BAB III
LANDASAN TEORI

Pada bab ini akan di jelaskan mengenai teori dasar yang digunakan oleh penulis sebagi acuan dalam membangun aplikasi.

3.1 Sistem Informasi Berbasis Mobile

Perkembangan *smartphone android* sangat berperan dalam dunia pendidikan seperti hadirnya media pembelajaran berbasis *mobile*, sistem informasi akademik berbasis *mobile* dan masih banyak lagi. Jika beberapa tahun lalu sistem informasi akademik terbatas dalam bentuk *website*, kini sistem informasi akademik juga sudah mulai dikembangkan pada perangkat *mobile* berbasis *android*. Hal ini sekarang menjadi hal yang umum karena terdapat segudang kelebihan yang dimiliki oleh aplikasi berbasis *mobile*. Dengan menggunakan sistem informasi akademik berbasis *mobile*, pengguna aplikasi dapat menggali informasi dimana saja dan kapan saja seperti Jadwal Kuliah, Pembayaran SPP Online (tidak perlu antri) mengisi KRS (Kartu Rencana Studi), melihat KHS (Kartu Hasil Studi), Perwalian, Diskusi, pengumuman nilai dan masih banyak lagi, semua itu bisa dilakukan lewat *smartphone*.

Keunggulan utama dalam mengembangkan Sistem Informasi berbasis *mobile* adalah dalam hal mobilitas. Aplikasi pada *smartphone* dapat digunakan dimana saja dan kapan saja. Berkat karakteristik tersebut, Sistem Informasi berbasis *mobile* menjadi jawaban yang tepat untuk mengatasi kendala pada sistem informasi akademik

berbasis web ataupun desktop. Dengan aplikasi sistem informasi akademik berbasis *mobile*, pengguna sistem informasi dapat mengakses informasi langsung dari *smartphone* mereka. Disamping dari sisi mobilitas, pengembangan sistem informasi akademik berbasis *mobile* juga menjadi kian optimal berkat adanya fitur-fitur khas *smartphone* seperti fitur notifikasi. Dengan fitur notifikasi maka informasi akan lebih cepat tersalurkan ke pengguna aplikasi. Tentu penggunaan aplikasi sistem informasi akademik berbasis *mobile* lebih efektif jika dibandingkan dengan menggunakan sistem informasi akademik web *based* yang harus menunggu pengguna untuk mengakses web.

Pada implementasi sistem informasi *mobile* seringkali diperlukan integrasi dengan website ataupun sistem lain. Misal jika sudah ada website kemahasiswaan maka aplikasi *mobile* akan bekerja lebih efektif jika saling terintegrasi dengan website tersebut. Dari pihak kampus juga akan lebih mudah mengolah data mahasiswa jika menggunakan aplikasi sistem informasi akademik berbasis *mobile* yang terintegrasi dengan aplikasi pengolahan data kemahasiswaan.

3.2 Aplikasi *Mobile*

Aplikasi *mobile* adalah sebuah sistem perangkat lunak yang memungkinkan setiap pemakai melakukan mobilitas pada telepon genggam atau seluler. *Android* dan *iOS* merupakan sistem operasi *mobile* yang sangat populer untuk saat ini .Aplikasi *mobile* merupakan peningkatan dari sistem perangkat lunak terpadu yang

ada pada PC Desktop. Aplikasi *mobile* itu sendiri dibagi menjadi 3 jenis yaitu:

