BAB II
TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

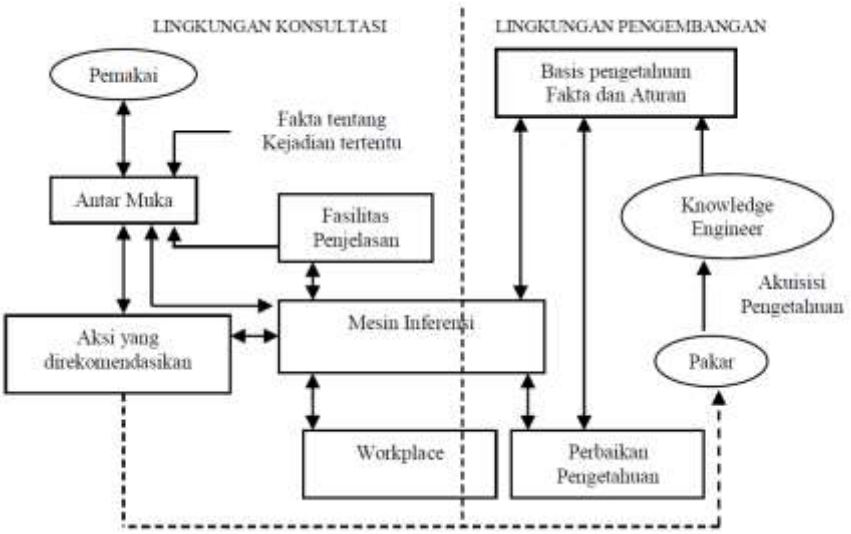
Penelitian terdahulu menjadi salah satu acuan dalam melakukan penelitian sehingga dapat memperkaya teori yang digunakan dalam mengkaji penelitian yang dilakukan. Berikut merupakan penelitian terdahulu berupa jurnal terkait dengan penelitian yang dilakukan ini.

- Penelitian yang dilakukan oleh Alfiansyah dan Rintana Arnie, 2016, yang berjudul Sistem pakar diagnosa penyakit umum dan P3K menggunakan metode forward chaining berbasis android. Berdasarkan hasil uji *pretest* keakuratan dalam mendiagnosa sebesar 20% sedangkan setelah ada sistem (*protest*) keakuratan sistem yang dibangun adalah sebesar 97%. Jadi dapat disimpulkan bahwa dengan adanya sistem aplikasi yang dibangun untuk mendiagnosa penyakit umum dan informasi P3K dengan gejala – gejala yang dialami lebih akurat berdasarkan data yang diuji yang terdapat pada data pengujian (Alfiansyah dan Rintana, 2016).
- Penelitian yang dilakukan oleh Tati Harihayati dan Luthfi Kurnia, 2012, yang berjudul Sistem pakar mendiagnosa penyakit umum yang sering diderita balita berbasis web si Dinas Kesehatan Kota Bandung. Berdasarkan hasil penelitian sistem yang dibangun kurang lebih 60% sudah dapat memudahkan masyarakat untuk mendapatkan informasi mengenai penyakit balita dan gejalanya dengan cara pendiagnosaan. Sistem yang dibangun dapat mengurangi resiko kesalahan yang dilakukan orang tua dalam melakukan pertolongan pertama kepada balitanya yang terindikasi penyakit dan keterlambatan dalam penanganan medis (Harihayati dan Kurnia 2012).
- Penelitian yang dilakukan oleh Mardi Turnip, 2015, yang berjudul Sistem pakar diagnosa penyakit THT menggunakan metode backward chaining. Berdasarkan hasil penelitian bahwa sistem pakar dapat membantu dokter THT untuk melakukan diagnosa. Berdasarkan rata – rata efisiensi didapatkan hasil rata – rata efisiensi proses manual adalah 21 menit 67 detik sedangkan dengan efisiensi sistem pakar didapatkan hasil 9 menit 40 detik (Mardi Turnip 2015).
- Penelitian yang dilakukan oleh Nurmala Mukhtar dan Samsudin, 2015, yang berjudul Sistem pakar diagnosa penyakit THT menggunakan metode backward chaining. Berdasarkan hasil penelitian bahwa sistem pakar ini akan mempermudah orang awam untuk melakukan diagnosa dampak *softlens* dan cara penanggulangannya (Nurmala Mukhtar dan Samsudin 2015.

