BAB 2
TINJAUAN PUSTAKA

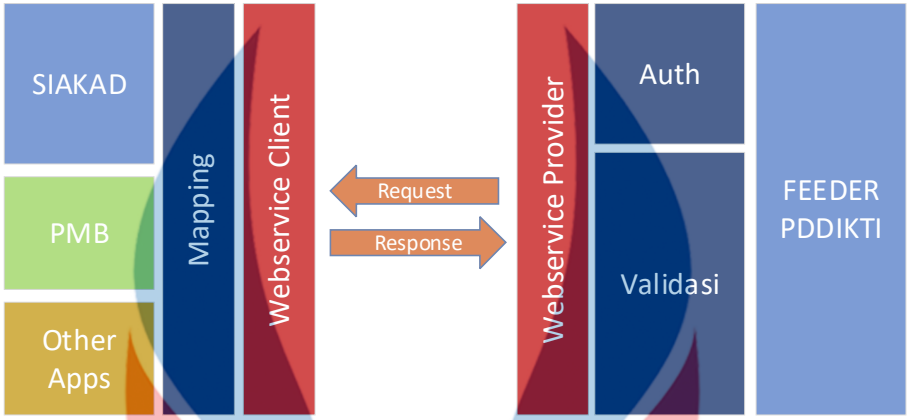
Bab ini menjelaskan teori-teori yang menjadi dasar dalam melakukan penelitian, yang bersumber dari buku, jurnal ataupun artikel. Tujuannya agar konsep dan teori-teori penyelesaian masalah yang digunakan dapat dipahami.

2.1 Pangkalan Data Pendidikan Tinggi

Pangkalan Data Pendidikan Tinggi (PD DIKTI) merupakan kumpulan informasi penyelenggaraan Pendidikan Tinggi (DIKTI) seluruh universitas secara nasional. PD DIKTI dikembangkan dan dioperasikan oleh Kementerian atau dikelola oleh lembaga yang telah ditunjuk dari Kementerian. Saat ini PD DIKTI telah menjadi salah satu sarana pelaksanaan penjaminan mutu (Suwarni, 2018). Dalam undang-undang No. 12 Tahun 2012 pasal 56 tentang Pendidikan Tinggi menyebutkan bahwa PD DIKTI sebagai halnya yang dimaksud pada ayat (1) berfungsi sebagai akar dari informasi bagi lembaga akreditasi, untuk melakukan akreditasi Program Studi dan Perguruan Tinggi (DIRJEN DIKTI, 2012). Kemudian untuk pemerintah memiliki fungsi untuk melakukan pengaturan, perencanaan pengawasan, pemantauan dan evaluasi serta pembinaan dan koordinasi program studi dan Perguruan Tinggi. Kemudian untuk masyarakat memiliki fungsi untuk mengetahui kinerja program studi dan Perguruan Tinggi. Dengan adanya peraturan di atas setiap Perguruan Tinggi wajib melaporkan serta mengintegrasikan data proses belajar mengajar setiap tahunnya kepada DIKTI. Oleh karena itu DIKTI menyediakan sebuah aplikasi yang bernama *Feeder* PD DIKTI yang digunakan untuk mengelola data Mahasiswa dan data Perkuliahan masing-masing Perguruan Tinggi yang selanjutnya diintegrasikan ke *database* yang dimiliki oleh DIKTI (Suwarni, 2018).

Feeder PD DIKTI adalah aplikasi yang berfungsi untuk mengatur data Mahasiswa dan data perkuliahan masing-masing Perguruan Tinggi. Aplikasi *Feeder* PD DIKTI dapat melakukan proses sinkronasi data antara yang ada pada aplikasi *Feeder* dengan yang ada di aplikasi yang dikelola DIKTI yaitu aplikasi

online report (Forlap). *Feeder* PD DIKTI dapat dikelola sendiri oleh masing-masing Perguruan Tinggi (Suwarni, 2018). PD DIKTI juga menyediakan *web service* yang dapat digunakan oleh setiap Perguruan Tinggi.



Gambar 2.1 Skema *Feeder* PD DIKTI (DIKTI, 2017)

Pada gambar 2.1 di atas menjelaskan bagaimana skema dari *Feeder* PD DIKTI, dimana PD DIKTI menyediakan *service* yang dapat dimanfaatkan oleh Perguruan Tinggi agar dapat saling terhubung antara *Feeder* PD DIKTI dengan sistem informasi yang telah ada di Perguruan Tinggi masing-masing. Sumber data yang dibutuhkan untuk memenuhi kebutuhan PD DIKTI bisa berasal dari sistem informasi dan data tersebut perlu dilakukan *mapping* terlebih dahulu untuk disesuaikan dengan standar yang sudah ditentukan oleh PD DIKTI (DIKTI, 2017).

