder, Menurut Sugiyono (2005 : 62) adalah data yang tidak langsung memberikan data kepada peneliti, misalnya penelitian harus melalui orang lain atau mencari melalui dokumen.

D. Informasi

Definisi informasi Menurut Marakas O'Brien(2013 : 23), Sebagai data yang telah diubah menjadi konteks yang berarti dan berguna bagi pengguna akhir tertentu. Dengan demikian, data yang biasanya mengalami proses nilai tambah (pengolahan data atau pengolahan informasi) selama bentuk nya dikumpulkan, dimanipulasi, dan terorganisir. konten nya dianalisis dan dievaluasi dan ditempatkan dalam konteks yang tepat

E. *Internet*

Definisi *Internet* Menurut Marakas O'Brien(2013 : 238), *Internet* sebagai sebuah jaringan yang terdiri dari jutaan jaringan pribadi yang lebih kecil, masing-masing dengan kemampuan untuk beroperasi independen, atau selaras dengan, semua jutaan lainnya dari jaringan yang terhubung ke *Internet*.

F. *Database* Transaksional

Menurut Simon Holmes(2016:15), Sebuah *database* transaksional dapat mengambil sejumlah operasi terpisah sebagai salah satu transaksi. Jika salah satu dari operasi dalam suatu transaksi harus gagal seluruh transaksi gagal, dan tidak ada operasi selesai. Hal ini penting jika perlu untuk memperbarui beberapa koleksi atau dokumen sekaligus.

G. *Data Store*

Menurut David Bryant Copeland(2016:v-ix) ,layer arsitektur biasanya mengacu pada *stack*, sejak kebanyakan diagram menggambarkan layer sebagai *stack block*. Tengah *stack* disebut *middleware* dimana pokok logika dari aplikasi. Bagian bawah *stack* adalah *data store* dimana data yang bernilai disimpan dan memanipulasi aplikasi. Ini

KWIK KIAN GIE SCHOOL OF BUSINESS
Hak Cipta Dilindungi Undang-Undang
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber:
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penyusunan laporan, penulisan kritik dan tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar IBIKKG.
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin IBIKKG.
© Hak cipta milik IBIKKG (Institut Bisnis dan Informatika Kwik Kian Gie)
Institut Bisnis dan Informatika Kwik Kian Gie

biasanya *relational database management system (RDMS*). Bagian atas *stack* adalah *user interface*. Setiap bagian dari *stack* mempunyai peran yang krusial dalam pengembangan *software*. *Data store* merupakan tempat asset perusahaan yang paling penting yaitu data. Meskipun perusahaan kehilangan semua *source code* nya, selama adanya data, perusahaan dapat tetap betahan. Kehilangan semua data akan menjadi bencana besar. Bagian atas *stack* juga penting, sebagai lajur pengguna melihat dan mengentri data. Bagi user, *user interface* merupakan *database*. *Middleware* yang jelek sulit untuk diubah, berarti biaya perubahan tinggi, dengan demikian kemampuan perusahaan untuk merespon perubahan lebih sulit.

H. Media sosial

Menurut Michael Dewing mengenai media sosial(2012:1),"Fungsi media sosial adalah manusia dapat saling bertukar foto dan *video*, saling berbagi pengalaman, *post* mengenai apa yang dipikirkan dan berpartisipasi dalam diskusi *online*. Media sosial juga memungkinkan individual, perusahaan, organisasi, pemerintah dan anggota parlemen untuk saling berkomunikasi dengan orang dalam jumlah yang banyak. Macam *internet service* yang sering diasosiasikan dengan media sosial :

1. *Blog(web log)* : jurnal online dimana halamannya ditampilkan dengan urutan kronologi terbalik. *blog* dapat di *hosting* secara gratis seperti *WordPress, Tumblr, dan Blogger.*
2. *Wikis* : kumpulan website dimana semua orang dapat berpartisipasi untuk memodifikasi halaman atau membuat halaman baru menggunakan *web browser*. salah satu contoh yang terkenal adalah *wikipedia, encyclopedia* online yang gratis
3. *Social Bookmarking* : situs yang memungkinkan pengguna untuk mengorganisasikan dan *share* atau berbagi *link website*. contoh : *reddit, StumbleUpon* dan *Digg*
4. *Social Network site* didefinisikan sebagai layanan berbasis *web* yang memungkinkan individual untuk membangun *profile* secara publik atau semi-publik dengan sistem yang dibatasi, berhubungan dengan list pengguna lain yang

KWIK KIAN GIE SCHOOL OF BUSINESS
Hak cipta milik IBI KKG (Institut Bisnis dan Informatika Kwik Kian Gie)
Hak Cipta Dilindungi Undang-Undang
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber:
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penyusunan laporan, penulisan kritik dan tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar IBIKKG.
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin IBIKKG.
Institut Bisnis dan Informatika Kwik Kian Gie

dapat berbagi koneksi, melihat dan terkait dengan list koneksi dan yang dibuat oleh pengguna lain malalui sistem. contoh : *Facebook* dan *LinkedIn*

5. *Status Update service (microblogging service)* : memungkinkan pengguna untuk *share update* secara singkat mengenai orang atau *event* dan untuk melihat *update* yang pengguna lain buat. contoh : *Twitter*
6. *Virtual world content* : situs yang menawarkan *virtual environment* seperti *game*. contoh : dunia imaginer yang dibuat pada *Second Life* yang mana user membuat *avatar*(*virtual* yang mempresentasikan pengguna) yang berinteraksi dengan lainnya
7. *Media sharing site* : situs memungkinkan user untuk posting *video* atau foto. contoh : *YouTube, Pinterest dan Instagram*

I. Data Administrasi

Data administrasi atau *data resource management* merupakan pekerjaan secara organisasi pada bagian sistem informasi dan *computer science* yang merancang, mengoraganisasikan mendeskripsikan dan mengontrol sumber data. Sumber data biasanya disimpan pada *database management sistem* atau *electronic spreadsheet*.

Menurut Craig S. Mullins(2013:15), *Data administration* (DA) memisahkan aspek bisnis untuk *data resource management* dari teknologi yang digunakan untuk mengelola data. Ini lebih mirip dengan business user data yang sebenarnya. DA bertanggung jawab untuk memahami kosa kata bisnis dan menerjemahkannya ke dalam logical data model. Mengacu kembali ke ADLC, DA akan terlibat lebih dalam pengumpulan persyaratan, analisis, dan tahap desain, DBA dalam desain, pengembangan, pengujian, dan operasional. DA bertanggung jawab untuk:

1. Identifikasi dan mendaftarkan data yang dibutuhkan oleh bisnis user
2. Memproduksi konseptual dan model data logis yang akurat menggambarkan hubungan antara elemen data untuk proses bisnis
3. Membuat data model perusahaan yang menggabungkan semua data yang digunakan oleh semua proses bisnis organisasi
4. Menetapkan kebijakan data untuk organisasi
5. Mengidentifikasi pemilik data dan steward
6. Menetapkan standar untuk kontrol dan penggunaan data

