Purwarupa *Framework* Aplikasi Desktop Menggunakan Teknologi Web

Firmansyah Adiputra*[1], Khabib Mustofa[2]
[1]Fakultas Teknik Universitas Trunojoyo, Madura
[2]Jurusan Ilmu Komputer dan Elektronika, FMIPA UGM, Yogyakarta
e-mail: *[1]**frm.adiputra@gmail.com**, [2]khabib@ugm.ac.id

Abstrak

Aplikasi desktop adalah aplikasi yang berjalan lokal dalam lingkungan desktop dan hanya dapat diakses oleh pengguna desktop. Ini berbeda dengan aplikasi web yang dapat diakses dari manapun melalui jaringan. Namun tidak seperti halnya aplikasi desktop, aplikasi web yang berjalan di atas web browser tidak dapat berintegrasi dengan aplikasi desktop yang berjalan pada sisi klien.

Dalam penelitian ini dibangun purwarupa framework yang diberi nama HAF (Hybrid Application Framework). HAF digunakan untuk mengembangkan dan mengeksekusi jenis aplikasi desktop baru yang diberi nama HyApp (Hybrid Application). Melalui HAF, HyApp dibangun menggunakan teknologi web dan dapat diakses secara lokal maupun melalui jaringan. Saat diakses secara lokal, walaupun dikembangkan dengan teknologi web, HyApp dapat berkomunikasi dengan aplikasi desktop lainnya. Selain itu, melalui API yang disediakan oleh HAF, HyApp akan dapat menerapkan perilaku yang berbeda berdasarkan modus pengaksesan yang dilakukannya.

Kata kunci*—framework, aplikasi desktop, aplikasi web*

Abstract

Desktop application is an application that runs locally in a desktop environment and can be accessed only by desktop users. It differs from web application which can be accessed from anywhere through networks. But unlike desktop applications, web applications cannot integrate nicely with desktop applications from where it is accessed.

This research developes a prototype of framework which is named HAF (Hybrid Application Framework). HAF is used for developing and executing a new type of desktop application, named HyApp (Hybrid Application). Through HAF, HyApp is built using web technologies and can be accessed either locally or from networks. When accessed locally, even though it is built using web technologies, it still can communicate with other desktop applications. Also by using APIs provided by HAF, HyApp is capable to behave differently based on whether it is accessed locally or remotely.

Keywords*—framework, desktop applications, web applications*

Received March 3rd,2014; Revised January 8th, 2015; Accepted January 20th, 2015

1. PENDAHULUAN

Aplikasi desktop adalah aplikasi yang berjalan lokal dalam lingkungan desktop dan hanya dapat diakses oleh pengguna desktop yang mengeksekusinya. Ini berbeda dengan aplikasi web yang memiliki jangkauan penggunaan yang lebih luas, yaitu dapat diakses dari manapun (melalui jaringan) dengan menggunakan *web browser*. Namun tidak seperti halnya aplikasi desktop, aplikasi web tidak dapat berintegrasi dengan baik dengan aplikasi desktop ataupun lingkungan desktop tempatnya diakses melalui *web browser*. Ini disebabkan karena *web browser* tidak menyediakan antarmuka komunikasi antara aplikasi web yang berjalan di atasnya dengan aplikasi desktop ataupun lingkungan desktop.

Seandainya aplikasi desktop dapat pula diakses melalui jaringan, maka saat tidak terhubung ke jaringan, aplikasi tersebut akan tetap dapat diakses secara lokal dan tetap dapat berintegrasi dengan aplikasi desktop lainnya. Saat terhubung ke jaringan, aplikasi tersebut akan dapat memiliki kemampuan untuk dikendalikan dari jarak jauh, menyediakan fungsi/layanan tambahan yang hanya cocok bila diakses melalui jaringan, atau bahkan mengurangi fungsi/layanannya bila dirasa tidak aman untuk diakses melalui jaringan.