2.2 *Data Mapper*

Data Mapper adalah *layer* dari perangkat lunak yang memisahkan antara objek dalam memori dari *database* (Fowler, 2003). Fungsinya adalah untuk melakukan transfer data antara keduanya dan juga memisahkan dari satu sama lain. Dengan menggunakan *Data Mapper*, objek yang berada dalam memori tidak perlu mengetahui bahwa ada sebuah *database* dan skema *database* tidak perlu mengetahui bahwa terdapat objek yang menggunakannya. Dalam rekayasa perangkat lunak, pola *Data Mapper* termasuk dalam *architectural pattern* (Fowler, 2003). Pada penelitian ini, *Data Mapper* akan digunakan sebagai metode dalam dalam pengambilan data yang terdapat di *database* Sikantan ITK.

Tabel 2.1 *Mapping* data

Tabel PD DIKTI	Tabel Sikantan
ajar_dosen	t_jadwal_detil_dosen, t_jadwal_kuliah
bobot_nilai	m_grade_nilai
daya_tampung	
dosen	m_dosen
dosen_pt	m_dosen
dosen_pembimbing	
kelas_kuliah	m_sesi_kuliah, t_jadwal_detil_sesi, t_jadwal_kuliah
kuliah_mahasiswa	m_sesi_kuliah, t_jadwal_detil_sesi, t_jadwal_kuliah
kurikulum	m_tahun_kurikulum
mahasiswa	m_mahasiswa
mahasiswa_pt	m_mahasiswa
mata_kuliah	m_mata_kuliah
nilai	t_detil_nilai, t_jenis_nilai
nilai_transfer	
satuan_pendidikan	
sms	m_jurusan

Pada tabel 2.1 adalah tabel *mapping* data dari tabel yang ada dari *database feeder* dan *database* Sikantan. Pada kolom tabel PD DIKTI merupakan nama-nama tabel dari *database* PD DIKTI. Pada kolom tabel Sikantan merupakan nama-nama tabel dari *database* Sikantan. Pada *database* Sikantan masih memiliki kekurangan data yang dibutuhkan oleh *database Feeder* seperti tabel daya_tampung, dosen_pembimbing, nilai_transfer dan satuan_pendidikan. Oleh karena itu aplikasi yang akan dikembangkan nantinya memiliki fungsi *import* untuk melengkapi data yang dibutuhkan sesuai *database Feeder.*

2.3 *Agile Methods*

Agile Methods adalah sebuah metode pengembangan aplikasi. Penerapan metode *Agile* memeliki kelebihan seperti, menghemat waktu dan biaya. Karena penerapan metode *Agile* memerlukan dokumentasi yang lebih sedikit. Oleh karena itu pengembang dapat lebih fokus dalam mengembangkan aplikasi daripada membuat dokumentasi. Metode *Agile* memberikan tahapan yang terbuka, jadi *customer* dapat melihat hasil yang lebih baik dari kondisi yang sedang

dikembangkan oleh pihak pengembang secara langsung. Dengan memanfaatkan metode *Agile*, tim pengembang mengalami proses ringan yang mendukung fokus pada penyampaian proses bisnis. Oleh karena itu pengembangan menggunakan metode *Agile* dapat menguraing resiko yang signifikan dan mengoptimalkan *value* bisnis. Dengan menyesuaikan aplikasi yang disampaikan berdasarkan kebutuhan bisnis, sehingga tim pengembang dapat dengan mudah jika ada perubahan kebutuhan pada proyek (Shankarmani, 2012). Berikut prinsip-prinsip metode *Agile* pada tabel 2.2.

Tabel 2.2 *The Principles of Agile Methods* (Sommerville, 2011)

Prinsip	Deskripsi
Customer involvement	Klien harus terlibat dalam proses pengembangan. Peran dari klien adalah menyediakan dan memprioritaskan persyaratan sistem baru dan untuk mengevaluasi dari setiap iterasi sistem.
Incremental delivery	Perangkat lunak ini dikembangkan secara bertahap dengan pelanggan yang menetapkan persyaratan untuk dimasukkan dalam setiap *increment*.
People not process	Keterampilan tim pengembangan harus diakui dan dieksploitasi. Anggota tim harus dibiarkan untuk mengembangkan cara kerja mereka sendiri tanpa proses preskriptif.
Embrace change	Tim pengembang harus selalu bersiap-siap dengan segala perubahan yang diberikan oleh pelanggan.
Maintain simplicity	Fokus pada kesederhanaan dalam perangkat lunak yang sedang dikembangkan dan dalam proses pengembangan. Sebisa mungkin menghilangkan kompleksitas dari sistem.