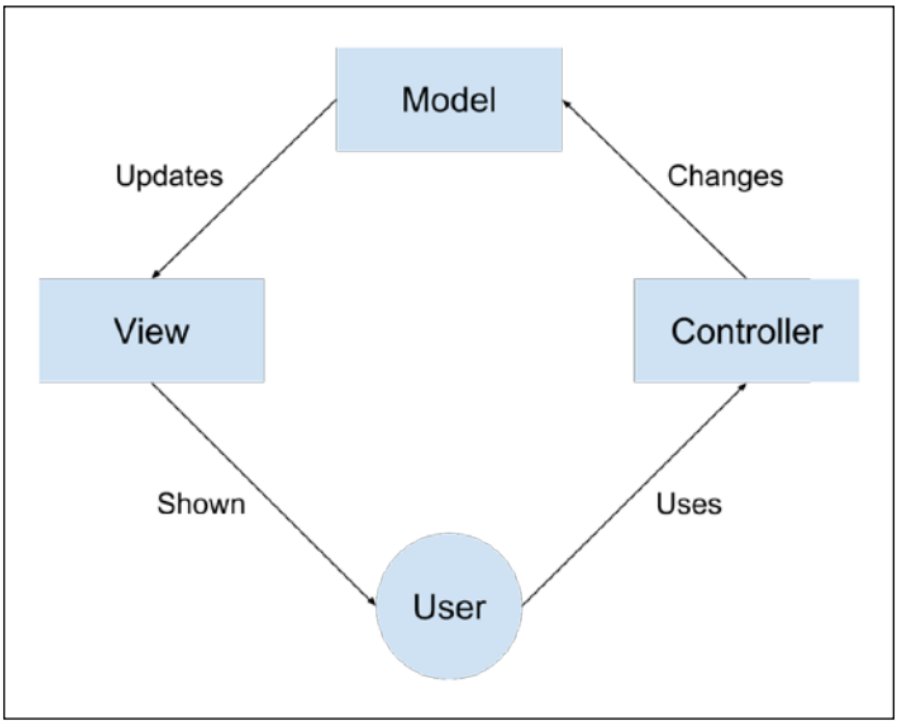
KWIK KIAN GIE SCHOOL OF BUSINESS
Hak Cipta Dilindungi Undang-Undang
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber:
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penyusunan laporan, penulisan kritik dan tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar IBIKKG.
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin IBIKKG.
© Hak cipta milik IBI KKG (Institut Bisnis dan Informatika Kwik Kian Gie)
Institut Bisnis dan Informatika Kwik Kian Gie

KWIK KIAN GIE SCHOOL OF BUSINESS
Hak cipta milik IBI KKG (Institut Bisnis dan Informatika Kwik Kian Gie)
Hak Cipta Dilindungi Undang-Undang
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber:
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penyusunan laporan, penulisan kritik dan tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar IBIKKG.
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin IBIKKG.
Institut Bisnis dan Informatika Kwik Kian Gie

J. *Three Tier Web Application Development*

Menurut Amos Q. Haviv(2014:8), Kebanyakan *web* aplikasi yang dibagun berdasarkan *Three-tier web application development* berisi tiga *layer* yaitu data, *logic* dan *presentation*. *Presentation layer* lebih befokus kepada *UI issue* dan bagaimana data disampaikan ke end user. *Business logic layer* berfokus kepada validasi data, proses data secara dinamis untuk menghasilkan informasi. *Data access layer* berfokus pada keutuhan data, akses data melalui *API* dan penyimpanan data. Struktur *modern web* aplikasi terbagi menjadi *database, server logic, client logic* dan *client UI*. Pradigma yang mengimplementasikan model ini adalah *MVC architectural pattern*. Pada *MVC, logic*, data dan visualisasi di bagi menjadi tiga tipe objek yang dapat dilihat pada Gambar 2.1. *View* menangani bagian visual dan interaksi user. *Controller* merespon sistem, *user event*, dan memerintahkan model dan view jika adanya perubahan. *Model* menangani manipulasi data, merespon *request* untuk informasi atau perubahan *state* tergantung pada instruksi controller.

Gambar 2.1
MVC architecture communication



Sumber : MEAN Web Development (2014:8)

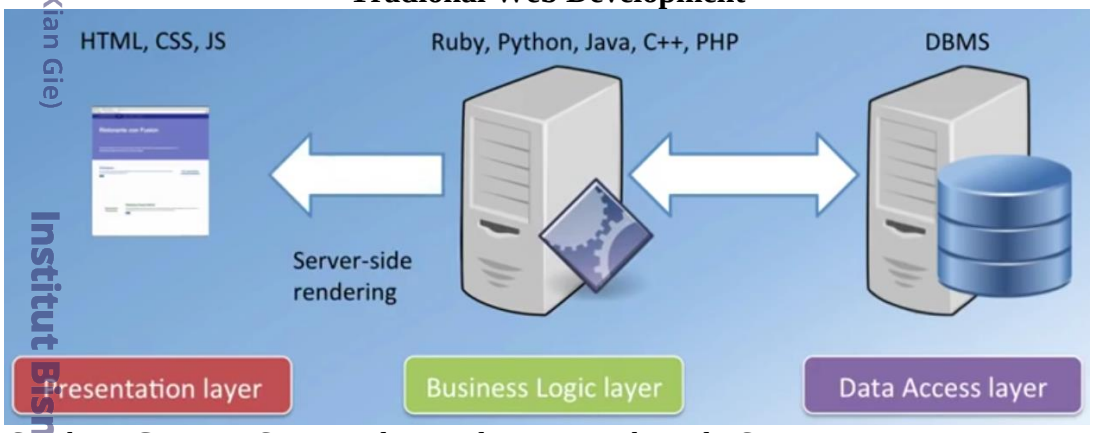
KWIK KIAN GIE SCHOOL OF BUSINESS
Hak Cipta Dilindungi Undang-Undang
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber:
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penyusunan laporan, penulisan kritik dan tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar IBIKKG.
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin IBIKKG.
© Hak cipta milik IBI KKG (Institut Bisnis dan Informatika Kwik Kian Gie)
Institut Bisnis dan Informatika Kwik Kian Gie

Menurut Frenando Monteiro(2014:11),"MVC merupakan pola architecture perangkat lunak yang telah dibuat sejak tahun 1980 yang mana memisahkan informasi visual representation dari interaksi user."

K. *Traditional Web Development*

Pendekatan *traditional web development* dilihat dari sisi *Three-tier web application development*, *Business logic layer* merupakan pada *server* yang berkomunakasi dengan *backend database management* sistem untuk bertukar data dan selanjutnya akan mengenerate *front end* untuk menampilkan data ke *user* dengan menggunakan *server side* untuk merender *HTML, CSS, javascript* dengan pendekatan ini, perlu dispesialisasikan dari setiap layer dimana *front end specialist* membuat tampilan dapat di *render* secara sempurna, *backend specialist* yang akan mengimplementasikan *server-side* dan *data specialist* yang akan berfokus pada data basis dan data persistent.

