APLIKASI DESKTOP MULTI PLATFORM UNTUK REDIS CLIENT BERBASIS TEKNOLOGI WEB MENGGUNAKAN FRAMEWORK ELECTRONJS DAN REACTJS

D. S. Ramdan
Teknik Informatika, Politeknik TEDC Bandung
Email: ramdanplusplus@gmail.com

Abstrak

Aplikasi *desktop* adalah perangkat lunak yang berdiri sendiri yang dapat dijalankan di *PC desktop* atau *laptop*. Dikalangan pengguna, perangkat lunak aplikasi *desktop* ini sangat populer, karena memeliki beberapa kelebihan diantaranya: (1) aplikasi *desktop* dapat diakses secara *offline*, (2) aplikasi *desktop* memiliki respon yang cepat, (3) aplikasi desktop kaya akan *user experience* (Patel et al., 2017). Perangkat lunak multi *platform* adalah perangkat lunak yang dapat berjalan pada lebih dari arsitektur komputer dan sitem operasi. Pengembangan aplikasi desktop multi *platform* memerlukan sumber daya yang banyak baik itu waktu, *human resource* maupun *cost*, karena setiap sistem operasi memiliki *API* yang berbeda untuk mejalan sebuah aplikasi diatasnya, sehingga *developer* harus menggunakan *tools* dan *code base* yang berbeda untuk membuat aplikasi yang berjalan pada multi *paltfrom* (Wikipedia, 2018). Penelitian ini bertujuan untuk mengebangkan aplikasi desktop multi *platform* yang memiliki *user experience* yag sama dengan *native desktop application* menggunakan *tools* dan *code base* yang sama untuk setiap *platform* sehingga dapat mengurangi sumber daya yang dibutuhkan baik waktu, *human resource* dan *cost* dengan menggunakan *ElectronJS* dan *ReactJS* dengan studi kasus aplikasi *Redis Client*.

Kata kunci: aplikasi *desktop*, multi *platform*, *electronjs*, *reactjs*, *redis,* teknologi *web*

Abstract

A desktop application is a standalone piece of software which runs in desktop or laptop. Desktop applications are popular among user due to the following benefits (1) Applications are available offline (2) Application has faster response (3) It provides rich user experience. Multi-platform software is software that can run on more than computer architecture and operating systems. Development of multi-platform desktop applications requires a lot of resources both time, human resources and cost, because each operating system has a different API to run an application on it, so developers must use different tools and code bases to create applications that run on multi paltfrom (Wikipedia, 2018). This study aims to develop multi-platform desktop applications that have same user experience as native desktop applications using the same tools and code base for each platform, so it can help to reduce the resources needed for development both time, human resources and cost using ElectronJS and ReactJS by study case of Redis Client application.

Keywords: *desktop application, multi platform, electronjs, reactjs, redis, web technology*

I. PENDAHULUAN

Aplikasi *desktop* adalah perangkat lunak yang berdiri sendiri yang dapat dijalankan di *PC desktop* atau *laptop*. Biasanya aplikasi *desktop* ini di kembangkan dengan menggunakan bahasa pemrograman seperti *VB*, *C#*, *Java* dan sebagianya. Dikalangan pengguna, perangkat lunak aplikasi desktop ini sangat populer, karena memeliki beberapa kelebihan diantaranya: (1) aplikasi *desktop* dapat diakses secara *offline*, (2) aplikasi *desktop* memiliki respon yang cepat, (3) aplikasi *desktop* kaya akan *user experience* (Patel et al., 2017).

Untuk menjalankan sebuah aplikasi atau perangkat lunak *PC desktop* atau *laptop* memerlukan sistem operasi yang dapat melakukan komunikasi dengan *hardware*, dan menyediakan layanan-layanan yang diperlukan oleh aplikasi yang berajalan diatasnya (Pankaj et al., 2014). Perangkat lunak multi *platform* adalah perangkat lunak yang dapat berjalan pada lebih dari arsitektur komputer dan sitem operasi. Mengembagkan aplikasi multi *pltaform* adalah sebuah tugas yang dapat memakan waktu yang sangat lama, karena setiap sistem operasi yang berbeda memiliki *Application*

Programming Interface (API) yang berbeda (Wikipedia, 2018).