Beberapa penelitian pernah dilakukan untuk mengembangkan *framework* yang bisa membuat aplikasi desktop dapat diakses melalui jaringan. Puder membangun sebuah *framework* yang mampu mengekspos GUI (*Graphical User Interface*) dari aplikasi desktop untuk dapat diakses melalui jaringan sebagai halaman web, tanpa perlu dilakukan perubahan kode program pada aplikasi desktop tersebut [1]. Saat diakses melalui jaringan, *framework* Puder akan mengeksekusi aplikasi desktop dengan menggunakan GUI *library* khusus. Melalui GUI *library* tersebut, perintah-perintah konstruksi GUI yang dipanggil oleh aplikasi desktop tidak lagi menghasilkan GUI seperti biasanya, melainkan menghasilkan GUI berupa halaman web yang akan dapat diakses oleh klien melalui *web server* yang disediakan oleh *framework*. *Framework* Puder memiliki kekurangan saat terjadi interaksi GUI yang intensif pada halaman web tersebut. Karena setiap interaksi diproses pada server, maka interaksi GUI yang intensif akan dapat menyebabkan *overhead* pada server. Kelemahan *framework* tersebut dipertegas oleh Gitzel dkk. yang menyatakan bahwa paradigma *event-based* yang biasa digunakan pada aplikasi desktop, tidak efisien untuk diterapkan pada arsitektur web yang berbasis *client/server* [2]. Serupa dengan Puder, Lamberti dan Sanna juga membangun sebuah *framework* yang mampu menyediakan GUI dari aplikasi desktop untuk dapat diakses melalui jaringan tanpa perlu dilakukan perubahan kode program pada aplikasi desktop tersebut [3]. Berbeda dengan *framework* Puder, pada *framework* Lamberti dan Sanna, sebelum aplikasi desktop dapat diakses melalui jaringan, terlebih dahulu dibuat deskripsi GUI dari aplikasi desktop tersebut. Deskripsi GUI tersebut dihasilkan dari proses analisis dan pengklasifikasian citra GUI dari aplikasi desktop. Pada perangkat klien, pengaksesan aplikasi desktop melalui jaringan, dilakukan dengan menggunakan perangkat lunak khusus. Melalui perangkat lunak tersebut, GUI dari aplikasi desktop akan dibangkitkan pada perangkat klien berdasarkan deskripsi GUI yang disediakan oleh *framework*. *Framework* Lamberti dan Sanna ini memiliki kelemahan pada metode analisis dan pengklasifikasian citra GUI yang digunakannya, yaitu metode tersebut tidak dapat secara otomatis mendeteksi perubahan-perubahan tampilan GUI yang terjadi secara dinamis, melainkan masih membutuhkan intervensi pengguna. Selain kekurangan-kekurangan yang telah disebutkan, kedua *framework* tersebut juga hanya dapat memberikan kemampuan kepada aplikasi desktop untuk dapat dikendalikan secara jarak jauh melalui jaringan.

Di sisi lain, aplikasi web saat ini telah mengalami perkembangan. Dengan menggunakan AJAX (*Asynchronous JavaScript and XML*), aplikasi web dapat memiliki interaktifitas dan kecepatan respon yang menyerupai aplikasi desktop [4]. Hal ini mendorong bermunculannya produk-produk *framework* pengembangan aplikasi desktop dengan menggunakan teknologi web, salah satunya adalah Adobe AIR, yaitu sebuah *runtime system*

yang mampu menjalankan aplikasi yang dibuat dengan teknologi web sebagai aplikasi desktop [5]. Selain Adobe AIR, juga terdapat Google Chrome Apps dan TideSDK yang juga merupakan *frameworks* pengembangan aplikasi desktop dengan menggunakan teknologi web [6,7]. Aplikasi yang dihasilkan oleh *frameworks* tersebut memang dibangun menggunakan teknologi web dan mampu mengakses layanan-layanan web melalui jaringan, namun sayangnya aplikasi itu sendiri tidak mampu menyediakan fungsi/layanannya untuk dapat diakses melalui jaringan.

Ide dari penelitian ini adalah membangun sebuah *framework* untuk mengembangkan dan mengeksekusi sebuah jenis aplikasi desktop yang dibangun menggunakan teknologi web, sehingga selain memiliki kemampuan layaknya aplikasi desktop biasa, juga akan dapat diakses melalui jaringan.

2. METODE PENELITIAN

Sistem yang akan dibangun dalam penelitian ini adalah sebuah *framework* yang diberi nama HAF (*Hybrid Application Framework*) yang akan digunakan untuk mengembangkan dan mengeksekusi jenis aplikasi baru yang untuk selanjutnya dalam penelitian ini disebut HyApp (*Hybrid Application*). HyApp memiliki karakteristik sebagai berikut:

- Dapat diakses secara lokal layaknya aplikasi desktop, dan dapat pula diakses melalui jaringan. Untuk mempermudah penyebutan, pengaksesan secara lokal akan disebut sebagai pegaksesan dalam modus desktop. Sedangkan untuk pengaksesan melalui jaringan akan disebut sebagai pengaksesan dalam modus daring.
- Dapat menerapkan perilaku yang berbeda saat diakses melalui jaringan.
- Dapat berkomunikasi dengan aplikasi desktop lainnya saat diakses secara lokal.

Agar HyApp dapat diakses secara lokal maupun melalui jaringan, maka arsitektur yang digunakan oleh HyApp haruslah didasarkan pada arsitektur *client/server*. Selain itu, karena teknologi web saat ini sudah mampu menghasilkan aplikasi dengan interaktifitas dan kecepatan respon yang menyerupai aplikasi desktop, maka HyApp nantinya juga akan dibangun dengan memanfaatkan teknologi web.

HyApp sebagai aplikasi desktop dengan arsitektur *client/server*, maka HAF harus dapat mengeksekusi aplikasi server dari HyApp tersebut tepat saat akan diakses oleh pengguna desktop, dan menghentikan pengeksekusiannya saat pengguna desktop tidak lagi membutuhkannya.