Pada tabel 2.2 di atas dijelaskan tentang prinsip-prinsip dari metode *Agile*. Pada kolom *principle* merupakan beberapa prinsip yang harus diterapkan dalam penggunaan metode *Agile* yaitu *Customer involvement*, *Incremental delivery*, *People not process*, *Embrace change* dan *Maintain simplicity*. Pada kolom

description merupakan penjelasan dari masing-masing dari prinsip yang harus diterapkan. Beberapa contoh yang model proses dari *Agile Development* adalah *SCRUM*, *Dynamic Systems Development Method* (DSDM), *Adaptive Software development* (ASD) dan *Extreme Programming* (XP) (Sommerville, 2011).

Pada penelitian ini, *Agile Methods* akan digunakan segabai metode dalam pengembangan aplikasi, karena dapat menghemat waktu dan pengembang dapat lebih fokus untuk mengembangkan aplikasi tersebut.

2.4 *Scrum*

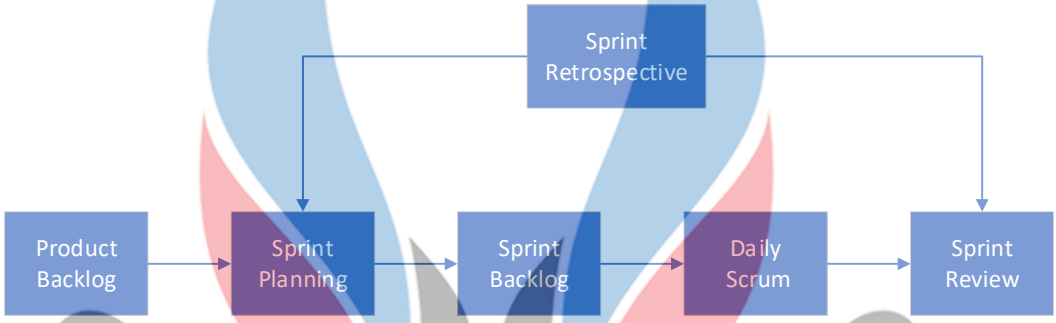
Scrum adalah sebuah kerangka kerja yang bertujuan untuk mengembangkan, menyampaikan dan mengelola produk yang kompleks. *Scrum* terbukti efektif dalam transfer pengetahuan secara berkala dan berkelanjutan. *Scrum* saat ini sudah digunakan secara luas untuk produk-produk, layanan-layanan dan manajemen perusahaan induk. *Scrum* menggunakan pendekatan yang bertahap dan berkelanjutan untuk mengoptimalkan kemampuan memprediksi dan memanajemen resiko. (Schwaber & Sutherland, 2017).

Penggunaan metode *Scrum* dalam pengembangan aplikasi harus memiliki *Scrum team. Scrum team* terdiri dari *product owner*, *development team* dan *Scrum master*. *Scrum Team* terdiri dari orang yang memiliki kemampuannya dan perannya masing-masing. *Scrum Team* dirancang untuk meningkatkan fleksibilitas, i dan produktifitas (Schwaber & Sutherland, 2017).

Product owner adalah orang yang bertanggung jawab untuk memastikan dari produk yang dihasilkan oleh *development team*. *Product owner* juga bertanggung jawab dalam pengelolaan *product backlog,* dimana *product backlog* merupakan daftar dari tugas-tugas yang akan dikerjakan selama *Sprint* berlangsung. Selanjutnya ada *development team* yang terdiri dari para ahli profesi yang berguna untuk menyelesaikan *backlog* yang akan dipindahkan ke *increment* ”selesai” yang berpotensi dirilis di setiap akhir *Sprint*. *Development team* dibentuk dan diberikan wewenang oleh organisasi untuk menyusun dan mengelola pekerjaan mereka sendiri. Terakhir ada *Scrum master* yang bertanggung jawab untuk mengenalkan dan membantu dalam penggunaan *Scrum*. *Scrum master* adalah pemimpin dalam *Scrum team*. *Scrum master* membantu orang-orang di luar *Scrum team* untuk dapat

memahami interaksi mana yang bermanfaat dan tidak bermanfaat. (Schwaber & Sutherland, 2017).