1. *Native App Native* adalah sebuah aplikasi yang ditujukan pada perangkat *mobile Android*, *iOS*, *Symbian*, *Windows Phone*, *WebOS* dan lainnya. Aplikasi ini dirancang dengan menggunakan bahasa pemrograman seperti *Objective-C* atau Swift dan *Java*, kemudian juga alat *Software Development Kit* (SDK) yang ditanam (*install*) langsung kedalam perangkat *mobile*. Aplikasi *Native* bisa berfungsi baik untuk anda yang menginginkan penggunaan GPS, kamera, mikrofon, kompas, daftar kontak, *swipe gestures* (layar sensor) dan sebagainya dimana juga bisa aktif ketika koneksi internet tiada karena memiliki sistem notifikasi pada *platform*. Aplikasi ini bisa anda dapati di *Google Play* atau *App Store*.
2. *Mobile Web App* Aplikasi ini sedikit berbeda dengan *Native* apps, yaitu pada bahasa pemrograman dan hanya dapat diakses ketika koneksi internet aktif. Aplikasi ini juga tidak tersedia di *Google Play* dan *App Store*. Bahasa pemograman yang digunakan pada aplikasi *mobile website* adalah *html5, javascript*, dan *Active Server Pages* (ASP) ditambahkan komponen seperti *php, database, CSS (Cascading Style Sheets)* ataupun lain sebagainya.
3. *Hybrid App Aplikasi Hybrid* adalah sebuah aplikasi yang juga populer pada perkembangan *mobile*. Komponen aplikasi ini terdiri dari kedua perangkat aplikasi sistem yang telah disebutkan sebelumnya yaitu *Native app* dan *Mobile Web App,* contohnya

html5, *css*, *javascript* dan *php* beserta alat lain seperti *Apache Cordova*, *Rubymotion*, *Appcelerator Titanium* yang akan dibungkus menjadi satu aplikasi *mobile* bernama *Hybrid* (Teo, 2017).

3.3 Reminder

Reminder merupakan sebuah fitur pesan pengingat yang dapat membantu setiap orang mengingatkan akan sesuatu, yang biasanya terpasang di ponsel maupun media pencatatan lainnya. Berbeda dari alarm yang hanya berbunyi pada saat tertentu, *reminder* dapat diatur di waktu yang ditentukan serta menampilkan sebuah pesan yang telah ditulis sebelumnya. *Reminder* adalah sesuatu yang digunakan untuk memberikan peringatan pada hari atau waktu tertentu bahwa ada kegiatan yang harus dilakukan. *Reminder* selalu berkaitan erat dengan suatu *alarm* atau janji. *Alarm* digunakan untuk memberikan sebuah peringatan kepada seorang pengguna bahwa akan ada kegiatan pada saat yang telah ditentukan jauh sebelum sebuah *alarm* berbunyi. Sebelum melakukan pengaturan *reminder*, biasanya di buat dahulu sebuah jadwal. Pengertian jadwal berdasarkan KBBI adalah pembagian waktu yang berdasarkan suatu rencana pengaturan urutan kerja, daftar kegiatan maupun rencana kegiatan dengan pembagian waktu pelaksanaan yang terperinci. Sedangkan arti dari penjadwalan merupakan proses, cara, pembuatan jadwal atau memasukkan rencana kegiatan ke dalam sebuah jadwal. Sedangkan cara kerja dari sistem *reminder* adalah server selalu mencocokkan jam yang sudah disimpan di dalam *database* dengan jam pada sistem operasi server.

Aplikasi *reminder* merupakan aplikasi yang dapat memunculkan notifikasi dan bunyi dari perangkat *mobile* yang berfungsi sebagai pengingat suatu jadwal atau agenda. Secara umum, aplikasi *reminder* biasanya di-set oleh pengguna berdasarkan waktu kemunculan *reminder*. Notifikasi dapat muncul pada jam ataupun hari tertentu sesuai dengan agenda yang diinputkan oleh pengguna. Hampir di seluruh perangkat mobile terdapat aplikasi *reminder* semacam itu, dikarenakan memang sudah banyak pengguna yang merasakan manfaat aplikasi tersebut. Seorang pengguna bisa meminimalisir adanya suatu agenda yang terlewati dan bisa lebih disiplin dengan waktu dengan menggunakan *reminder.*