Karena HyApp pada dasarnya adalah aplikasi desktop, maka aplikasi server HyApp akan dieksekusi dalam lingkungan eksekusi yang dimiliki oleh pengguna desktop. Hal tersebut berarti proses yang merupakan hasil pengeksekusian dari server HyApp akan memiliki hak akses yang identik dengan hak akses pengguna desktop. Bila server HyApp tersebut dapat diakses melalui jaringan (dalam modus daring), maka siapapun yang dapat mengaksesnya akan dapat pula menggunakan fungsi/layanan yang disediakan layaknya pengguna desktop. Oleh karena itu HAF harus dapat menyediakan informasi modus pegaksesan HyApp, sehingga saat diakses melalui jaringan, berdasarkan informasi modus pengaksesan tersebut, HyApp akan dapat mengamankan pengaksesan yang dilakukan terhadapnya dengan cara menerapkan perilaku yang berbeda atau dengan cara membatasi fungsi/layanannya.

Seperti yang telah disebutkan bahwa HyApp nantinya akan dibangun menggunakan teknologi web. Itu berarti bahwa aplikasi klien HyApp nantinya akan diakses menggunakan *web browser*. Karena *web browser* tidak menyediakan antarmuka yang dapat digunakan oleh aplikasi yang berjalan di atasnya untuk dapat berkomunikasi dengan aplikasi desktop, maka HyApp yang berjalan di atas *web browser* juga tidak akan dapat berkomunikasi dengan aplikasi desktop. Saat HyApp diakses dalam modus daring hal ini tidaklah menjadi masalah, namun saat HyApp diakses dalam modus desktop maka HyApp akan kehilangan karakteristiknya sebagai aplikasi desktop. Oleh karena itu HAF harus memiliki *web browser* yang khusus digunakan

untuk pengaksesan HyApp dalam modus desktop. *Web browser* tersebut harus dapat menyediakan API yang dapat digunakan oleh HyApp yang berjalan di atasnya untuk berkomunikasi dengan aplikasi desktop. Dalam sistem operasi berbasis GNU/Linux, yang merupakan target sistem operasi dari HAF, aplikasi desktop dan berbagai layanan yang disediakan oleh sistem operasi saling berkomunikasi menggunakan protokol D-Bus. Oleh karena itu, antarmuka komunikasi yang disediakan oleh HAF haruslah menggunakan protokol D-Bus.

Untuk dapat menentukan apakah HAF mampu memenuhi kebutuhan-kebutuhan tersebut, maka dirancang pengujian-pengujian sebagai berikut:

1. **Mekanisme eksekusi HyApp**. Pengujian ini dilakukan untuk menguji kemampuan HAF dalam mengendalikan pengeksekusian HyApp. Dalam pengujian ini terdapat dua kasus pengujian yaitu:
    a. Pengeksekusian HyApp.
    b. Penghentian eksekusi HyApp.
2. **Pengaksesan HyApp**. Pengujian ini dilakukan untuk menguji kemampuan HAF dalam menyediakan HyApp untuk diakses dalam modus desktop maupun modus daring. Dalam pengujian ini terdapat kasus-kasus pengujian sebagai berikut:
    a. Pengaksesan dalam modus desktop.
    b. Pengaksesan dalam modus daring.
3. **Penerapan perilaku berdasarkan modus pengaksesan**. Pengujian ini dilakukan untuk mengetahui apakah melalui API yang disediakan HAF, aplikasi klien HyApp dapat menerapkan perilaku berbeda berdasarkan modus pengaksesannya. Dalam pengujian ini terdapat kasus-kasus pengujian sebagai berikut:
    a. Perilaku klien HyApp saat berjalan dalam modus desktop
    b. Perilaku klien HyApp saat berjalan dalam modus daring
4. **Pemberian respon berdasarkan modus pengaksesan**. Pengujian ini dilakukan untuk mengetahui apakah melalui API yang disediakan HAF, server HyApp dapat memberikan respon berbeda berdasarkan modus pengaksesannya. Dalam pengujian ini terdapat kasus-kasus pengujian sebagai berikut:
    a. Respon server HyApp saat diakses dalam modus desktop
    b. Respon server HyApp saat diakses dalam modus daring
5. **Komunikasi dengan aplikasi desktop lain**. Pengujian ini dilakukan untuk mengetahui apakah HAF mampu menyediakan antarmuka komunikasi D-Bus antara aplikasi klien HyApp yang berjalan dalam modus desktop dengan aplikasi desktop. Dalam pengujian ini terdapat kasus-kasus pengujian sebagai berikut:
    a. Memanggil *method*.
    b. Mengakses nilai *property*.
    c. Memberi nilai *property*.
    d. Memberi respon terhadap perubahan nilai *property*.
    e. Memberi respon terhadap *signal*.