Gambar 2.2
Tradional Web Development



Sumber : Coursera Server-side Development with NodeJS

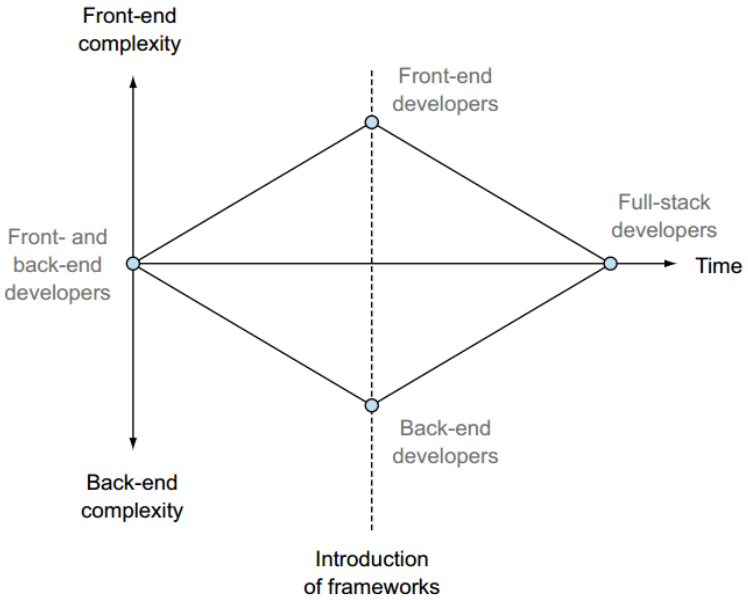
L. Teknologi *Full Stack*

Menurut Simon Holmes mengenai *full stack*(2016:3-5), *Full stack* merupakan teknologi yang melibatkan *database* dan *web server* pada *back end* yang berisi *application logic* dan kontrol di tengah serta user *interface* pada *front end. Back-end developer* berfokus pada mekanisme yang berjalan dibelakang, *front-end developers* berfokus pada membangun *user experience* yang baik. Selama tahun 2000-an *library*

dan *framework* mulai menjadi populer dan lazim untuk bahasa yang paling umum, pada kedua *front* dan *back end*. *Framework* ini dirancang untuk membuat hidup pengembang lebih mudah, menurunkan hambatan masuk. Sebuah *library* yang baik atau *framework* mengurangi beberapa kompleksitas dalam *development*, yang memungkinkan untuk *code* lebih cepat dan mengurangi keahlian mendalam dalam pembuatan. *Front end* menyediakan data ke *user* dalam bentuk tampilan dimana user akses *web* melalui *browser* yang mana menggunakan teknologi *HTML*, *CSS* dan *Javascript*. *Backend* lebih mengarah pada *business logic* dan data pada *web* aplikasi.

Menurut Simon Holmes (2016:6), Tren penyederhanaan telah menghasilkan munculnya *full stack developer* yang membangun kedua *front end* dan logika di balik aplikasi. Melanjutkan tren framework, beberapa tahun terakhir telah adanya peningkatan laju untuk memindahkan logika aplikasi dari *server* dan ke *front end*. Salah satu alasan menggunakan pendekatan ini yaitu mengurangi *load* pada *server*, sehingga mengurangi biaya dimana *resource* yang padat, komputasi diperlukan untuk aplikasi dengan mendorong beban ke *browser* pengguna. Simon Holmes pun menggambarkan arsitektur Full stack pada gambar 2.3

Gambar 2.3
Full Stack Architecture

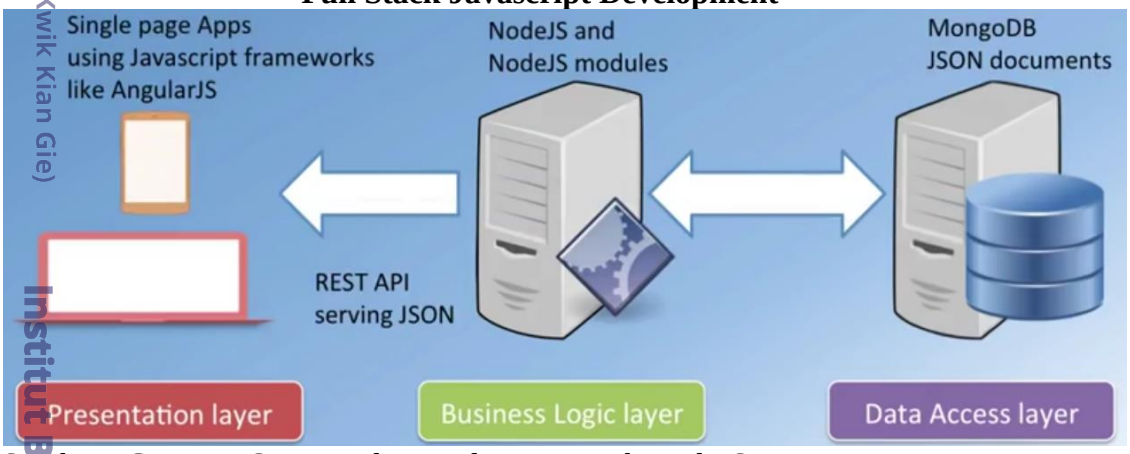


Sumber : Getting MEAN with Mongo, Express, Angular, and Node (2016:6)

KWIK KIAN GIE SCHOOL OF BUSINESS
Hak cipta milik IBIKKG (Institut Bisnis dan Informatika Kwik Kian Gie)
Hak Cipta Dilindungi Undang-Undang
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber:
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penyusunan laporan, penulisan kritik dan tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar IBIKKG.
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin IBIKKG.
Institut Bisnis dan Informatika Kwik Kian Gie

Terdapat kenaikan *trend* terhadap *full stack web development* dimana bahasa pemograman yang digunakan untuk implementasi untuk seluruh *stack* dimana anda dapat memiliki *front* sebagai *single page application* dengan menggunakan salah satu *javascript framework* seperti *angularjs*, *server-side* dapat diimplementasikan menggunakan teknologi seperti *Nodejs* dan *NodeJS module* serta *data access* atau *data storage* dapat diimplementasikan dengan teknologi seperti *MongoDB* yang mana menerima data dalam bentuk *JSON* dan mampu menyajikan informasi dalam bentuk *JSON data*. *JSON* telah menjadi standard format untuk pertukaran data diantara tiga *layer* seperti yang digambarkan pada Gambar 2.4. *Server side* sangat dipermudah untuk mengirimkan data melalui *REST API* sehingga mampu merencanakan *multiple platform* untuk tampilan seperti untuk *mobile device(hybrid)* dan *web development.*

Gambar 2.4
Full Stack Javascript Development



Sumber : Coursera Server-side Development with NodeJS

M. *REST API*

Menurut Simon Holmes(2016:25), *architecture full stack application* menggunakan *representational state transfer(REST) API* yang memberikan data pada *single page application*.

Menurut Valentin Bojinov (2015:iii), *RESTful services* telah menjadi penyedia data standard untuk *social services*, *news feeds*, dan *mobile devices*. Mereka memberikan sejumlah besar data ke jutaan pengguna. Dengan demikian, mereka perlu untuk mengatasi kebutuhan *high-availability*, seperti *reliability* dan *scalability*.