3.4 Firebase

Firebase merupakan sebuah *Backend as a Service* yaitu layanan *backend* yang sekarang dikembangkan oleh *Google. Firebase* adalah sebuah solusi dan alternatif yang diberikan oleh *Google* agar memudahkan kerja para pengembang *Mobile Aplication.* Dengan menggunakan *Firebase* diharapkan para pengembang aplikasi *mobile* hanya fokus untuk mengembangkan sebuah aplikasi yang handal, dan tanpa harus memikirkan infrastruktur, keamanan, kehandalan, dan perawatan *backend* dan server. *Firebase* pertama kali didirikan pada tahun 2011 oleh Andrew Lee dan James Tamplin. Produk yang pertama kali dikembangkan adalah *Realtime Database,* di mana *developer* dapat menyimpan dan melakukan sinkronasi data ke banyak user. Kemudian berkembang menjadi layanan penyedia pengembangan aplikasi. Pada Oktober 2014, perusahaan tersebut diakusisi oleh *Google.* Berbagai fitur terus dikembangkan hingga diperkenalkan pada Mei

2016 di *Google I/O.* Berikut ini fitur-fitur yang tersedia dilayanan *Firebase* :

a. *Firebase Analytics*

Fitur ini digunakan untuk analisis koleksi data dan laporan untuk aplikasi *Android* dan *IOS.* Selain itu dengan *Firebase Analytics* juga dapat melihat fungsi atau *part* dari aplikasi mana yang banyak dan sering dipakai atau diakses pengguna. Yang paling menarik dari *Firebase Analytics* adalah fitur ini dapat membuat atau melakukan segmentasi dari pengguna berdasarkan *User Attribute.* Definisi dari *User Attribute* merupakan parameter yang digunakan sebagai penyaring laporan dan juga notifikasi.

b. *Firebase Authentication*

Fitur ini adalah layanan dari *Firebase* yang digunakan untuk fungsi user *membership.* Fitur-fitur yang diberikan adalah registrasi dan *login* dengan beberapa metode email dan *password*, akun *Google*, akun *Facebook*, akun *Twitter*, akun *GitHub*, dan metode *login Anonymous.*

c. *Firebase Realtime Database*

Realtime Database adalah *database* bertipe *NoSQL*, *NoSQL database* adalah *database* yang tidak menggunakan sistem relasi seperti *database* SQL. Metode penyimpanan data pada *NoSQL* menggunakan format *JavaScript Object*

Notation (JSON). Firebase memungkinkan untuk menggunakan *database* yang dishare kepada semua user, dan ketika terjadi perubahan pada *database*, user akan segera mendapatkan *update* data secara *realtime.* Selain itu aplikasi juga bisa menyimpan data secara local saat tidak memiliki koneksi internet, kemudian melakukan *synchronize* data segera setelah koneksi internet kembali normal.

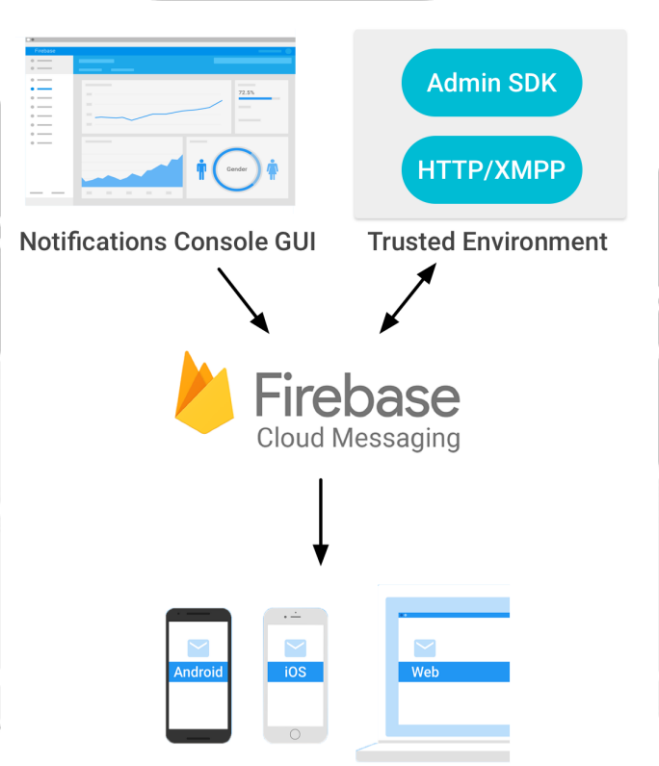
d. *Firebase Crash Reporting*

Crash Reporting merupakan layanan yang disediakan oleh *Firebase* yang digunakan untuk keperluan merekam setiap *exception* yang terjadi di aplikasi. *Report* yang diberikan cukup detail dengan beberapa filter seperti yang digunakan pada *Analytics. Crash Reporting* ini terbagi atas dua bagian yaitu *Non Fatal exception,* untuk *exception* yang tidak berdampak fatal dan *Fatal exception,* untuk *exception* yang fatal (aplikasi *crash).*