3. HASIL DAN PEMBAHASAN

Dalam purwarupa dari HAF yang berhasil dibangun, ditentukan bahwa HyApp harus menggunakan arsitektur *client*/*server* seperti yang ditunjukkan pada Gambar 1. Dalam arsitektur tersebut, aplikasi server HyApp selalu dieksekusi dan berjalan dalam lingkungan desktop. Pengaksesan HyApp secara lokal (modus desktop) harus dilakukan dari lingkungan desktop yang sama dengan lingkungan desktop tempat servernya dieksekusi. Sedangkan pengaksesan HyApp melalui jaringan (modus daring) dapat dilakukan dari luar lingkungan desktop tersebut.

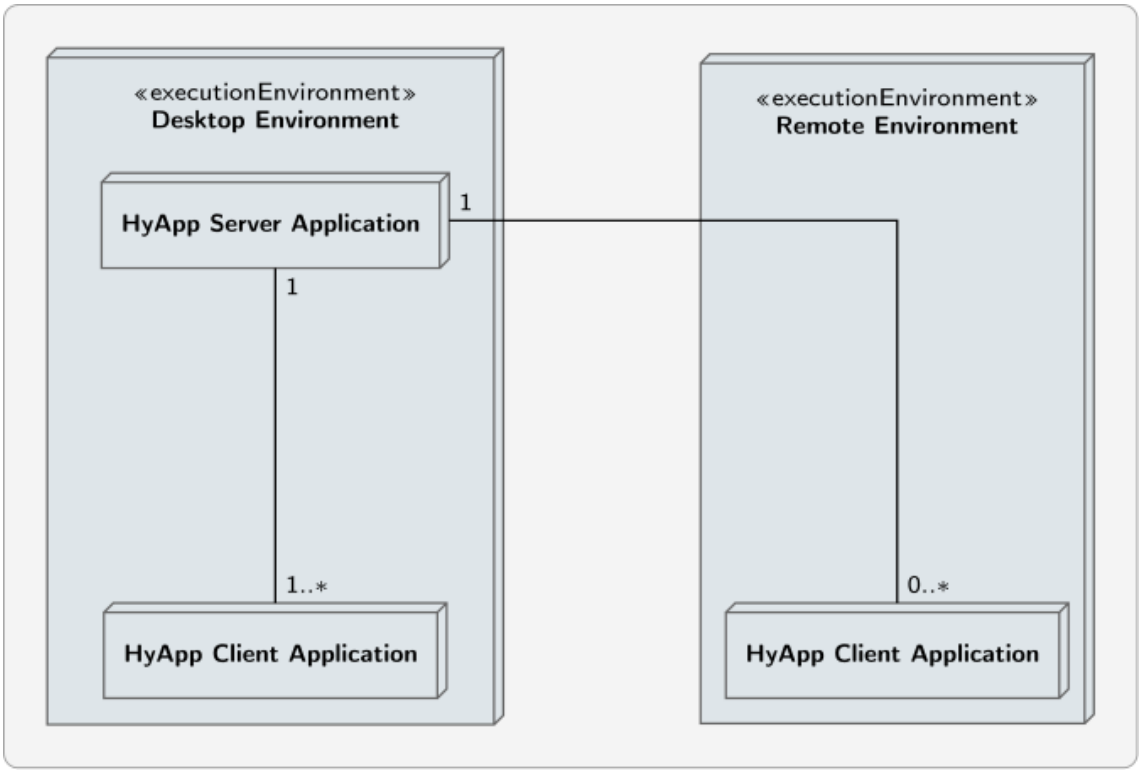
HAF memiliki dua buah subsistem, yaitu lingkungan eksekusi yang digunakan untuk mengeksekusi HyApp, dan API yang digunakan untuk mengembangkan HyApp. Dalam

lingkungan eksekusinya, HAF memiliki tiga buah komponen yang saling bekerja sama untuk mengelola dan mengeksekusi HyApp. Seperti yang ditunjukkan pada Gambar 2, tiga buah komponen tersebut adalah:

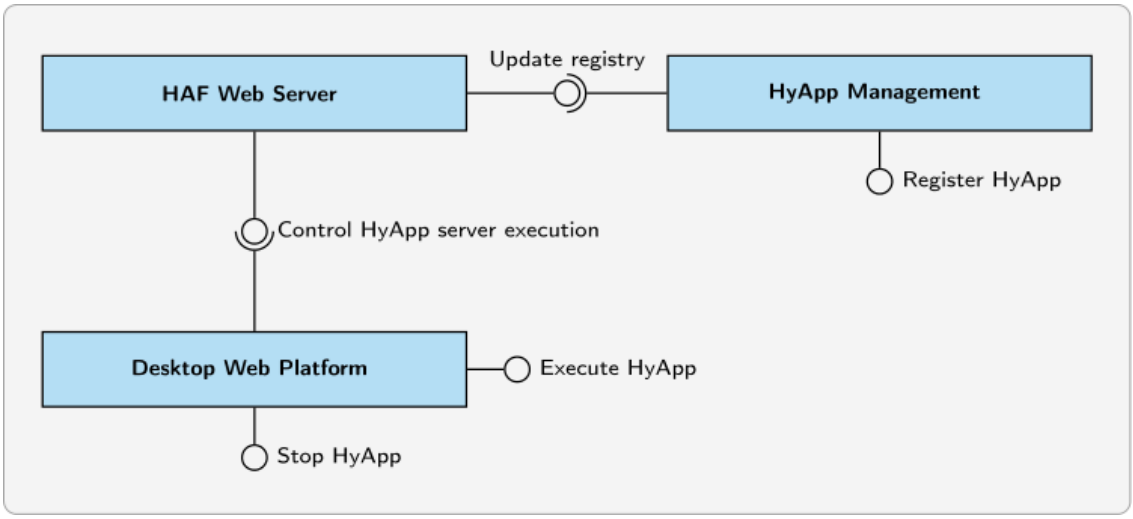
- **HAF Web Server**. Komponen ini adalah *web server* yang berfungsi sebagai *proxy* dari aplikasi server HyApp. Selain sebagai *proxy*, HAF Web Server juga mengendalikan pengeksekusian, pengaksesan, dan pengkonfigurasian server HyApp
- **HyApp Management**. Komponen ini adalah komponen yang digunakan oleh pengguna desktop untuk meregistrasikan HyApp. Saat meregistrasikan HyApp baru, komponen ini juga akan memberikan notifikasi kepada HAF Web Server untuk memperbarui daftar HyApp yang dimilikinya. Setelah diregistrasikan, HAF Web Server akan mengenali identitas HyApp dan akan dapat mengeksekusi aplikasi servernya.
- **Desktop Web Platform**. Komponen ini adalah *web browser* yang disediakan oleh HAF khusus untuk pengaksesan dalam modus desktop.

Untuk pengembangan HyApp, seperti yang ditunjukkan pada Gambar *3*, HAF menyediakan tiga paket API, yaitu:

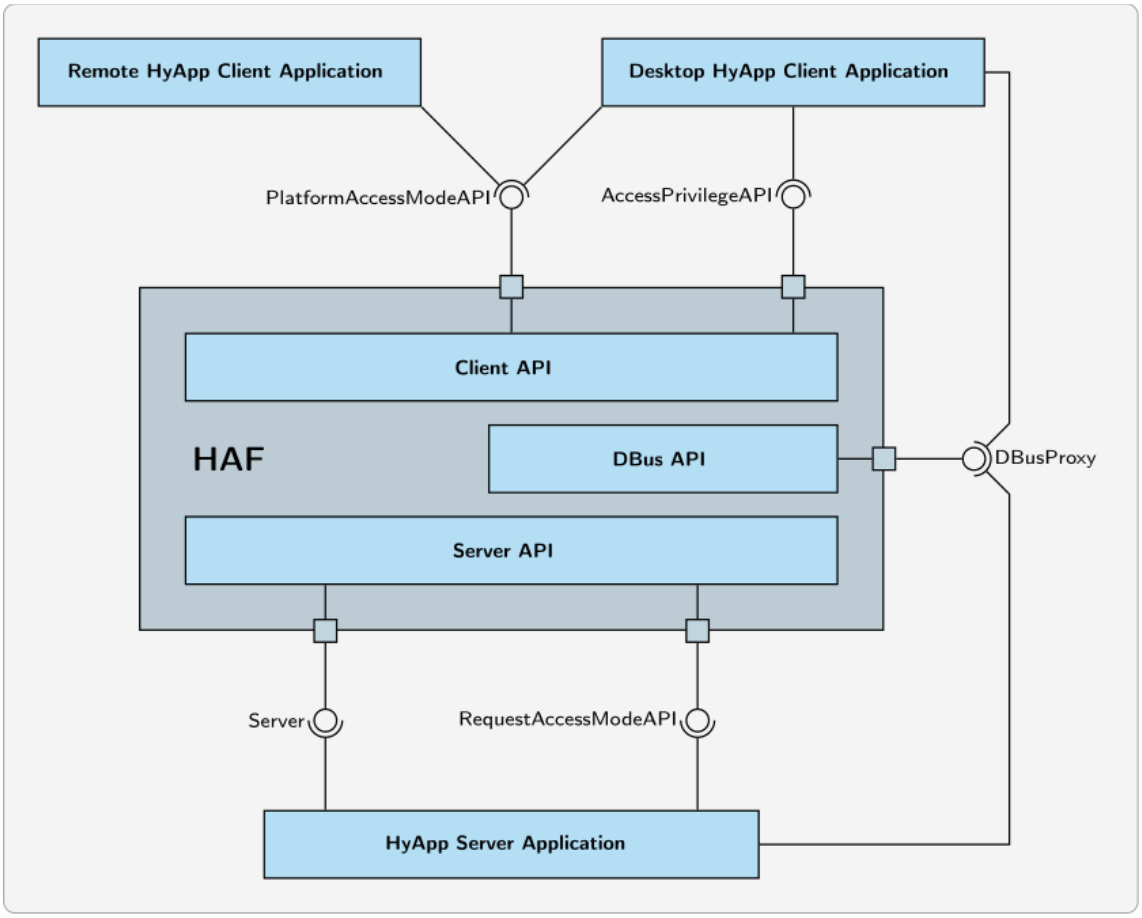
- **Client API**. API yang disediakan oleh HAF untuk digunakan oleh aplikasi klien HyApp. Melalui API ini, aplikasi klien HyApp dapat mengetahui modus pengaksesan yang sedang dilakukannya.
- **Server API**. API yang disediakan oleh HAF untuk digunakan oleh aplikasi server HyApp. Melalui Server API, aplikasi server HyApp akan dapat melakukan hal-hal sebagai berikut:
    - Mengimplementasikan server HTTP.
    - Mengetahui modus pengaksesan dari *request* yang akan diprosesnya.
- **DBus API**. API yang disediakan oleh HAF agar aplikasi klien HyApp yang berjalan dalam modus desktop dapat berkomunikasi dengan aplikasi desktop lainnya.