KWIK KIAN GIE SCHOOL OF BUSINESS
Hak cipta milik IBIKKG (Institut Bisnis dan Informatika Kwik Kian Gie)
Hak Cipta Dilindungi Undang-Undang
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber:
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penyusunan laporan, penulisan kritik dan tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar IBIKKG.
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin IBIKKG.
Institut Bisnis dan Informatika Kwik Kian Gie

KWIK KIAN GIE SCHOOL OF BUSINESS
Hak Cipta Dilindungi Undang-Undang
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber:
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penyusunan laporan, penulisan kritik dan tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar IBIKKG.
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin IBIKKG.
Hak cipta milik IBI KKG (Institut Bisnis dan Informatika Kwik Kian Gie)
Institut Bisnis dan Informatika Kwik Kian Gie

Menurut Valentin Bojinov (2015:2-7), Prinsip *HTTP* dan standar *URI*, yang akan membuat aplikasi *HTTP* menjadi *RESTful-service-enabled application*:

1. *Everything is a resource* - merepresentasikan data oleh format tertentu dan bukan dengan file fisik. Setiap bagian dari data yang tersedia di Internet memiliki format yang bisa digambarkan oleh jenis konten. Misalnya, Gambar *JPEG*; *video MPEG; html*, *xml*, dan teks dokumen; dan data biner semua sumber daya dengan jenis konten berikut*: image / jpeg*, *video / mpeg*, *text / html*, *text / xml*, *dan application / octet-stream.*
2. *Each resource is identifiable by a unique identifier (URI).*
   Sejak *internet* mengandung begitu banyak sumber daya yang berbeda, mereka semua harus dapat diakses melalui *URI* dan harus diidentifikasi secara unik. Selanjutnya, *URI* dapat berupa format *humanreadable*. *URI* menyimpan *data self-descriptive* dan memudahkan pengembangan lebih lanjut. Selain itu, dengan menggunakan *URI humanreadable* membantu *Developer* untuk mengurangi risiko kesalahan logis dalam program Anda seminimal mungkin. *URI humanreadable* mengekspos berbagai jenis sumber daya secara langsung
3. *Use the standard HTTP methods*

Tabel 2.1
Penggunaan standard HTTP method

HTTP verb	Action	Response status code
GET	Request an existing resource	"**200 OK**" if the resource exists, "**404 Not Found**" if it does not exist, and "**500 Internal Server Error**" for other errors
PUT	Create or update a resource	"**201 CREATED**" if a new resource is created, "**200 OK**" if updated, and "**500 Internal Server Error**" for other errors
POST	Update an existing resource	"**200 OK**" if the resource has been updated successfully, "**404 Not Found**" if the resource to be updated does not exist, and "**500 Internal Server Error**" for other errors
DELETE	Delete a resource	"**200 OK**" if the resource has been deleted successfully, "**404 Not Found**" if the resource to be deleted does not exist, and "**500 Internal Server Error**" for other errors

Sumber : RESTful Web API Design with Node.js(2015:4)

4. *Resources can have multiple representations*
   Fitur utama dari sumber daya adalah bahwa mereka dapat dipresentasikan dalam bentuk yang berbeda dari yang disimpan. Dengan demikian, dapat di*request* atau diposting dalam representasi yang berbeda. Selama format yang ditentukan didukung, *REST-enabled endpoint* harus digunakan.

5. *Communicate statelessly*

Operasi manipulasi sumber daya melalui *HTTP request* harus selalu dipertimbangkan *atomic*. Semua modifikasi dari sumber daya harus dilakukan dalam isolasi *HTTP request*. Setelah eksekusi *request*, sumber daya terdapat pada *final state*, yang secara implisit berarti bahwa pembaruan sumber daya parsial tidak didukung. Anda harus selalu mengirim *complete state* dari sumber daya.

Menurut Frenando Monteiro(2014:49), *Application Programming Interface (API)* merupakan suatu set *programming* standard untuk membangun aplikasi. *API* merupakan engine dari aplikasi yang mana berjalan dibelakang. *API* dapat terhubung pada beberapa sistem dan aplikasi lain. Dengan menggunakan *API*, aplikasi mampu berkomunikasi satu dengan lainnya tanpa adanya intervesi user. *API* bekerja melalui komunikasi antara kode yang berbeda, sehingga mendefinisikan specific behavior dari objek tertentu pada sebuah *interface*.

Menurut Caio Ribeiro Pereira(2016:15), Jenis aplikasi ini sedang dikembangkan oleh banyak proyek dan perusahaan, karena memiliki keuntungan dimana aplikasi terfokus hanya pada memberikan data pada aplikasi *client-side*. Saat ini, itu cukup umum untuk membuat *web* dan *mobile* aplikasi yang mengambil data lebih dari satu *API*. Ini berarti bahwa beberapa jenis aplikasi *client* terhubung dengan *server* yang sama berfokus pada berurusan dengan data. Selain itu juga memungkinkan untuk bekerja dengan oleh tim yang berbeda. Untuk mengembangkan API, programmer dapat menggunakan framework web yang sangat populer disebut Express

N. *Node.JS*

Menurut Caio Ribeiro Pereira(2016:1-3), Mengembangkan sistem untuk berbagai jenis perangkat perlu membangun suatu *web service*, juga dikenal dengan *application program interfaces(API)*. Pada dasarnya, *API* adalah *back-end* sistem yang dirancang untuk bekerja hanya dengan data secara terpusat, memungkinkan aplikasi *client-side* untuk dikembangkan secara terpisah, memiliki antarmuka yang unik untuk pengguna akhir. aplikasi *client-side* ini biasanya aplikasi *mobile*, *desktop*, atau aplikasi *web*.

KWIK KIAN GIE SCHOOL OF BUSINESS
Hak Cipta Dilindungi Undang-Undang
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber:
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penyusunan laporan, penulisan kritik dan tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar IBIKKG.
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin IBIKKG.
© Hak cipta milik IBI KKG (Institut Bisnis dan Informatika Kwik Kian Gie)
Institut Bisnis dan Informatika Kwik Kian Gie

Sejak 2010 *Node.js* telah semakin terbukti menjadi *platform* yang sangat baik untuk memecahkan banyak masalah, terutama untuk membangun *REST API*. Arsitektur *single-thread* yang melakukan *nonblocking I/O* yang berjalan di atas *JavaScript*, yang merupakan bahasa di mana-mana di semua *browser* saat ini, menunjukkan kinerja yang baik dalam pengolahan berbagai jenis aplikasi.