2.2 Struktur Sistem Pakar

Ada dua bagian utama yang dibutuhkan dalam membuat apliaksi kecerdasan buatan yaitu basis pengetahuan (*knowledge base*) dan mesin inferensi (*Inference Engine*) (Puspita dkk, 2013). Lingkungan pengembangan dimasukkan untuk pengembangan pakar ke dalam lingkungan sistem pakar, sedangkan lingkungan konsultasi digunakan oleh pengguna yang bukan pakar guna memperoleh pengetahuan pakar (Lempao, 2011).

Istilah sistem pakar berasal dari istilah *knowledge-based expert system*. Istilah ini muncul karena untuk memecahkan masalah, sistem pakar menggunakan pengetahuan seorang pakar uang dimasukkan ke dalam komputer. Seseorang yang bukan pakar menggunakan sistem pakar untuk meningkatkan kemampuan pemecahan masalah, sedangkan seorang pakar yang menggunakan sistem pakar untuk *knowledge assistant* (Sutojo dkk, 2010). Tujuan utama sistem pakar bukan untuk menggantikan kedudukan seorang ahli maupun pakar, tetapi untuk memasyarakatkan pengetahuan dan pengalaman pakar – pakar yang ahli di bidangnya (Saputra, 2011). Sistem pakar disusun oleh dua bagian utama, yaitu lingkungan pengembangan (*development environment*) dan lingkungan konsultasi (*consultation environment*). Komponen – komponen sistem pakar (Rachmawati dkk, 2012) dapat dilihat pada Gambar 2.1



Gambar 2. 1 *struktur sistem pakar*

2.3 Representasi Pengetahuan Sistem Pakar

Terdapat beberapa teknik representasi pengetahuan yang biasa digunakan dalam pengambangan suatu sistem pakar yaitu: (1) *rule based knowledge*, (2) *frame-based*

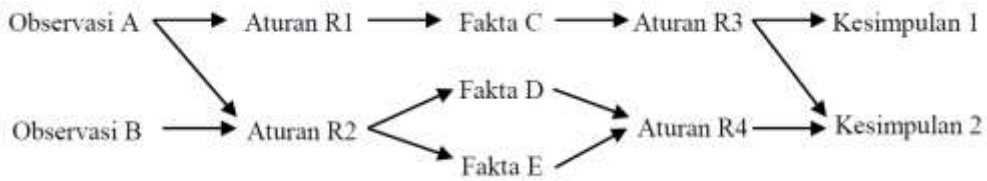
knowledge, (3) *object-based knowledge* dan (4) *case-base reasoning* (Ramadhan, 2011). Salah satu metode yang paling umum untuk merepresentasikan pengetahuan adalah dalam bentuk tipe aturan (*rule*) (Maradesa, 2012).

Knowledge Base (Basis Pengetahuan) merupakan hasil akuisisi dan representasi pengetahuan dari seorang pakar. Basis pengetahuan yang umum digunakan, yaitu: (1) menggunakan aturan berbentuk: *If-Then*. Penalaran ini digunakan jika terdapat sejumlah pengetahuan pakar pada suatu permasalahan tertentu, dan pakar dapat melakukan penyelesaian secara berurutan. (2) Penalaran Berbasis Kasus (*Cased Based Reasoning*) yaitu basis pengetahuan akan berisi solusi – solusi yang telah dicapai sebelumnya, kemudian diturunkan suatu solusi untuk keadaan yang terjadi sekarang (Hartati dan Iswanti, 2008).