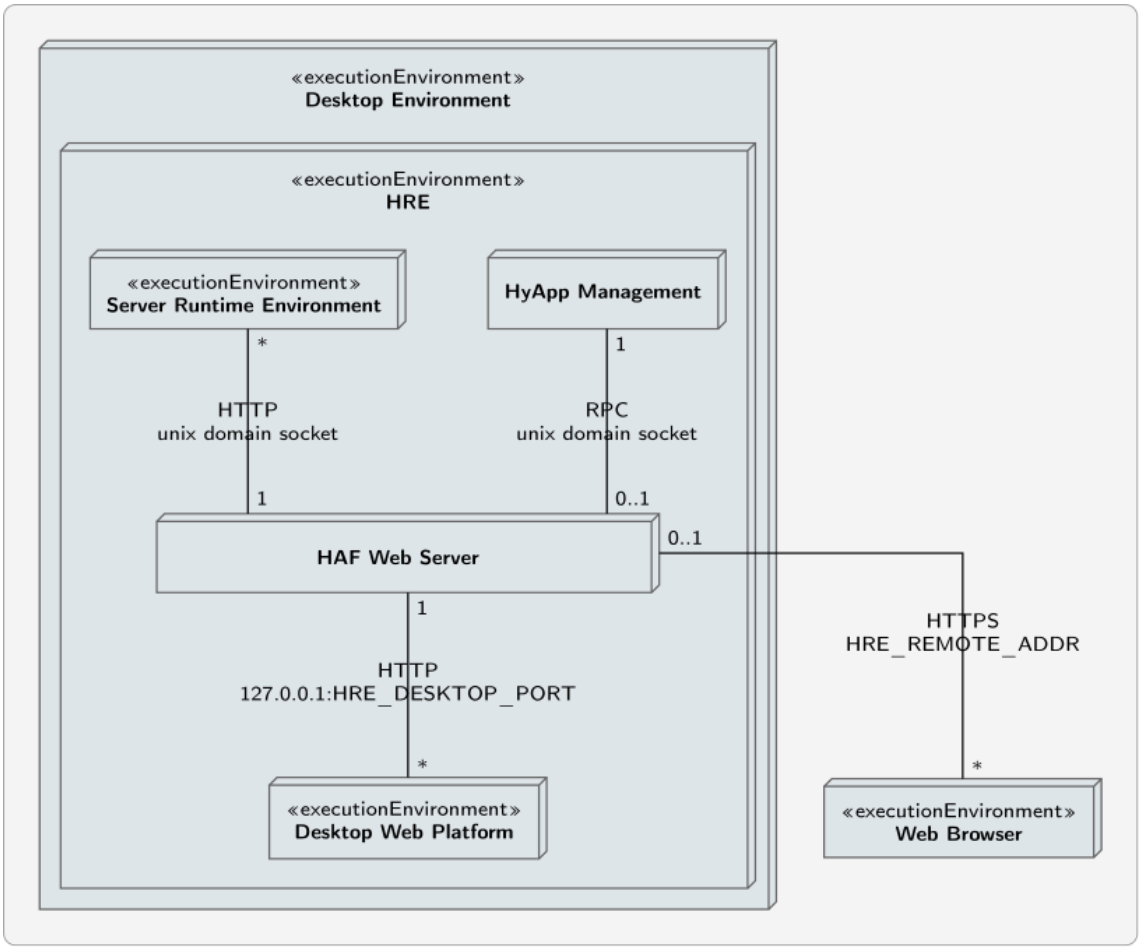
Gambar 1 Arsitektur HyApp



Gambar 2 Komponen-komponen lingkungan eksekusi HAF



Gambar 3 Komponen-komponen API



Gambar 4 Arsitektur lingkungan eksekusi HAF

Berdasarkan hasil pengujian, yang karena keterbatasan jumlah halaman tidak dapat disertakan pada naskah ini, dapat ditunjukkan bahwa melalui HAF, HyApp dapat diakses baik secara lokal maupun melalui jaringan, dapat menerapkan perilaku yang berbeda saat diakses melalui jaringan, serta dapat berkomunikasi dengan aplikasi desktop lainnya saat diakses secara lokal.