e. *Firebase Cloud Messaging*

Firebase Cloud Messaging atau FCM merupakan layanan dari *Firebase* untuk menggantikan *Google Cloud Messaging* atau bisa disebut GCM. Fitur-fitur yang dimiliki oleh *Firebase Cloud Messaging* sebenarnya tidak jauh berbeda dengan *Google Cloud Messaging.* FCM dapat mengirim *push notification* dan membuat komunikasi dua arah antar *device.* Teknologi yang dipakai ada dua yaitu XMPP (*Extensible Messaging and Presence*

Protocol) dan HTTP (*Hypertext Transfer Protocol).*



Gambar 3.1 Arsitektur *Firebase Cloud Messaging* (sumber:*firebase.google.com)*

3.5 Web Service

Web Service (WS) adalah aplikasi perangkat lunak yang diidentifikasi oleh URI yang antar muka dan pengikatannya dapat didefinisikan, dijelaskan, dan ditemukan oleh XML, artefak dan mendukung interaksi langsung dengan aplikasi perangkat lunak lain yang menggunakan pesan berbasis XML melalui protokol berbasis internet. Menurut Kiet (2013), WS merupakan paradigma teknologi, proses dan perangkat lunak yang

memberikan dukungan untuk integrasi bisnis terutama melalui lingkungan berbasis internet. Standar WS bergantung pada standar lain yaitu, *Simple Object Access Protocol* (SOAP), *Web Service Description Language* (WSDL) dan *Universal Description Discovery and Integration* (UDDI) berungsi secara efisien (Kiet T. Tran, 2013).

Didalam *Web Server* terdapat *Web Service*, *Web service* merupakan kumpulan fungsi dan method yang terdapat pada sebuah *web server* yang dapat dipanggil oleh klien dari jarak jauh, kemudian untuk memangil method-method tersebut dapat digunakan aplikasi yang dibuat dengan bahasa pemrograman apa saja yang dijalankan pada platform apa saja (Piedro, 2016). *Web service* memiliki hasil yang dapat berbentuk XML atau dalam bentuk JSON. Kedua hasil *web service* tersebut dinamakan dengan teknik SOAP dan REST.

3.6 RESTful

REST *(REpresentational State Transfer)* merupakan standar arsitektur komunikasi berbasis web yang sering diterapkan dalam pengembangan layanan berbasis web. Umumnya menggunakan HTTP *(Hypertext Transfer Protocol)* sebagai protocol untuk komunikasi data. REST pertama kali diperkenalkan oleh Roy Fielding pada tahun 2000. Pada arsitektur REST, REST *server* menyediakan *resources* (sumber daya/data) dan REST *client* mengakses dan menampilkan *resource* tersebut untuk penggunaan selanjutnya. Setiap *resource* diidentifikasi oleh URIs (*Universal Resource Identifiers*) atau global ID. *Resource* tersebut direpresentasikan dalam bentuk format

teks, JSON atau XML. Pada umumnya formatnya menggunakan JSON dan XML. Berikut ini metode HTTP yang umum digunakan dalam arsitektur berbasis Rest.

Tabel 3.2 Metode HTTP dan penggunaannya dalam REST (Riyadi,2013)

Metode	Deskripsi
GET	Mendapatkan (read) sebuah sumber daya (resource) yang diidentifikasi dengan URI (Uniform Resource Identifier).
POST	Mengirimkan sumber daya (resource) ke server. Digunakan untuk membuat (create) sumber daya baru.
PUT	Mengirimkan sumber daya (resource) ke server. Digunakan untuk memasukkan (insert) atau memperbaharui (update) sumber daya yang tersimpan.
DELETE	Menghapus sumber daya (resource) yang diidentifikasi oleh URI.
HEAD	Mendapatkan meta data (response header) dari sumber daya (resource) yang diidentifikasi oleh URI.