Inference Engine (Mesin Inferensi) mengarahkan pencarian melalui basis pengetahuan, proses yang akan dilibatkan aplikasi aturan inferensi disebut pencocokan pola. Program diagnosa memutuskan aturan mana yang diinvestigasi, diagnosa yang mana yang dieliminasi dan atribut mana yang disesuaikan (Fadhilah dkk, 2012).

2.4 Mesin Inferensi Sistem Pakar

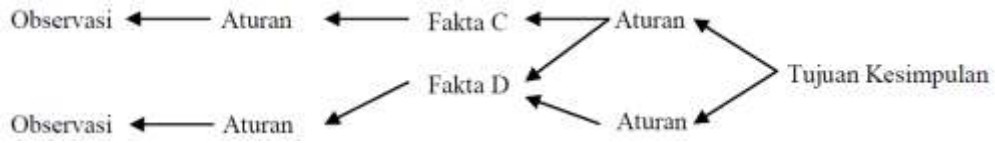
Secara umum mesin inferensi yang utama pada sistem pakar dapat dibedakan menjadi dua (Minarni dan Hidayat, 2013) yaitu runut maju dan runut balik. Runut maju (*forward chaining*) merupakan pendekatan yang dimonitori data (*data-driven*). Dalam pendekatan ini pelacakan dimulai dari informasi masukan, dan selanjutnya mencoba menggambarkan kesimpulan. Pelacakan ke depan mencari fakta yang sesuai dengan bagian *If* dari aturan *If…then* (Reisa dkk, 2013) dapat dilihat pada Gambar 2.2



Gambar 2. 2 *Proses Forward Chaining (Maradesa, 2012)*

Runut balik (*backward chaining*) merupakan strategi pencarian yang arahnya kebalikan dari runut maju (*forward chaining*) (Dahria, 2012). Percobaan fakta atau pernyataan dimulai dari bagian sebelah kanan (*THEN*) dulu. Dengan kata lain, penalaran dimulai dari hipotesis terlebih dahulu dan untuk menguji kebenaran hipotesis tersebut harus dicari fakta – fakta uang ada dalam basis pengetahuan. Proses pencarian dimulai dari tujuan, yaitu kesimpulannya merupakan solusi yang ingin dicapai, kemudian dari kaidah – kaidah yang diperoleh, masing – masing kesimpulan

Backward Chaining jalur yang mengarah ke kesimpulan tersebut merupakan solusi yang dicari, jika tidak sesuai makan kesimpulan tersebut bukan merupakan solusi yang dicari. *Backward Chaining* memulai proses pencarian dengan suatu tujuan sehingga strategi ini disebut *goal-driven* terlihat pada gambar 2.3.



Gambar 2. 3 *Proses Backward Chaining (Erhet, 2013)*

Kaidah menyediakan cara formal untuk merepresentasikan rekomendasi, arahan, atau strategi. Kaidah produksi dituliskan dalam bentuk jika-maka (*if-then).* Kaidah *if-then* menghubungkan anteseden dengan konsekuensi yang mengakibatkannya. Berbagai struktur kaidah *if-then* yang menghubungkan objek atau atribut sebagai berikut: (1) *IF* premis *THEN* konklusi (2) *IF* masukan *THEN* keluaran (3) *IF* kondisi *THEN* tindakan (4) *IF* anteseden *THEN* konsekuen (5) *IF* data *THEN* hasil (6) *IF* tindakan *THEN* tujuan (7) *IF* aksi *THEN* reaksi *IF* sebab *THEN* akibat (8) *IF* gejala *THEN* diagnosa.