Scrum memiliki aktivitas yang wajib dilakukan yaitu wajib melakukan *Sprint*. *Sprint* adalah jantung dari *Scrum*, *Sprint* memiliki sebuah batasan waktu atau durasi yang ditentukan oleh *Scrum Team*. *Sprint* memiliki *event-event* seperti *Sprint Planning*, *Daily Scrum*, *Sprint Review* dan *Sprint Retrospective*. Alur dari *Sprint* dapat dilihat pada gambar 2.2 (Schwaber & Sutherland, 2017).



Gambar 2.2 Alur dari *Sprint* (Flewelling, 2018)

Pada gambar 2.2 alur pertama yang dilakukan ketika melakukan *Sprint* adalah melakukan *Sprint Planning*. *Sprint Planning* merancang apa yang akan dilakukan ketika *Sprint* dimulai, perancangan didapatkan dari *product backlog* yang telah dibuat dan *Sprint Planning* menghasilkan *Sprint backlog*. Perancangan ini dihadiri oleh seluruh anggota *Scrum team* (Schwaber & Sutherland, 2017).

Selanjutnya ada *Daily Scrum* yaitu pertemuan yang dilakukan setiap hari selama *Sprint* berjalan. *Scrum team* akan berkumpul dan mendiskusikan apa yang telah dicapai dari *goal* yang ditentukan sejak terakhir melakukan *Daily Scrum*, apa yang harus dikerjakan hingga melakukan *Daily Scrum* lagi dan apakah ada masalah atau hambatan saat pengembangan.

Selanjutnya ada *Sprint Review* adalah pertemuan yang diadakan di akhir siklus *Sprint*. *Sprint Review* dihadiri seluruh *Scrum team* dan *stakeholders* dari produk yang dibuat. *Product owner* akan menjelaskan *user story* mana yang "selesai" dan "belum". *Sprint Review* juga mendapatkan *feedback* dari seluruh *team* dan *stakeholders* yang kemudian *product owner* dapat menyusun *backlog* kembali untuk *Sprint* selanjutnya.

Terakhir ada *Sprint Retrospective* yang biasanya berjalan setelah melakukan *Sprint Review* dan ini adalah pertemuan terakhir dari iterasi. *Sprint Retrospective*

adalah kesempatan *Scrum team* untuk memeriksa dan menyesuaikan prosesnya. Secara umum, *Scrum* team akan membahas apa yang berjalan dengan baik, apa yang tidak berjalan dengan tidak baik dan apa yang harus diperbaiki. Hasil dari pertemuan ini adalah menghasilkan perbaikan bagi team yang dapat dilakukan saat *Sprint* selanjutnya dilaksanakan (Flewelling, 2018).

Increment merupakan pengelompokan untuk *Product Backlog item*, yang telah dikerjakan dalam *Sprint* dan total dari *Increment* dari setiap *Sprint* sebelumnya. *Increment* adalah langkah untuk mendapatkan tujuan atau visi dari pengembangan produk. Jika *Increment* dinyatakan "selesai", maka *Increment* dapat digunakan dan berfungsi (Schwaber & Sutherland, 2017).

Pada penelitian ini, *Framework Scrum* akan digunakan untuk melakukan proses pengembangan aplikasi, untuk membantu proses dokumentasi dari pengembangan aplikasi.

2.5 Node.js

Node.js atau bisa disebut node merupakan sebuah *platform* yang ada di aplikasi JavaScript. Node.js dijalankan dari sisi server. Node.js dikembangkan dari *engine* JavaScript yang dibuat oleh Google untuk *browser* web. Node.js dapat digunakan untuk server web yang ringan dan cepat dan ideal untuk membangun sebuah *web-service* API (Poulter, et al., 2015).

2.6 Express.js

Express.js adalah salah satu *framework* paling popular di Node js. Dokumentasi express.js yang lengkap sehingga penggunaannya yang mudah, express.js juga dapat mengembangkan berbagai produk seperti aplikasi web ataupun *RESTful API* (Poulter, et al., 2015).

2.7 Vue.js

Framework VueJs atau bisa disebut dengan Vue, merupakan *progresif framework* untuk membangun antarmuka pengguna atau *user interfaces*. *Core library* dari Vue fokus pada *view layer* atau pada antar muka saja. Vue juga dapat menjalankan aplikasi *single-page* yang canggih dengan lancar bila menggunakan

dengan *tools* dan *library* yang dapat membantu dalam pembuatan suatu proyek yang sedang dikerjakan (Gore, 2017).