Aplikasi *web* merupakan alikasi multi *platform* karena dapat diakses dari semua *platform* menggunakan aplikasi *web browser*. Sedangkan teknologi *web* adalah *tools* yang digunakan untuk membuat aplikasi multi *platform* berbasis *web* yang dapat diakses dari bermacam-macab *web browser* pada sistem operasi yang berbeda (Wikipedia, 2018). Aplikasi berbasis *web* ini dapat dikembangkan dengan menggunakan Bahasa pemrograman *HTML*, *CSS* dan *Javascript* (Patel et al., 2017). *ReactJS* adalah *view rendering JavaScript library* yang sangat bagus untuk menangani *JavaScript view component* yang berbasis *single page application* dengan menggunakan *data-driven UI* dengan cara yang efisien, dinamis, dan menarik secara visual. *ReactJS* akan secara otomatis mengelola pembaruan UI ketika ada perubahan data (Ian, 2017).

ElectronJS adalah *runtime* yang memungkinkan pengembang aplikasi membuat aplikasi *desktop* dengan menggunakan bahasa pemrograman *HTML*, *CSS*, dan *JavaScript*. *ElectronJs* memungkinkan *web developer* untuk menggunakan teknologi *web* yang sudah diketahuinya untuk membangun aplikasi *desktop* multi *platfrom* yang dapat berjalan pada beberapa sistem operasi seperti *Windows*, *MacOS*, dan *Linux*. *ElectronJS* menggabungkan *Chromium Content Module* dan *runtime NodeJS*. *ElectronJs* adalah salah satu pilihan yang bagus untuk untuk membangun aplikasi *desktop* multi *platform* menggunkan teknologi *web* yang mempunyai *user experience* seperti *native desktop application* (Steve, 2018).

Setiap aplikasi memerlukan database untuk menyimpan data, baik *user setting data* atau *business data*. *Redis* adalah *NoSQL (Not only SQL) database* yang berbasis *key-value*. *Redis* juga disebut *data structure server* karena memiliki *powerful data types*, seperti *Strings*, *Hashes*, *Lists*, *Sets*, *Sorted Sets*, *Bitmaps*, dan *HyperLogLogs*. Secara *default*, Redis menyimpan semua data dalam memori, oleh karena itu operasi baca dan tulis menjadi sangat cepat. Selain disimpan di memori, redis juga bisa diset untuk menyimpan data pada *persistence disk* (Maxwell et al., 2015). *Redis* dapat digunakan untuk *login cookies*, *shopping cart cookies*, *caching generated web pages*, *caching database rows*, *analyzing web page visits*, *task queueing messaging* (Josiah, 2013).

Pengembangan aplikasi *desktop* multi *platform* memerlukan sumber daya yang banyak baik itu waktu, *human resource* maupun *cost*, karena setiap sistem operasi memiliki *API* yang berbeda untuk mejalan sebuah aplikasi diatasnya, sehingga *developer* harus menggunakan *tools* dan *code base* yang berbeda untuk membuat aplikasi yang berjalan pada multi *paltfrom* (Wikipedia, 2018).

Penelitian ini bertujuan untuk mengebangkan aplikasi *desktop* multi *platform* yang memiliki *user experience* yag sama dengan *native desktop application* menggunakan *tools* dan *code base* yang sama untuk setiap *platform* sehingga dapat mengurangi sumber daya yang dibutuhkan baik waktu, *human resource* dan *cost* dengan menggunakan *ElectronJS* dan *ReactJS* dengan studi kasus aplikasi *Redis Client*.