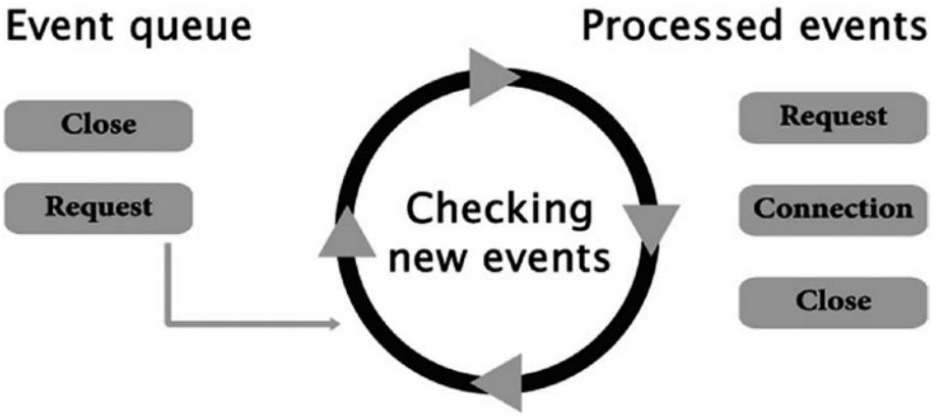
Single-Thread Arsitektur

Sebuah arsitektur *single-thread* bekerja dengan hanya *main thread instance* untuk setiap proses dimulai. Pada *multithread architecture*, seperti *Java* atau *.NET* menggunakan pengolahan *concurrency*. *Single-Thread* Arsitektur memungkinkan untuk bekerja dengan pemrosesan paralel menggunakan *multiple proces*

Event-Loop

Node.js memiliki paradigma *event-oriented*. Ini mengikuti filosofi *event oriented* yang sama dengan client side *JavaScript*. satu-satunya perbedaan adalah jenis *event*, yang berarti bahwa tidak ada *event* seperti *mouse click*, *key up* dari keyboard, atau event lain dari *web browser*. Proses Event loop digambarkan pada gambar 2.5

Gambar 2.5
Proses Event Loop



Sumber : Building APIs with Node.js(2016:3)

O. *Single Page Web Application*

Menurut Frenando Monteiro(2014:1),aplikasi *web* yang dinamis dan memiliki kemampuan *update* secara *real time* tanpa adanya *refresh* halaman, *scalable*, *high performance*, dan memiliki *interface* yang kaya, semakin mendominasi pasar untuk pengembangan web dengan cara baru untuk pengembangan isi web yang disebut *Single Page Web Application (SPA). Single Page Web Application* memberikan kemudahan

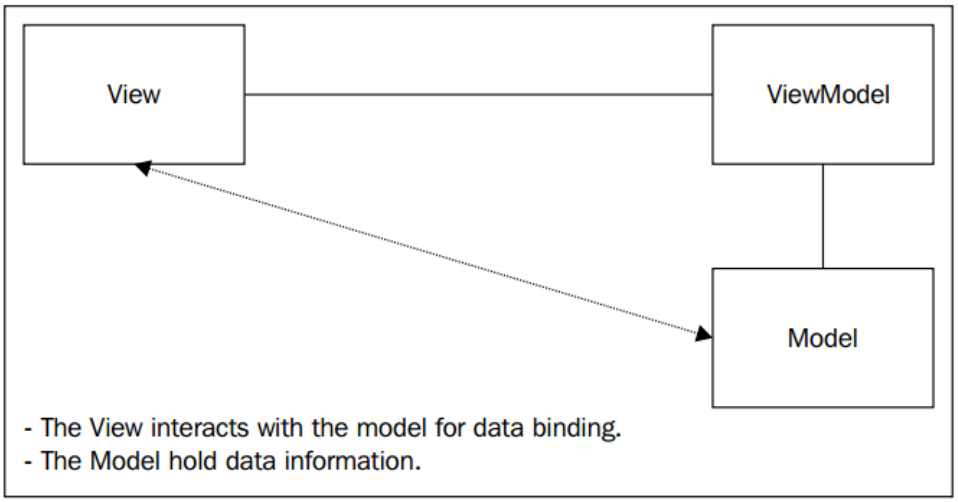
KWIK KIAN GIE SCHOOL OF BUSINESS
Hak cipta milik IBIKKG (Institut Bisnis dan Informatika Kwik Kian Gie)
Hak Cipta Dilindungi Undang-Undang
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber:
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penyusunan laporan, penulisan kritik dan tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar IBIKKG.
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin IBIKKG.
Institut Bisnis dan Informatika Kwik Kian Gie

dengan penulisan *server-side code* yang lebih sedikit dan lebih pada *client-side code* yang mana menyediakan user experience yang lebih baik dengan cara baru untuk berinteraksi dengan aplikasi. Menurut Frenando Monteiro (2014:10), dengan satu halaman tidak terbatas untuk satu file, kita dapat memiliki banyak template pada file yang berbeda. SPA mampu melakukan simple *create, read, update, delete (CRUD), to-do-list* sampai level yang lebih kompleks dengan *views, libraries, template, script,* dan *validation*. Tujuan utama SPA adalah mempu melakukan *update* pada bagian *interface* tanpa mengirimkan atau menerima *full-page request*. Semua data yang dibutuhkan, di inject pada page melalui *Ajax* dan *HTML template* untuk merender konten.

Menurut Frenando Monteiro (2014:12), *MVVM* memiliki abstraksi *view* yang berisi *view state* dan *behavior* dan memiliki *user interface development* yang lebih jelas dari pada *business logic* dan *behavior* pada aplikasi. Konsep MVVM pada gambar 2.6:

1. Model: *domain-spesific data* atau informasi pada aplikasi
2. View: bagian pada aplikasi dimana *user* dapat berinteraksi dan berisi *data binding, event* dan *behavior*. *View* merupakan *UI* yang interaktif yang menggambarkan *state* pada *ViewModel*
3. ViewModel: sering dikenal sebagai *controller* yang bekerja sebagai data converter. *ViewModel* mengubah informasi pada model ke informasi pada *view*, mengirimkan perintah dari view ke model.

Gambar 2.6
MVVM Architecture



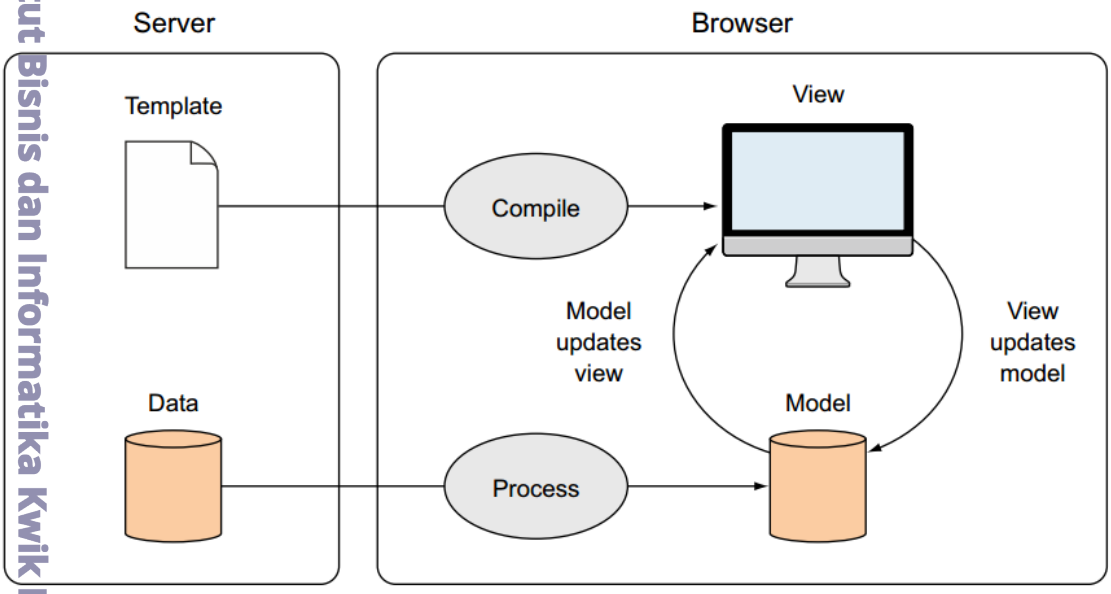
Sumber: Learning Single-page Web Application Development (2014:13)