Sejak rilis pertamanya, Vue sudah menjadi salah satu proyek *open source* paling popular yang digunakan saat ini dan banyak pro di GitHub. Popularitas ini didapatkan karena Vue memiliki fungsi – fungsi yang kuat, tetapi juga mudah untuk digunakan (Gore, 2017).

2.8 Penelitian Terdahulu

Berikut ini merupakan penelitian terdahulu yang pernah dilakukan dan berkaitan dengan penelitian yang akan dilakukan, dapat dilihat pada tabel 2.2.

Tabel 2.3 Penelitian Terdahulu

No	Nama dan Tahun Publikasi	Penelitian yang Dilakukan
1	Slamet Widodo, 2018	Masalah: Data akademik yang banyak dan bervariasi menjadi sebuah kendala dalam proses pelaporan. Kendala tersebut meliputi waktu proses yang cukup lama dan rentan mengalami kesalahan dalam pemasukan data ke aplikasi *Feeder* PDDIKTI. Metode: Melakukan proses *maping* data atau menyesuaikan struktrut *database* yang dibutuhkan Feeder, kemudian menggunakan teknologi *web service* untuk melakukan *request* dan *response*. Hasil: penelitian ini yaitu mengembangkan aplikasi *web service* yang terdiri dari tiga modul utama yang dapat mengakomodir kebutuhan-kebutuhan pelaporan data akademik yang sudah berjalan ke DIKTI
2	Rifki Indra Perwira, 2017	Masalah: Data akademik yang sangat banyak dan bermacam menjadikan sebuah kendala. Adanya perbedaan struktur *database* antara milik Universitas Pembangunan Nasional "Veteran" Yogyakarta dengan yang dimiliki DIKTI Metode: Melakukan proses *maping* terlebih dahulu sebelum dimasukkan pada *database Feeder* agar data sesuai dengan format yang dibutuhkan oleh PD DIKTI. Penelitian ini memanfaatkan modul *web service*. Hasil: Mengembangkan *tools web service* yang dapat mengakomodir proses pelaporan data akademik ke DIKTI.
3	Agus Siswanto, 2016	Masalah: Data atau proses kegiatan akademik yang sebelumnya telah di*input* oleh pihak BAAK dan diserahkan kepada pihak operator PDPT. Kemudian pihak operator PDPT akan meng*input*kan kembali data tersebut ke aplikasi Feeder PDDIKTI.

No	Nama dan Tahun Publikasi	Penelitian yang Dilakukan
		Metode: Pengembangan *web service* untuk mengintegrasikan *database*. Hasil: pengembangan aplikasi sinkronasi *database* antara sistem informasi akademik STIKOM Dinamika Bangsa Jambi dengan *Feeder* PDDIKTI dengan teknologi *web service*.
4	M. Agung Sutrisno, 2016	Masalah: Sistem Informasi Akademik (SIAKAD) yang dimiliki UIM belum memiliki fungsi untuk mengintegrasikan data, sehingga cara manual masih digunakan untuk saat ini untuk melakukan pelaporan data akademik ke PDDIKTI. Metode: menggunakan *web service* dan *Framework Laravel* untuk kerangka kerja pemprograman PHP. Hasil: SIAKAD yang dapat terintegrasi dengan data *Feeder* PD DIKTI dengan memanfaatkan *web service* dan *Framework Laravel*

Pada tabel 2.2 di atas dijelaskan tentang beberapa hasil penelitian terdahulu yang memiliki keterkaitan dengan penelitian yang akan dilakukan. Maka dapat diketahui aplikasi *Feeder* PD DIKTI yang sudah disediakan oleh DIKTI memiliki beberapa kendala seperti, pertama memiliki perbedaan struktur *database* aplikasi *Feeder* antara *database* yang dimiliki Perguruan Tinggi, kedua saat ini aplikasi *Feeder* tidak memiliki fungsi *import* data, ketiga data akademik yang sangat banyak dan bermacam. Sehingga dalam proses peng*input*an data akademik kedalam aplikasi *Feeder* PD DIKTI memerlukan waktu yang lama. Berdasarkan penelitian terdahulu yang telah dilakukan dapat disimpulkan bahwa Perguruan Tinggi dapat mengembangkan aplikasi *Feeder* internal yang dapat terintegrasi dengan data akademik yang dimiliki oleh Perguruan Tinggi, dengan cara melakukan proses *mapping* antara data yang dimiliki Perguruan Tinggi dengan data *Feeder* PD DIKTI. Berdasarkan permasalahan yang ada di ITK dalam penelitian ini peneliti mengembangkan aplikasi *Feeder* PD DIKTI internal yang dapat secara otomatis melakukan *mapper data* Sikantan ke PD DIKTI.