II. LANDASAN TEORI

Aplikasi Desktop

Aplikasi berbasis *desktop* adalah perangkat lunak yang di-*install* pada satu komputer yang akan melakukan fungsi dan tugas tertentu yang dirancang untuknya. Namun, perangkat yang sama dapat mengakomodasi banyak pengguna dengan bantuan jaringan. Contohnya adalah *media player*, *word processor*, dll (Maxwell et al., 2015). Aplikasi *desktop* adalah perangkat lunak yang berdiri sendiri yang dapat dijalankan di *PC desktop* atau *laptop*. Biasanya aplikasi *desktop* ini di kembangkan dengan menggunakan bahasa pemrograman seperti *VB, C#, Java* dan sebagianya (Patel et al., 2017). Aplikasi *desktop* memiliki beberapa kelebihan, diantaranya (Midhun, 2018):

1. *Data Security*: Tidak seperti aplikasi *web*, semua data disimpan dalam sistem komputer pengguna, jadi tidak perlu khawatir akan diretas. Pengguna memiliki kontrol penuh atas aplikasi yang berdiri sendiri dan oleh karena itu memungkinkan perlindungan dari berbagai kerentanan.
2. *Available Controls*: Jika dibandingkan dengan aplikasi berbasis *web*, aplikasi *desktop* dilengkapi dengan sejumlah *native control*. *Native control* memungkinkan aplikasi mengakses perangkat keras dan komponen *OS* yang mendasarinya.
3. *Flexibilty*: aplikasi *desktop* dapat menggunakan perangkat keras komputer pengguna seperti *port serial*, kamera, *port* jaringan.

Performance: Aplikasi *desktop* jauh lebih cepat dan lebih responsif jika dibandingkan dengan aplikasi *web*. Ini karena aplikasi *web* secara inheren

harus berkomunikasi degan *web server* melalui akses *internet*. Di sisi lain, aplikasi *desktop*, jika dirancang dengan benar hanya akan memuat apa yang dibutuhkan saja. Jadi, aplikasi *desktop* menggunakan lebih sedikit memori dan lebih sedikit sumber daya, sehingga meningkatkan kinerja dan meningkatkan efisiensi aplikasi.

Teknologi *Web*

Teknologi *web* adalah *tools* yang digunakan untuk membuat aplikasi multi *platform* berbasis *web* yang dapat diakses dari bermacam-macab *web browser* pada sistem operasi yang berbeda (Wikipedia, 2018). Aplikasi berbasis *web* ini dapat dikembangkan dengan menggunakan Bahasa pemrograman *HTML*, *CSS* dan *Javascript* (Patel et al., 2017). Aplikasi berbasis *web* biasanya digambarkan sebagai aplikasi multi platform karena aplikasi dapat diakses dari berbagai *web browser* pada sistem operasi yang berbeda. Aplikasi web umumnya menggunakan arsitektur sistem *client-server*, dan meiliki kompleksitas dan fungsionalitas yang sangat bervariasi.

HTML adalah *markup language*, yang membuat *text* dapat ditampilkan dalam bentuk tertentu (sesuai dengan *text* yang di *markup*) pada *web browser*. *Cascading Style Sheets* (*CSS*) adalah *grouping of formatting instructions* yang mengontrol tampilan dari halaman *web HTML* sehingga tampilan *web* menjadi lebih menarik dan nyaman untuk dilihat. *JavaScript* adalah *web scripting language* yang didukung oleh hampir semua *web browser* yang berguna untuk membuat halaman *web HTML* menjadi lebih *interactive* (Julie, 2012).