Premis mengacu pada fakta yang harus benar sebelum konklusi tertentu dapat diperoleh. Masukan mengacu pada data yang harus tersedia sebelum keluaran dapat diperoleh. Kondisi mengacu pada keadaan yang harus berlaku sebelum tindakan dapat diambil. Anteseden mengacu pada situasi yang terjadi sebelum konsekuen dapat diamati. Data mengacu pada informasi yang harus tersedia sehingga sebuah hasil dapat diperoleh. Tindakan mengacu pada kegiatan yang harus dilakukan sebelum hasil dapat diharapkan. Aksi mengacu pada kegiatan yang menyebabkan munculnya efek dari tindakan tersebut. Sebab mengacu pada keadaan tertentu. Gejala mengacu pada keadaan yang menyebabkan adanya kerusakan atau keadaan tertentu yang mendorong adanya pemeriksaan (Ramadhan, 2011).

Sebelum sampai pada bentuk kaidah produksi, terdapat langkah-langkah yang harus ditempuh dari pengetahuan yang didapatkan dalam domain tertentu. Langkah-langkah tersebut adalah menyajikan pengetahuan yang berhasil didapatkan dalam tabel keputusan (*decision table)* yang merupakan suatu cara untuk mendokumentasikan pengetahuan. Tabel keputusan merupakan matrik kondisi yang dipertimbangkan dalam pendeskripsian kaidah. Kaidah yang disajikan dalam bentuk kaidah produksi disusun dari tabel keputusan. Pembuatan suatu kaidah dilakukan dengan beberapa tahapan. Meskipun kaidah secara langsung dapat dihasilkan dari tabel keputusan tetapi untuk menghasilkan kaidah yang efisien terdapat suatu langkah yang harus ditempuh yaitu membuat pohon keputusan terlebih dahulu (Hartati dan Iswanti, 2008). Kemudian dari tabel keputusan dibuat pohon keputusan (*decision tree*) merupakan metode klasifikasi dan prediksi yang kuat. Metode pohon keputusan mengubah fakta yang sangat besar menjadi pohon keputusan yang mempresentasikan

aturan. Aturan dapat dengan mudah dipahami dengan bahasa alami. Dan mereka juga dapat diekspresikan dalam bentuk bahasa basis data seperti *Structured Query Language* untuk mencari *record* pada kategori tertentu. Pohon keputusan juga berguna untuk mengeksplorasi data, menemukan hubungan tersembunyi antara sejumlah calon variabel masukan dengan sebuah variabel target. Karena pohon keputusan memadukan antara eksplorasi data dan permodelan, maka pohon keputusan sangat bagus sebagai langkah awal dalam proses permodelan bahkan ketika dijadikan sebagai model akhir dari beberapa teknik lain. Pohon keputusan adalah struktur *flowchart* yang menyerupai *tree* (pohon), dimana setiap simpul internal menandakan suatu tes pada atribut, setiap cabang merepresentasikan hasil tes, dan simpul daun merepresentasikan kelas atau distribusi kelas. Alur pada pohon keputusan ditelusuri dari simpul akar ke simpul daun yang memegang prediksi kelas untuk contoh tersebut. Pohon keputusan mudah untuk dikonversi ke aturan klasifikasi (*classification rules*) (Syahril, 2011).

Ada sepuluh komplikasi yang biasa timbul akibat pemakaian *softlens* yaitu: noda kornea, *blepharitis,* reaksi alergi, sindrom mata kering, *corneal edema,* infeksi mata, *infitrates, mocrobila keratitis, vaskularisasi kornea,* dan *giant papilary conjunctivitas.*

2.5 Android

Android merupakan sistem operasi berbasis linux yang bersifat terbuka (*open source)* dan dirancang untuk perangkat seluler layar sentuh seperti *smartphone* dan *computer tablet.* Android dikembangkan oleh Android, Inc,. dengan dukungan finansial dari google yang kemudian dibeli pada tahun 2005. Android dirilis secara resmi pada tahun 2007, bersamaan dengan didirikannya *Open Handset Alliance.*