3.7 Xamarin

Xamarin adalah sebuah *software development tool* yang bisa digunakan untuk membangun aplikasi *mobile* berbasis *android*, *ios* dan *windows phone*. Xamarin awalnya adalah perusahaan *software* yang terletak di San Francisco, california dan berdiri semenjak Mei 2011. *Engginer* yang bergerak di perusahaan ini adalah yang membuat *Mono*, *MonoTouch* dan *Mono for Android yang mana cross-platform* dan dapat di implementasikan ke *Common*

Language Infrastructure (CLI) dan *Common Language Specifications* (atau yang biasa disebut Microsoft .NET). Dengan based nya adalah bahasa pemograman C# dan mengimplemetasikan metode *shared codebase*, *developers* bisa menggunakan Xamarin untuk menggembangkan aplikasi *native iOS, Android*, and *Windows* dengan *native* user *interfaces* dan *share code across multiple platforms*. Pada awalnya Xamarin dirilis sebagai *software* berbayar. Baru pada tahun 2013, sejak bergabung dengan Visual Studio, Xamarin pun mengeluarkan versi gratisnya – tepatnya dibundle dengan Visual Studio.

Xamarin memiliki banyak fitur-fitur dan keunggulan, dari banyak fitur dan keunggulan tersebut berikut beberapa diantaranya:

1. ***Cross-platform Development*** yaitu dengan mengandalkan bahasa pemograman C# kita sudah bisa membuat dan mengembangkan aplikasi di banyak platform seperti *iOS, Android, Mac* dan *Windows*.

2. ***Visual Studio Integration*** yaitu Xamarin sudah bisa terintegrasi dengan Visual Studio. Jadi dengan menggunakan Visual Studio kita sudah bisa menanamkan Xamarin dan mulai membuat aplikasi-aplikasi *Mobile* yang kita inginkan.

3. ***Native UI, Native Performance*** yaitu Xamarin memberikan sebuah performa tinggi dalam mengcompile kode dan memberikan akses penuh ke semua *Native API*, jadi kita bisa membuat aplikasi native sesuai dengan perangkat yang spesifik.

4. ***Reuse Existing Code*** yaitu kita bisa menggunakan .NET *library* dan juga dengan mudah menggunakan *library-library* atau *framework* lain dalam pembuatan atau membangun aplikasi kalian di Xamarin.

5. ***Fully Featured IDE*** yaitu jika kita tidak biasa menggunakan Visual Studio, Xamarin juga menyediakan full fitur IDE yang dikhususkan untuk membuat aplikasi *Mobile*. Didalam IDE ini sudah ada fitur seperti *Code Completion*, *integrated designer*, *debugger* dan lain-lain. Nama dari IDE ini adalah Xamarin Studio.

6. **Point and Click UI Design** yaitu Xamarin menyediakan *Android UI Designer* kelas dunia. Dan juga menggunakan *Apple Xcode UI* Designer untuk membuat interface dan storyboard yang secara otomatis tersinkron dengan Xamarin.iOS projek kita.

Selain fitur-fitur yang disediakan Xamarin diatas, masih ada banyak keunggulan lain dari Xamarin yaitu membuat aplikasi *Native* dengan hanya bahasa C#. semua yang bisa dilakukan dengan Java dan *Objective* C bisa dilakukan C# di Xamarin. Dengan Xamarin studio kita bisa dengan mudah dalam membangun aplikasi Mobile. Xamarin adalah satu-satunya *platform* yang bisa memperbolehkan kita membangun atau membuat aplikasi Native iOS dan Android dari Visual Studio. Xamarin selalu *Update* API-API terbaru dari *Google* (*Android*) dan *Apple*(iOS). Menyediakan banyak dokumentasi, tutorial, *guide* dan *support* yang membantu user dalam mengembangkan aplikasinya dengan Xamarin.