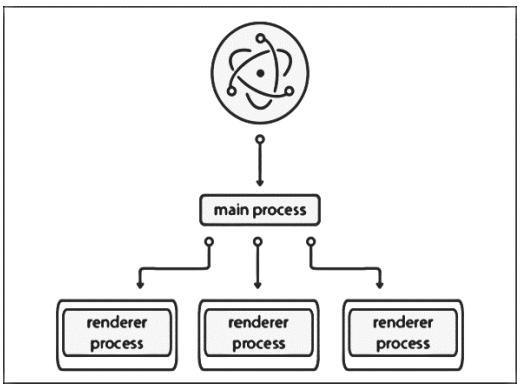
ReactJS

ReactJS adalah *view rendering JavaScript library* yang sangat bagus untuk menangani *JavaScript view component* yang berbasis *single page application* dengan menggunakan *data-driven UI* dengan cara yang efisien, dinamis, dan menarik secara visual. *ReactJS* akan secara otomatis mengelola pembaruan *UI* ketika ada perubahan data. ReactJS cukup fleksibel sehingga dapat digunakan dalam berbagai kapasitas dan konteks terlepas dari teknologi yang digunakan.

Sejak awal rilis pada tahun 2013, *ReactJS* telah diterima dengan baik oleh komunitas *JavasSript* diseluruh dunia, dan sejak itu sampai saat ini popularitasnya terus meningkat. *ReactJS* memiliki beberapa kelebihan dalam hal *performance*, *scalability*, *simplicity* dalam proses pengembangan baik dalam ukuran maupun kompleksitas data yang selalu berubah setiap waktu. *ReactJS* akan secara otomatis mengelola semua pembaruan *UI* Anda ketika data yang mendasarinya berubah, dan itu akan dilakukan dengan cepat! Itu adalah salah satu dari banyak hal yang disukai oleh para *developer* (Ian, 2017).

ElectronJs

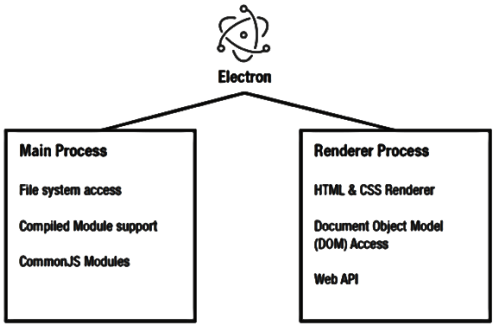
ElectronJS adalah *open source library* yang dikembangkan oleh *Github* untuk membangun aplikasi desktopmulti *platform* menggunakan *HTML*, *CSS* dan *JavaScript*. *ElectronJS* bekerja dengan cara menggabungkan *Chromium* dan *NodeJS* menjadi *single runtime* sehingga aplikasi dapat di-*package* menjadi *installer* untuk *Mac OS*, *Windows* dan *Linux*. Proyek *ElectronJS* dimulai pada tahun 2013 untuk membangun aplikasi *text editor* yang bernama *Atom*, kemudian pada tahun 2014 menjadi proyek *Open Source* (ElectronJs, 2018).



Gambar 1. *ElectronJS process*

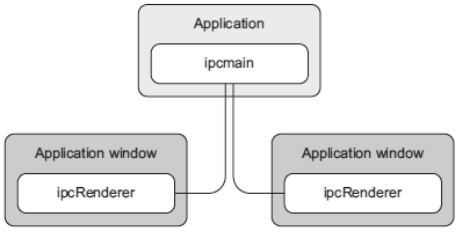
Aplikasi yang berbasis *ElectronJS* berjalan dalam dua proses yang berbeda, yaitu *Main Process* dan *Renderer Process*. Masing-masing dari dua *process* tersebut memiliki tanggung jawab yang spesifik dalam menjalankan aplikasi (Chris et al., 2017).

ElectronJS menyediakan koleksi *NodeJS Modules* yang bisa digunakan dalam aplikasi, modul-modul tersebut berjalan di proses tertentu. Sebagai contoh, akses untuk *API* dari sistem operasi hanya tersedia di *Main Process*, sedangkan akses ke *system's clipboard* tersedia di *Main* dan *Renderer Process* (Chris et al., 2017). *Main Process* menangani *OS integration* (*file system access*, *compiled module support*, *commonJS module*). Sedangkan *Renderer Process* menangai *user interface* dan merespon *user event (HTML & CSS rednderer, Document Object Model Access, web API)* (Steve, 2018).