KWIK KIAN GIE SCHOOL OF BUSINESS
Hak cipta milik IBIKKG (Institut Bisnis dan Informatika Kwik Kian Gie)
Hak Cipta Dilindungi Undang-Undang
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber:
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penyusunan laporan, penulisan kritik dan tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar IBIKKG.
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin IBIKKG.
Institut Bisnis dan Informatika Kwik Kian Gie

Konsep *data binding* disederhanakan sebagai proses yang membangun hubungan antara *UI* aplikasi dan logika bisnis. Hal ini juga dapat berarti bahwa ketika data berubah, data tersebut dapat merefleksikan perubahan tersebut. *Two-way data binding* merupakan kemampuan untuk membind data terhadap perubahan pada *UI*.

Menurut Simon Holmes (2016:16-17), *one-way model* adalah basis untuk kebanyakan *database-driven websites*. Pada model ini kebanyakan pekerjaan dilakukan pada *server*, dan hanya mempergunakan *browser* hanya untuk me*render HTML* dan menjalankan *JavaScript* secara interaktif. *Two-way data binding* berbeda. Pertama, *template* dan data dikirim secara independen ke *browser*. *Browser* itu sendiri mengkompilasi *template* ke tampilan dan data ke model. Perbedaan yang nyata adalah bahwa tampilan seperti "hidup". Pandangan diikatkan ke model, sehingga jika model berubah, tampilan dapat mengalami perubahan secara langsung dan model juga berubah. Data *store* diekspos melalui *API* dan tidak erat digabungkan ke aplikasi tersebut, biasanya terdapat beberapa pengolahan yang terlibat sebelum menambahkannya ke model. *Developer* harus memastikan bahwa mengikat data yang relevan dan benar ke tampilan. Two ways data binding diilustrasikan pada gambar 2.7

Gambar 2.7
Two Ways Data Binding



Sumber: Getting MEAN with Mongo, Express, Angular, and Node (2016:16-17)

KWIK KIAN GIE SCHOOL OF BUSINESS
Hak cipta milik IBIKKG (Institut Bisnis dan Informatika Kwik Kian Gie)
Hak Cipta Dilindungi Undang-Undang
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber:
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penyusunan laporan, penulisan kritik dan tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar IBIKKG.
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin IBIKKG.
Institut Bisnis dan Informatika Kwik Kian Gie

KWIK KIAN GIE
SCHOOL OF BUSINESS
Hak Cipta Dilindungi Undang-Undang
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber:
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penyusunan laporan, penulisan kritik dan tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar IBIKKG.
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin IBIKKG.
© Hak cipta milik IBI KKG (Institut Bisnis dan Informatika Kwik Kian Gie)
Institut Bisnis dan Informatika Kwik Kian Gie

P. PostgreSQL

Menurut David Bryant Copeland (2016: v-ix), *PostgreSQL* merupakan *open source SQL database* yang dirilis pada tahun 1997. *PostgreSQL* mendukung banyak fitur advance yang tidak ditemukan pada open source database lain seperti *MySQL* atau database komersial seperti *Microsoft SQL Server*. Fitur:

1. *Check constraints*: *Developer* dapat membuat *constraint* yang sangat kompleks pada kolom tabel seperti yang didapatkan dengan tidak *null*. *PostgreSQL* mampu melakukan pengecekan pada *database layer* yang berarti tidak ada *bug* dalam *code*, tidak ada *script*, tidak ada *developer* di *console*, dan tidak ada program di masa mendatang yang dapat memasukan data yang buruk ke dalam *database*. Integritas data seperti tidak dapat dilakuakn dengan *Middleware* saja.
2. *Advanced indexing*: Dalam banyak sistem *database*, *Developer* hanya dapat *indeks value* dalam kolom *database*. *Postgres* dapat melakukan indeks *transformed values.*
3. *Materialized views*: Sebuah tampilan *database* adalah tabel logis berdasarkan *SELECT statement.* Dalam *Postgres materialized views* adalah tampilan yang isinya disimpan dalam *actual tabel* - mengakses *materialized views* tidak dapat menjalankan *query* lagi seperti pada tampilan normal.
4. *Advanced data types*

   Postgres memiliki dukungan untuk tipe *enumerated*, *array*, dan *dictionaries (*disebut *HSTOREs).* Pada kebanyakan sistem database harus menggunakan tabel terpisah untuk model struktur data ini.
5. *Free-form JSON*

   Postgres mendukung tipe data *JSON*, yang memungkinkan untuk menyimpan data dalam kolom. Ini berarti dapat menggunakan *Postgres* sebagai *data store* dokumen, atau untuk menyimpan data yang tidak sesuai secara kuat dengan skema dan, dengan menggunakan tipe data *JSONB*, dapat memastikan bahwa *JSON* dapat diindeks, seperti *structured table fields*. Meskipun dapat melakukan *serialize hash* untuk *JSON* pada *middleware* menggunakan tipe data *TEXT*, namun tidak dapat melakuakan *query*, dan tidak dapat melakukan indeks. Field *JSONB* dapat beroperasi dengan banyak sistem, dan mereka memberikan kinerja yang baik.

ORDBMS *(Object Relational Database Management Sistem)* merupakan *RDBMS* dengan tambahan fitur object dengan menitikberatkan pada extensibilitas dan

penyesuaian standard. *Database server* memiliki fungsi utama untuk menyimpan data secara aman dan mengembalikan data di *response* dari *request* dan mampu menangani *workload single machine application* sampai *internet facing application(data warehouse)* dengan banyak pengguna secara berbarengan.

Multiversion concurrency control (MVCC)

PostgreSQL mengelola concurrency melalui sistem yang dikenal *Multiversion concurrency control (MVCC)*, yang memberikan setiap transaksi sebuah "*snapshot*" dari database, yang memungkinkan perubahan harus dibuat tanpa terlihat transaksi lainnya sampai perubahan di*commit*.