Kemampuan HyApp untuk dapat diakses baik secara lokal (modus desktop) maupun melalui jaringan (modus daring) dapat diwujudkan melalui kerjasama antar komponen-komponen yang ada dalam lingkungan eksekusi HAF. HAF Web Server yang merupakan *proxy* dari aplikasi server HyApp, memiliki dua buah antarmuka jaringan (seperti yang ditunjukkan pada Gambar 4). Kedua antarmuka jaringan tersebut masing-masing disediakan untuk melayani pengaksesan HyApp dalam modus desktop dan modus daring. Antarmuka jaringan yang melayani pengaksesan HyApp dalam modus desktop berada pada alamat IP lokal (127.0.0.1) dengan TCP *port* yang dapat dikonfigurasi melalui *environment variable* `HRE_DESKTOP_PORT`. Sedangkan antarmuka jaringan yang digunakan untuk melayani pengaksesan dalam modus daring berada pada alamat IP dan TCP *port* yang dikonfigurasi melalui *environment variable* `HRE_REMOTE_ADDR`.

Dalam HAF, HyApp hanya dapat diakses dalam modus desktop oleh pengguna desktop melalui Desktop Web Platform, sehingga HAF Web Server dapat memastikan bahwa setiap *request* yang ditujukan untuk mengakses HyApp dalam modus desktop pastilah berasal dari Desktop Web Platform. HAF Web Server mengeksekusi aplikasi server HyApp saat menerima *request* yang berasal dari Desktop Web Platform, dan aplikasi server dari HyApp tersebut tidak sedang berjalan. Sedangkan penghentian pengeksekusian aplikasi server HyApp dilakukan oleh

HAF Web Server saat Desktop Web Platform yang mengakses HyApp tersebut juga dihentikan pengeksekusiannya. Dengan cara tersebut, HAF dapat mengeksekusi aplikasi server HyApp secara efisien, yaitu hanya saat dibutuhkan oleh pengguna desktop.

Melalui Desktop Web Platform, HAF dapat menentukan modus pengaksesan dari HyApp. Dalam HAF, *request* dikatakan berasal dari pengaksesan dalam modus desktop jika dan hanya jika *request* tersebut berasal dari Desktop Web Platform. Aplikasi klien HyApp dikatakan berjalan dalam modus desktop jika dan hanya jika diakses/dijalankan di atas Desktop Web Platform. HyApp dapat mengetahui informasi modus pengaksesan tersebut melalui API yang disediakan HAF. Dengan menggunakan Client API, aplikasi klien HyApp dapat mengetahui modus pengaksesan yang sedang dilakukannya. Sedangkan dengan menggunakan Server API, aplikasi server HyApp dapat mengetahui modus pengaksesan dari *request* yang diterimanya. Melalui informasi modus pengaksesan yang disediakan oleh kedua API tersebut, HyApp memiliki kemampuan untuk menerapkan perilaku yang berbeda berdasarkan modus pengaksesannya.

Karena dibangun menggunakan teknologi web, maka pengaksesan HyApp, baik secara lokal maupun melalui jaringan, harus dilakukan menggunakan *web browser*. *Web browser* tidak menyediakan antarmuka yang dapat digunakan oleh halaman web yang berjalan di atasnya untuk berkomunikasi dengan aplikasi desktop. Dalam HAF, pengaksesan HyApp secara lokal (dalam modus desktop) dilakukan menggunakan Desktop Web Platform. Agar aplikasi klien HyApp yang berjalan di atas Desktop Web Platform dapat berkomunikasi dengan aplikasi desktop, maka HAF menyediakan API yang disebut DBus API. DBus API tersebut berupa JavaScript *library* yang ditanamkan dalam lingkungan eksekusi JavaScript pada Desktop Web Platform. Melalui DBus API tersebut, aplikasi klien HyApp yang berjalan di atas Desktop Web Platform dapat berkomunikasi dengan aplikasi desktop menggunakan protokol D-Bus.

Berdasarkan hasil penelitian ini, pada Tabel 1 dan Tabel 2 ditunjukkan perbedaan antara HyApp dengan aplikasi desktop dan aplikasi web. Dari tabel tersebut tampak bahwa HyApp memiliki karakteristik yang merupakan gabungan dari karakteristik aplikasi desktop dan karakteristik aplikasi web.