Gambar 2. *ElectronJS process responsibility*

Walaupun kedua proses tesebut tepisah dan terisolasi, namun keduanya bisa saling berkomikasi mengunakan *IpcMain* dan *IpcRenderer* (lihat Gbr. 2)(Chris et al., 2017).



Gambar 3. Komunikasi antar *process* pada *framework ElectronJS*

Redis

Redis adalah *NoSQL (Not only SQL) database* yang berbasis *key-value*. *Redis* juga disebut *data structure server* karena memiliki *powerful data types*, seperti *Strings*, *Hashes*, *Lists*, *Sets*, *Sorted Sets*, *Bitmaps*, dan *HyperLogLogs*. Secara *default*, *Redis* menyimpan semua data dalam memori, oleh karena itu operasi baca dan tulis menjadi sangat cepat. Selain disimpan di memori, redis juga bisa diset untuk menyimpan data pada *persistence disk*.

Persistensi data dalam *Redis* dapat dicapai dengan membuat *binary snapshot* dari data yang disimpan atau *human-readable file* dengan urutan semua perintah yang dijalankan dari waktu ke waktu. Ini masing-masing dikenal sebagai *snapshotting* dan *journaling*. *Redis* adalah singkatan dari *REmote DIctionary Server*. *Redis* ditulis dalam bahasa *C* oleh *Salvatore Sanfilippo* pada tahun 2006 dan saat ini memiliki banyak kontributor. *Redis* adalah proyek *open source* yang mapan dan telah digunakan dalam *production environment* selama bertahun-tahun oleh perusahaan besar, termasuk *Twitter*, *GitHub*, *Tumblr*, *Pinterest*, *Instagram*, *Hulu*, *Flickr*, dan *The New York Times* (Maxwell et al., 2015).

Redis menawarkan fitur yang bermanfaat untuk menyelesaikan beberapa masalah, seperti *replication*, *persistence*, dan *client-side sharding*. Fitur tersebut memungkinkan *redis* untuk melakukan *scaling* dari *protorype system* yang secara umum memiliki jumlah data dan *request* yang realtive kecil menjadi *production system* yang memiliki ratusan *gigabyte* data dan jutaan *request* perdetik. *Redis* dapat digunakan untuk *login cookies*, *shopping cart cookies*, *caching generated web pages*, *caching database rows*, *analyzing web page visits*, *task queueing messaging* (Josiah, 2013).

III. ANALISA DAN DESAIN

Analisa Sistem Yang Akan Dikembangkan

Sistem yang akan dikembangkan sebagai studi kasus dalam peneilitian ini adalah aplikasi *Redis Client* berbasis *desktop*. *Redis* adalah *NoSQL* database berbasis *key-value* dalam menyimpan data dan memiliki popularitas yang tinggi di komunitas pengembangan perangkat lunak (Maxwell et al., 2015). Sistem yang akan dikembangkan ini memiliki beberapa fitur *basic CRUD (Create, Read, Update, Delete)* diantaranya:

1. Autentikasi menggunakan *password*
2. Memilih *database*
3. Melihat list *key* yang sudah disimpan
4. Melihat *value* dari *key* yang dipilih
5. Menambah *key* dan *value* baru
6. Mengubah *key* dan *value*
7. Menghapus *key* dan *value*
8. Menampilka *key* dalam bentuk *raw text* atau format *JSON.*