Q. Metodologi Pengembangan Perangkat Lunak

Agile Development Method adalah sekelompok metodologi pengembangan perangkat lunak yang didasarkan pada prinsip-prinsip yang sama atau pengembangan sistem jangka pendek yang memerlukan adaptasi cepat dari pengembang terhadap perubahan dalam bentuk apapun. Agile memiliki pengertian bersifat cepat, ringan, bebas bergerak, dan waspada. Dengan prinsip-prinsip tersebut *agile* berusaha untuk menyiasati tiga masalah yang biasanya dihadapi saat proses pembuatan perangkat lunak, yaitu:

1. Kebutuhan perangkat lunak sulit diprediksi dari awal dan selalu akan berubah. Selain itu, prioritas klien juga sering berubah seiring berjalannya proyek.
2. Desain dan pembangunan sering tumpang tindih. Sulit diperkirakan seberapa jauh desain yang diperlukan sebelum pembangunan.
3. Analisis, desain, pembangunan dan testing tidak dapat diperkirakan

Tujuan *agile*:

1. *High-value & working App system*, diharapkan dengan memakai *agile development methods* dapat dihasilkan perangkat lunak yang mempunyai nilai jual yang tinggi, biaya pembuatan bisa di tekan dan perangkat lunak bisa berjalan dengan baik.
2. *Iterative, incremental, evolutionary, agile* adalah metode pengembangan perangkat lunak yang iteratif, selalu mengalami perubahan, dan evolusioner. Tim harus

KWIK KIAN GIE SCHOOL OF BUSINESS
Hak Cipta Dilindungi Undang-Undang
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber:
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penyusunan laporan, penulisan kritik dan tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar IBIKKG.
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin IBIKKG.
© Hak cipta milik IBIKKG (Institut Bisnis dan Informatika Kwik Kian Gie)
Institut Bisnis dan Informatika Kwik Kian Gie

bekerja dalam waktu yang singkat dan selalu menambah fungsionalitas dari perangkat lunak sesuai dengan kebutuhan klien.

3. *Cost control & value-driven development*, salah satu tujuan dari agile yaitu pengembangan perangkat lunak disesuaikan dengan kebutuhan pengguna, tim bisa dengan cepat merespon kebutuhan yang diinginkan pengguna sehingga waktu dan biaya pembuatan perangkat lunak bisa dikontrol.
4. *High-quality production*, walaupun biaya pembuatan perangkat lunak bisa ditekan dan proses pembuatan bisa dipercepat, tetapi kualitas dari perangkat lunak yang dibuat harus tetap dijaga. Dengan melakukan tes setiap fungsionalitas perangkat lunak setelah selesei dibuat berarti agile juga mengakomodir kebutuhan ini.
5. *Flexible & risk management*, Mulai dari pertemuan dengan sistem analis untuk mengubah sistem perangkat lunak, perubahan rencana rilis produk hingga perubahan biaya produksi. Pertemuan dengan klien untuk melakukan tes perangkat lunak juga sering dilakukan sehingga fungsionalitas perangkat lunak mudah diubah dan akhirnya kegagalan perangkat lunakpun bisa diminimalisir.
6. *Collaboration*, dengan menggunakan *agile*, tim pengembang diharuskan sering bertemu untuk membahas perkembangan proyek dan *feedback* dari klien yang nantinya akan ditambahkan dalam perangkat lunak, sehingga tim bisa berkolaborasi dengan maksimal.

Peneliti menjabarkan kelebihan dari macam - macam model agile yang paling popular pada table 2.2

Tabel 2.2
Macam Model Agile yang paling populer

Unified Process	Extreme Programming	Scrum
Mempunyai waktu pengembangan yang panjang	Keperluan berubah dengan cepat	Keperluan berubah dengan cepat
Menyediakan akses yang mudah terhadap pengetahuan dasar bagi anggota tim	Biaya lebih murah	Pekerjaan terbagi-bagi sehingga dapat diselesaikan dengan cepat
Mempunyai tim programmer yang cukup banyak	Tim programmer sedikit, yaitu 2-10 orang	Tim programmer sedikit, yaitu 5-9 orang
Pengembangan perangkat lunak yang berorientasi objek dengan berfokus pada UML	Mampu mengotomatiskan tes	Dokumentasi dan pengujian terus menerus dilakukan setelah software dibangun
Pengembangan dan perubahan perangkat lunak berdasarkan kebutuhan user	Banyak ide baru dan setiap kali kesalahan ditemukan, langsung diperbaiki.	Proses menghasilkan beberapa software increment

Sumber : ilustrasi peneliti

KWIK KIAN GIE SCHOOL OF BUSINESS
Hak Cipta Dilindungi Undang-Undang
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber:
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penyusunan laporan, penulisan kritik dan tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar IBIKKG.
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin IBIKKG.
© Hak cipta milik IBI KKG (Institut Bisnis dan Informatika Kwik Kian Gie)
Institut Bisnis dan Informatika Kwik Kian Gie

KWIK KIAN GIE
SCHOOL OF BUSINESS

R. UML

Unified Modeling Language (UML) adalah bahasa spesifikasi standar untuk mendokumentasikan, menspesifikasikan, dan membangun sistem perangkat lunak.

UML adalah suatu bahasa yang digunakan untuk menentukan, memvisualisasikan, membangun, dan mendokumentasikan suatu sistem informasi. UML dikembangkan sebagai suatu alat untuk analisis dan desain berorientasi objek oleh Grady Booch, Jim Rumbaugh, dan Ivar Jacobson. Namun demikian UML dapat digunakan untuk memahami dan mendokumentasikan setiap sistem informasi. Penggunaan UML dalam industri terus meningkat. Ini merupakan standar terbuka yang menjadikannya sebagai bahasa pemodelan yang umum dalam industri peranti lunak dan pengembangan sistem.

UML diagram dipresentasikan dalam 2 bentuk model sistem:

1. Static (or structural) view: menunjukan struktur sistem secara statik menggunakan objek, attribute, operasi dan relasi. Yang termasuk pada structural UML diagram adalah Class diagram, component diagram, composite structure diagram, deployment diagram, object diagram, package diagram, profile diagram
2. Dynamic (or behavioral) view: menunjukan dinamik behavior sistem dengan menunjukan kolaborasi antara objek dan perubahan state internal pada objek. Yang termausk pada Behaviour UML diagram adlaah activity diagram, communication diagram, interaction overview diagram, sequence diagram, state diaghram, timing diagram, use case diagram.

Pengertian use case diagram menurut Satzinger, Jackson dan Burd (2009:242), Diagram untuk menunjukkan peran dari berbagai pengguna dan bagaimana peran-peran menggunakan sistem". Use case diagram tidak menjelaskan secara detil, tetapi hanya memberi gambaran singkat hubungan antara usecase, aktor, dan sistem. Didalam use case ini akan diketahui fungsi - fungsi yang berada pada sistem yang dibuat.

Menurut Whitten L. Jeffery et al (2004:432) class diagram adalah gambar grafis mengenai struktur objek statis dari suatu sistem, menunjukan class-class objek yang menyusun sebuah sistem dan juga hubungan antara class objek tersebut.

Hak cipta milik IBIKKG (Institut Bisnis dan Informatika Kwik Kian Gie)
Institut Bisnis dan Informatika Kwik Kian Gie
Hak Cipta Dilindungi Undang-Undang
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber:
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penyusunan laporan, penulisan kritik dan tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar IBIKKG.
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin IBIKKG.