Tampilan android didasarkan pada manipulasi langsung, menggunakan masukan sentuh yang serupa dengan tindakan di dunia nyata, seperti menggesek, mengetuk, mencubit dan membalikkan cubitan untuk memanipulasi objek di layar. Sifat android yang terbuka telah membuat bermunculannya sejumlah besar komunitas pengembang aplikasi untuk menggunakan android sebagai dasar proyek pembuatan aplikasi, dengan menambahkan fitur-fitur baru bagi android pada perangkat yang secara resmi dirilis dengan menggunakan sistem operasi lain. (Salbino, 2014)

2.6 Framework Laravel

Laravel adalah sebuah framework PHP yang dirilis dibawah lisensi MIT, dibangun dengan konsep *Model View Controller* (MVC). Laravel adalah pengembangan website berbasis MVP yang ditulis dalam PHP yang dirancang untuk meningkatkan kualitas perangkat lunak dengan mengurangi biaya pengembangan

awal dan biaya pemeliharaan, dan untuk meningkatkan pengalaman bekerja dengan aplikasi dengan menyediakan sintaks yang ekspresif, jelas dan menghemat waktu.

MVC adalah sebuah pendekatan perangkat lunak yang memisahkan aplikasi logika dari presentasi. MVC memisahkan aplikasi berdasarkan komponen- komponen aplikasi, seperti : manipulasi data, controller, dan user interface.

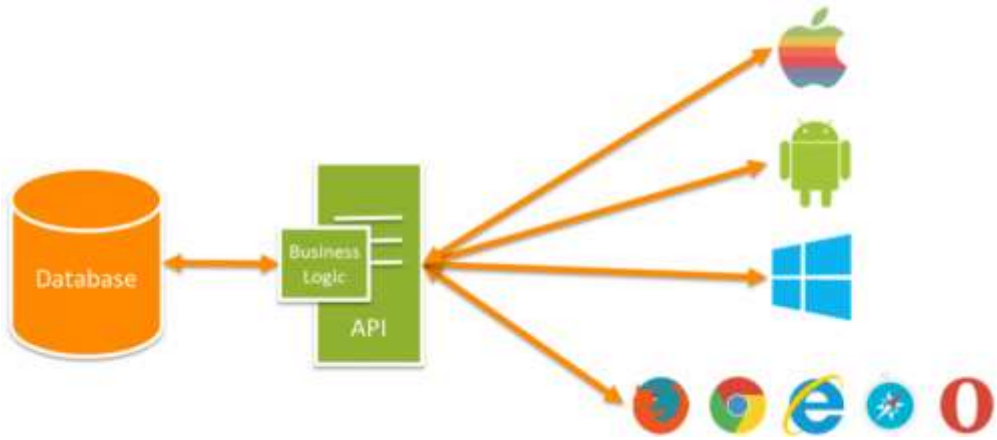
1. Model, Model mewakili struktur data. Biasanya model berisi fungsi-fungsi yang membantu seseorang dalam pengelolaan basis data seperti memasukkan data ke basis data, pembaruan data dan lain-lain.
2. View, View adalah bagian yang mengatur tampilan ke pengguna. Bisa dikatakan berupa halaman web.
3. Controller, Controller merupakan bagian yang menjembatani model dan view.

Beberapa fitur yang terdapat di Laravel :

- Bundles, yaitu sebuah fitur dengan sistem pengemasan modular dan tersedia beragam di aplikasi.
- Eloquent ORM, merupakan penerapan PHP lanjutan menyediakan metode internal dari pola *"active record"* yang menagatasi masalah pada hubungan objek database.
- Application Logic, merupakan bagian dari aplikasi, menggunakan controller atau bagian Route.
- Reverse Routing, mendefinisikan relasi atau hubungan antara Link dan Route.
- Restful controllers, memisahkan logika dalam melayani HTTP GET and POST.
- Class Auto Loading, menyediakan loading otomatis untuk class PHP.
- View Composer, adalah kode unit logikal yang dapat dieksekusi ketika view sedang loading.
- IoC Container, memungkin obyek baru dihasilkan dengan pembalikan controller.
- Migration, menyediakan sistem kontrol untuk skema database.
- Unit Testing, banyak tes untuk mendeteksi dan mencegah regresi.