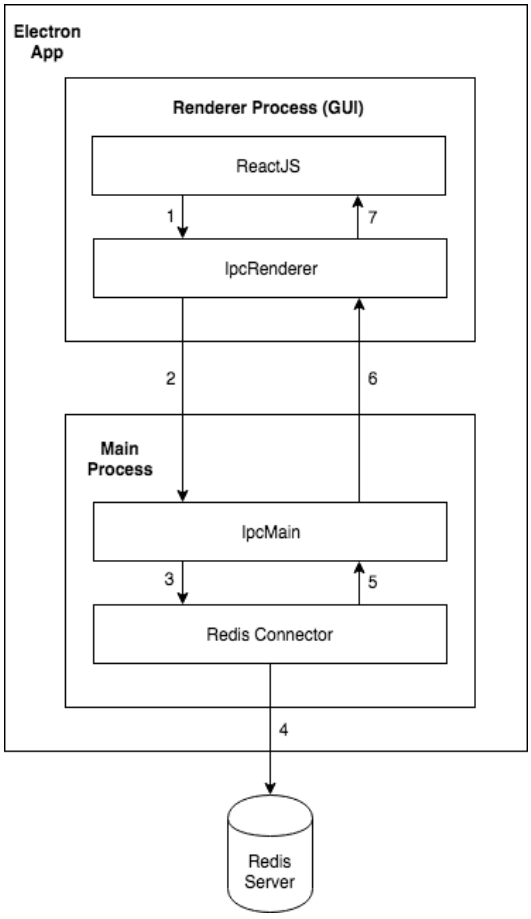
Arsitektur Sistem

Membangun sistem *Redis Client*, arsitektur perangkat lunak nya dibangun berdsarakan dua proses yang berbeda pada *ElectronJS* yaitu *Main Process* dan *Renderer Process*. Dibawah ini adalah arsitektur perangkat lunak dari sistem yang akan dibangun.

Dibawah ini penjelesan mengenai arsitektur sistem yang akan dikembang seperti pada Gambar 2:

1. *GUI ReactJS* pada *renderer process* akan mengirimkan *request* ke *main process* melalui *IpcRenderer*.
2. *IpcRenderer* akan meneruskan *request* dari *GUI* ke *main process* melalui *IpcMain*.
3. *IpcMain* akan memanggil fungsi terkait pada *Redis Connector* sesuai dengan *request* yang dikirim dari *GUI*.

4. *Redis Connector* berkomunikasi dengan *Redis Server* untuk melakukan *I/O process*.
5. *Redis Connector* akan mengembalikan data respon sesuai dengan *request* yang dikirim dari *GUI*.
6. *IpcMain* akan meneruskan data respon dari *Redis Connector* ke *GUI* melalui *IpcRenderer*.
7. *IpcRenderer* akan mengirimkan data untuk di-*render* ke halaman tertentu yang mengirimkan *request* untuk diolah dan ditampilkan di *GUI*.



Gambar 4. Arsitektur aplikasi yang akan dikembangkan

Pengembangan Sistem

Sistem ini dikembangkan berdasarkan pada *framework ElectronJS* dan *ReactJS* sehingga bisa berjalan di multi *platform*. Bahasa pemrograman yang digunakan untuk mengembangkan sistem adalah *HTML*, *CSS*, dan *JavaScript*. Pada dasaranya pembuatan atau pengembangan perangkat lunak dapat dilakukan di sistem operasi mana saja, baik *MacOS*, *Windows* ataupun *Linux*.

Namun pada penelitian ini, proses pengembangan perangkat lunak menggunakan *MacOS*. Kemudian untuk membuat *installer* aplikasi, harus menggunakan sistem operasi tujuan. Misalkan, kita akan membuat *installer* untuk sistem operasi *Windows*, maka kita harus melakukan *packaging installer* di sistem operasi *Windows*.

IV. IMPLENTASI DAN PENGUJIAN

Implementasi Sistem

Implementasi dan pengujian sistem dilakukan pada tiga sistem operasi yang berbeda, yaitu *MacOS High Sierra*, *Windows 10* dan *Ubuntu 17.10*. Aplikasi yang di-*install* dan dijalankan diatas sistem operasi tersebut dapat berjalan sesuai dengan fungsinya masing-masing. Dibawah ini adalah hasil pengujian *Black Box* dari sistem yang telah dikembangkan (aplikasi *Redis Client*).