S. Delapan Aturan Emas (Eight Golden Rules)

Dalam mendesain user interface yang baik, Ben Shneiderman dan Catherine Plaisant (2005:74) memberikan delapan aturan yang dapat dijadikan pedoman:

1. Konsistensi

   Dalam situasi apapun urutan tindakannya harus konsisten. Selain itu warna, font, dan tata letak juga harus konsisten.

2. Memenuhi kebutuhan-kebutuhan umum

   Mengenali kebutuhan pengguna yang mempunyai beragam desain, mulai dari yang pemula sampai yang sudah ahli.

3. Memberikan umpan balik yang informatif

   Untuk setiap tindakan dari pengguna harus ada umpan balik, untuk tindakan yang sering dan tindakan yang tidak penting harus ada umpan balik yang sederhana, dan untuk setiap tindakan yang penting harus ada umpan balik yang substansial.

4. Mendesain dialog untuk menghasilkan suatu penutupan

   Urutan tindakan harus diorganisir pada suatu kelompok dengan bagian awal, tengah, dan akhir serta memberikan umpan balik yang informatif pada setiap tindakan yang dilakukan oleh pengguna.

5. Mencegah terjadinya kesalahan

   Sistem sebaiknya didesain sebagaimana mungkin untuk menghindari kesalahan fatal yang dilakukan pengguna. Jika terjadi kesalahan, sistem dapat memberikan penanganan yang sederhana dan sistem dapat dengan sederhana mendeteksi kesalahan.

6. Mudah kembali ke tindakan sebelumnya

   Sistem dapat kembali ke tindakan sebelumnya. Misalnya pengguna dapat membatalkan tindakan yang sudah dilakukannya sehingga pengguna tidak perlu takut untuk mengeksplorasi pilihan-pilihan yang tersedia.

7. Mendukung tempat pengendali internal (internal locus of control)

   Sistem harus dapat mendukung penggunanya dalam hal mengontrol sistem itu sendiri sehingga pengguna dapat merasakan bahwa mereka yang mengontrol sistem.

8. Mengurangi ingatan jangka pendek

   Keterbatasan ingatan pada manusia membutuhkan tampilan yang sederhana, penyatuan banyak tampilan, pengurangan frekuensi window motion, dan memberikan cukup waktu untuk pelatihan.

KWIK KIAN GIE SCHOOL OF BUSINESS
Hak cipta milik IBI KKG (Institut Bisnis dan Informatika Kwik Kian Gie)
Institut Bisnis dan Informatika Kwik Kian Gie
Hak Cipta Dilindungi Undang-Undang
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber:
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penyusunan laporan, penulisan kritik dan tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar IBIKKG.
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin IBIKKG.

KWIK KIAN GIE
SCHOOL OF BUSINESS

T. ERD

Menurut Mata-Toledo dan Cushman (2007: 139), Entity Relationship Diagram (ERD) merupakan representasi grafis dari logika database dengan menyertakan deskripsi detail mengenai seluruh entitas (entity), hubungan (relationship), dan batasan (constraint).

U. Penelitian Sebelumnya

Terdapat beberapa penelitian yang dijadikan dasar penelitian ini. Sebagai fondasi dasar geknologi yang digunakan dikutip berdasarkan jurnal yang dibuat oleh Bahubali Akiwate, Ayazahmed Patel, Tasleem Nabiwale, Namita Naik, Suraj Patil yang berjudul "*Web Based Student Information Management System using MEAN Stack*" yang mana menggunakan teknologi *MEAN(MongoDB, ExpressJS, AngularJS and Node.JS) Stack*.

Pada jurnal tersebut disebutkan keuntungan menggunakan teknologi ini (2016:2) :

1. Sistem *user friendly*.
2. Menyediakan "*better and efficient*" *service* ke user.
3. Mengurangi *work load user*.
4. Semua detail akan tersedian dengan hanya *single click.*
5. Semua bagian pemprosesan selesai pada *client side* karenanya, proses pengambilan data cepat.

Kesimpulan dari jurnal tersebut (2016:5) adalah pendekatan ini mempresentasikan efesiensi sistem informasi manajemen siswa dengan *MEAN Stack* teknologi menggunakan bahasa *javascript* pada *client* dan *server side*. Dan semua komputasi diproses pada *client side.* Dengan demikian sistem ini membantu mengurangi pekerjaan secara manual sehingga kurangnya tenaga kerja yang diperlukan.

Selain itu, peneliti juga mengambil referesi dari jurnal Mrudula Deore, Mayuri Kambli, Chinmayi Kulkarni, Asst. Prof. Sunil Chaudhari yang berjudul "*Modern Web Apps using Full Stack Development and Containerization*" yang menggambarkan aplikasi web yang dibuat menggunakan teknologi *MEAN Stack, Docker, Amazon S3* dan *Amazon EC2.*

Hak Cipta Dilindungi Undang-Undang
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber:
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penyusunan laporan, penulisan kritik dan tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar IBIKKG.
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin IBIKKG.
© Hak cipta milik IBIKKG (Institut Bisnis dan Informatika Kwik Kian Gie)
Institut Bisnis dan Informatika Kwik Kian Gie

Pada jurnal ini (2016:6) menyatakan bahwa *Full Stack Development* menggunakan *MEAN Stack* bersifat kuat, fleksibel, *scalable* dan merupakan solusi aplikasi web modern. Ini menyediakan isolasi yang baik antara 4 teknologi dengan menyajikannya dalam struktur folder hirarki seperti *MEAN.JS* dan *MEAN.IO*. *Full Stack Development* tidak membatasi pengembang di satu atau beberapa lapisan pengembangan perangkat lunak, tetapi memungkinkan pengembang untuk mengeksplorasi semua lapisan *stack* dengan kekuatan hanya satu bahasa pemrograman yaitu *JavaScript*.

Jurnal yang berjudul "*Comparative Performance Analysis of PostgreSQL High Availability Database Clusters through Containment*" yang dibuat oleh Le Quan Ha, Jeff Xie, Darrell Millington dan Amgad Waniss mempengaruhi pendapat peneliti dalam pemilihan database yang menjelaskan analisis performa *PostgreSQL*.

Peneliti terinspirasi dalam pemilihan judul skripsi ini berdasarkan masalah yang telah dijelaskan pada latar belakang masalah dan jurnal yang berjudul "*Data Management and Data Administration : Assessing 25 Years of Practice*" yang dibuat oleh Peter Aiken, Mark L, Xihui Zhang, David Rafner. Jurnal ini mengkomparasi aspek *Data Management (DM),* mendapatkan data menggunakan *set subject matter expert*. Menggunakan informasi ini untuk mengamati evolusi profesi. Analisis menunjukkan keberhasilan investasi dalam tantangan DM terhadap IT dan organisasi.

KWIK KIAN GIE SCHOOL OF BUSINESS
Hak Cipta Dilindungi Undang-Undang
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber:
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penyusunan laporan, penulisan kritik dan tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar IBIKKG.
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin IBIKKG.
© Hak cipta milik IBI KKG (Institut Bisnis dan Informatika Kwik Kian Gie)
Institut Bisnis dan Informatika Kwik Kian Gie