2.7 Web API

API adalah singkatan dari *Application Programming Interface*, dan memungkinkan *developer* untuk mengintegrasikan dua bagian dari aplikasi atau dengan aplikasi yang berbeda secara bersamaan. *API* terdiri dari berbagai elemen seperti *function, protocols, dan tools* lainnya yang memungkinkan *developers* untuk membuat aplikasi. Tujuan penggunaan *API* adalah untuk mempercepat proses *development* dengan menyediakan *function* secara terpisah sehingga *developer* tidak perlu membuat fitur yang serupa. Penerapan *API* akan sangaat terasa jika fitur yang diinginkan sudah sangat kompleks, tentu membutuhkan waktu untuk membuat yang serupa dengannya. Misalnya: integrasi dengan *payment gateway*. Terdapat berbagai jenis sistem *API* yang dapat digunakan, termasuk sistem operasi, *library*, dan web.



Gambar 2. 4 *cara kerja web API*

API yang bekerja pada tingkat sistem operasi membantu aplikasi berkomunikasi dengan *layer* dasar dan satu sama lain mengikuti serangkaian protokol dan spesifikasi. Contoh yang dapat menggambarkan spesifikasi tersebut adalah *Portable Operating System Interface* (POSIX). Dengan menggunakan standar POSIX, aplikasi yang di-*compile* untuk bekerja pada sistem operasi tertentu juga dapat bekerja pada sistem lain yang memiliki kriteria yang sama. *Software library* juga memiliki peran penting dalam menciptakan *compatibility* antar sistem yang berbeda.

Aplikasi yang berinteraksi dengan *library* harus mengikuti serangkaian aturan yang ditentukan oleh *API*. Pendekatan ini memudahkan *software developer*untuk membuat aplikasi yang berkomunikasi dengan berbagai *library* tanpa harus memikirkan kembali strategi yang digunakan selama semua *library* mengikut API yang sama. Kelebihan lain dari metode ini menunjukkan betapa

mudahnya menggunakan *library* yang sama dengan bahasa pemrograman yang berbeda.

Seperti namanya, Web API dalam diakses melalui protokol HTTP, ini adalah konsep bukan teknologi. Kita bisa membuat Web API dengan menggunakan teknologi yang berbeda seperti *PHP*, *Java*, *.NET*, dll. Misalnya *Rest* API dari *Twitter* menyediakan akses *read* dan *write* data dengan mengintegrasikan *twitter* kedalam aplikasi kita sendiri.

2.8 Fitur Web API

Untuk membuat Web API, beberapa hal yang harus disediakan adalah:

1. Mendukung fungsi CRUD yang bekerja melalui HTTP protocol dengan method GET, POST, PUT dan DELETE
2. Memiliki *response Accept Header* dan HTTP *status code*
3. *Response* dengan format JSON, XML atau format apapun yang kamu inginkan. Akan tetapi kebanyak digunakan kedalam format JSON.
4. Mendukung fitur MVC seperti *routing, controllers, action results, filter, model, IOC container*, dll.
5. Web API dapat berjalan di *Apache* atau *web server* lainnya yang didukung sesuai bahasa pemrograman yang digunakan.

Web API seperti sebuah alamat web (*end point*) yang dibuat untuk menangani beberapa *task* sesuai *request* yang diterima, juga terkadang memiliki *parameter*sebagai data yang dibutuhkan agar dapat menampilkan hasil yang diinginkan, juga pada beberapa kasus untuk mengakses API dibutuhkan kode otentikasi yang telah diizinkan untuk melihat data yang diinginkan. Semua *rule* ini ditentukan oleh *programmer* yang membuatnya.