Tabel 1 Perbedaan HyApp dengan aplikasi desktop dan aplikasi web

APLIKASI DESKTOP	APLIKASI WEB	HYAPP
Arsitektur		
tergantung kebutuhan	2-*layer* atau N-*layer*	didasarkan pada arsitektur *client/server* (2-*layer*)
GUI		
dibangun menggunakan GUI *toolkit*	dibangun dengan teknologi web	dibangun dengan teknologi web
Pengaksesan		
lokal	melalui jaringan dengan menggunakan *web browser*	lokal dengan menggunakan *web browser* yang disediakan HAF dan melalui jaringan dengan menggunakan *web browser* biasa

Tabel 2 Perbedaan HyApp dengan aplikasi desktop dan aplikasi web (lanjutan)

APLIKASI DESKTOP	APLIKASI WEB	HYAPP
Pengeksekusian		
dieksekusi dalam lingkungan desktop dan dikendalikan oleh pengguna desktop	server berjalan dalam lingkungan eksekusi yang ditentukan dan oleh *system administrator*, dengan konfigurasi yang ditentukan secara manual, dan pengeksekusian yang dilakukan sedini mungkin agar selalu siap melayani *request*	server berjalan dalam lingkungan desktop, dengan pengeksekusian dan pengonfigurasiannya dikendalikan oleh HAF dan hanya dilakukan saat pengguna desktop akan menggunakannya
Mekanisme proteksi		
pengguna sebagai domain	ditentukan dan diterapkan oleh aplikasi	dalam modus desktop menggunakan mekanisme proteksi dengan pengguna sebagai domain, dalam modus daring dapat menerapkan mekanisme proteksinya sendiri
Komunikasi		
dapat berkomunikasi dengan aplikasi desktop lainnya	tidak dapat berkomunikasi dengan aplikasi desktop namun dapat berkomunikasi dengan aplikasi web lainnya melalui jaringan	dapat berkomunikasi dengan aplikasi web lainnya, dan saat diakses dalam modus desktop, dapat berkomunikasi secara lokal dengan aplikasi desktop lainnya

4. KESIMPULAN

Dalam penelitian ini dapat dibangun purwarupa *framework* yang diberi nama HAF (*Hybrid Application Framework*). HAF merupakan *framework* untuk mengembangkan dan mengeksekusi jenis aplikasi desktop yang untuk selanjutnya disebut sebagai HyApp (*Hybrid Application*). Berdasarkan pengujian yang dilakukan, dapat ditunjukkan bahwa:

1. Melalui *proxy* yang disediakan HAF, HyApp dapat diakses dalam dua modus pengaksesan, yaitu secara lokal dengan menggunakan *web browser* khusus yang disediakan HAF, serta melalui jaringan dengan menggunakan *web browser* biasa.
2. HyApp dapat menerapkan perilaku yang berbeda berdasarkan informasi modus pengaksesan yang didapatkan melalui API yang disediakan oleh HAF.
3. Aplikasi HyApp yang diakses secara lokal dapat berkomunikasi dengan aplikasi desktop lainnya melalui API yang disediakan HAF. HAF menanamkan API tersebut pada lingkungan eksekusi JavaScript dalam *web browser* yang disediakannya khusus untuk pengaksesan HyApp secara lokal.

5. SARAN

Pada purwarupa HAF yang dikembangkan dalam penelitian ini masih belum terdapat mekanisme untuk mendeteksi kebergantungan antara sebuah HyApp dengan HyApp lainnya. Untuk penelitian selanjutnya diharapkan dapat dikembangkan mekanisme tersebut sehingga nantinya HAF akan dapat mengeksekusi server HyApp tidak hanya saat dibutuhkan oleh pengguna desktop tetapi juga saat dibutuhkan oleh HyApp lainnya.

DAFTAR PUSTAKA

[1] Puder, A., 2004, Extending Desktop Applications to the Web, *ISICT '04 Proceedings of the 2004 International Symposium on Information and Communication Technologies*, Dublin.

[2] Gitzel, R., Korthaus, A. dan Schader, M., 2007, Using Established Web Engineering Knowledge In Model-Driven Approaches, *Science of Computer Programming*, 66, 2, pp. 105–124.

[3] Lamberti, F. dan Sanna, A., 2008, Extensible GUIs for Remote Application Control on Mobile Devices, *Computer Graphics and Applications, IEEE*, 28, 4, pp. 50–57.

[4] Zepeda, J. dan Chapa, S., 2007, From Desktop Applications Towards AJAX Web Applications, *Fourth International Conference on Electrical and Electronics Engineering, 2007. ICEEE 2007.* IEEE, pp. 193–196.

[5] Vieriu, V. dan Tuican, C., 2009, Adobe AIR, Bringing Rich Internet Applications to the Desktop, *Ann. Univ. Tibiscus Comp. Sci. Series*, VII, pp. 367–380.

[6] Google, 2013, *What Are Chrome Apps?*, Google, http://developer.chrome.com/apps/about_apps.html, diakses tanggal 26 Okt. 2013.

[7] TideSDK Team, 2012, *Create Multi-Platform Desktop Apps with HTML5, CSS3 and JavaScript*, TideSDK Team, http://www.tidesdk.org/, diakses tanggal 6 Jun. 2013.