Tabel 1. Hasil pengujian *Blackbox* dari aplikasi yang telah dikembangkan

Fitur	***MacOS High Sierra***	***Windows 10***	***Ubuntu 18.04***
Autentikasi menggunakan *password*	*OK*	*OK*	*OK*
Memilih *database*	*OK*	*OK*	*OK*
Melihat list *key* yang sudah disimpan	*OK*	*OK*	*OK*
Melihat *value* dari *key* yang dipilih	*OK*	*OK*	*OK*
Menambah *key* dan *value* baru	*OK*	*OK*	*OK*
Mengubah *key* dan *value*	*OK*	*OK*	*OK*
Menghapus *key* dan *value*	*OK*	*OK*	*OK*
Menampilka *key* dalam bentuk *raw text* atau format *JSON*	*OK*	*OK*	*OK*

V. KESIMPULAN DAN SARAN

Kesimpulan

Berdasarkan perancangan dan implementasi aplikasi sistem informasi unit kegiatan mahasiswa berbasis web di Politeknik TEDC Bandung yang telah dilakukan, dapat mengambil kesimpulan sebagai berikut:

Dibawah ini dalah beberapa kesimpulan yang dapat diambil dari penelitian yang telah dilakukan:

1. Aplikasi *desktop* multi *platform* yang memiliki *user experience* yag sama dengan *native desktop application* dapat dikembangkan dengan menggunakan *framework ElectronJS* dan *ReactJS* yang berbasis teknologi *web*.
2. Pengembangan aplikasi *desktop* multi *platform* dengan menggunakan *ElectronJS* dan *ReactJS* dapat mengurangi sumber daya yang dibutuhkan karena *developer* hanya perlu membuat *code base* satu kali untuk bisa berjalan di multi *platform*.

Saran

Penelitian ini berfokus pada bagaimana membuat aplikasi *desktop* multi *platform* dengan menggunakan teknologi *web* (*HTML*, *CSS* dan *JavaScript*). Untuk kedepanya bisa dilakukan penelitian mengenai *performance* dari aplikasi *desktop* yang dibuat menggunakan framework *ElectronJS* dan *ReactJS*, apakah lebih baik, setara atau lebih buruk dari *native desktop application development*.

DAFTAR PUSTAKA

Patel, Sandeep Kumar.(2017). *Quick Desktop Application Development Using Electron*. Leanpub.

G. Pankaj, K. Piyush, Sandeep, W. Saksham, Y. Vishal. (2014). *Operating System.* International Journal of Computer Science and Information Technology Research, vol. 2.

Wikipedia, *Cross Platform*, Wikipedia.com, https://en.wikipedia.org/wiki/Cross-platform, 2018, (diakses: March 01, 2018)

Steve, K.. (2018). *Electron In Action*, Manning Publications Co.

Maxwell, D. D. S., Hugo, L. T.(2015). *Redis Essentials*, Packt Publishing.

Josiah, L. C.(2013). *Redis in Action*, Manning Publications Co.

A. D. Midhun, *Using Electron for Cross Platform Desktop Application Development: An Introduction*, Cabotsolutions.com, https://www.cabotsolutions.com/2017/11/using-electron-for-cross-platform-desktop-application-development-an-introduction, 2018, (diakses: March 05, 2018)

Julie, C. M.(2012). *Sams Teach Yourself HTML, CSS, and JavaScript All in One*, Pearson Education.

ElectronJS, *About Electron*, ElectronJS.com, https://electronjs.org/docs/tutorial/about, 2016, (diakses: March 07, 2018)

Chris, G., Leif, W.(2017), *Electron: From Beginner to Pro: Learn to Build Cross Platform Desktop Applications using Github's Electron*, Appres Publishing.

Ian, C.(2017). *Effective React*, Leanpub.