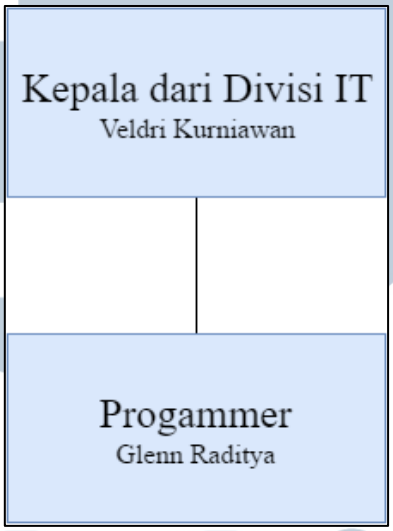
BAB III

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Praktik kerja magang yang dilaksanakan di CV Ciptaloka Intermedia Nusantara memiliki struktur kerja seperti Gambar 3.1.



Gambar 3.1 Struktur Koordinasi Pelaksanaan Kerja Magang

Veldri Kurniawan selaku kepala dari divisi IT memimpin dan mengkoordinasikan praktik kerja magang. Selama melakukan kerja magang, kepala dari divisi IT mengawasi, membimbing, dan mengarahkan pengerjaan *widget* dan aplikasi *print server*.

Kerja magang dilakukan sendiri. Koordinasi dengan kepala dari divisi IT dilakukan secara langsung dengan menunjukkan *progress* aplikasi yang dibangun. Jika ada tambahan atau revisi, diberitahukan secara langsung.

Untuk setiap aplikasi yang dikerjakan, diberikan jangka waktu pengerjaannya. Ketika jangka waktu yang ditentukan sudah tiba, akan dilakukan *testing* aplikasi oleh kepala dari divisi IT. Setelah dilakukan *testing*, kepala dari divisi IT akan memberikan revisi dan ide-ide penambahan fitur lainnya.

3.2 Tugas yang Dilakukan

Tugas yang pertama adalah pembuatan *widget* untuk *website* ciptaloka.com. *Widget* yang dirancang dan dibangun ini dibuat untuk membantu para desainer memasarkan hasil desainnya di suatu website di luar *website* ciptaloka.com. Para desainer dapat membuat *widget* dengan berbagai ukuran sesuai dengan *website* yang akan digunakan untuk memasarkan. *Widget* yang telah dibuat juga akan melakukan pembaharuan secara otomatis jika ada desain baru yang dibuat. *User Interface* pada *widget* ini dibuat menggunakan HTML dan Javascript. Fungsionalitas *widget* dibuat menggunakan *framework* Mithril dengan bahasa pemrograman Javascript dan JSON yang disediakan oleh API untuk memperoleh data yang diperlukan.

Tugas yang kedua adalah pembuatan aplikasi *print server* untuk mencetak *invoice* dan *label* dari setiap pemesanan produk yang sudah dibuat secara otomatis. *Print server* dapat digunakan di komputer Windows yang mempunyai Adobe Acrobat, dapat digunakan secara *portable*, tidak perlu melakukan instalasi terlebih dahulu, tidak mempunyai batas maksimum mencetak, dan dapat mencetak secara otomatis selama ada berkas yang disediakan di dalam JSON. *User Interface* dan fungsionalitas pada aplikasi ini dibuat menggunakan *Windows form*

dengan bahasa pemrograman C#, *framework* Json.NET, dan JSON yang disediakan oleh API untuk memperoleh data yang diperlukan.

Kedua tugas yang diberikan dibuat berdasarkan rancangan dasar yang telah diberikan oleh kepala dari divisi IT.

Rincian tugas yang dilakukan dalam kerja magang ini dapat dilihat pada Tabel 3.1.

Tabel 3.1 Realisasi Kerja Magang

Minggu	Pekerjaan yang Dilakukan
1	• *Training* penggunaan Mithril
2	• Merancang sistem *widget* • Merancang *user interface* untuk *widget*
3	• Membuat *widget*
4	• *Testing* dan revisi *widget*
5	• Training penggunaan Json.NET • Merancang sistem aplikasi *print server* • Merancang *user interface* untuk aplikasi *print server*
6	• Membuat aplikasi *print server*
7	• *Testing* dan revisi aplikasi *print server*
8	• *Finisihing widget* dan aplikasi *print server*

3.3 Uraian Pelaksanaan Kerja Magang

Pelaksanaan kerja magang dibagi menjadi beberapa bagian, yaitu proses pelaksanaan, kendala yang ditemukan, dan solusi atas kendala yang ditemukan.

3.3.1 Proses Pelaksanaan

Pembuatan *widget* dibuat menggunakan perangkat lunak dan keras. Perangkat lunak yang digunakan adalah sebagai berikut.

1. *Framework* Mithril versi 0.2.5

Mithril merupakan sebuah *framework* dengan bahasa pemrograman Javascript. Berdasarkan dokumentasi, Mithril memiliki kelebihan dibandingkan dengan *framework* Javascript lain seperti jQuery, Angular, Backbone, dan React. Kelebihan tersebut adalah kecepatan saat memuat konten Javascript ke dalam sebuah halaman *web*.

2. Sublime Text 3 versi 3103

Sublime Text merupakan program *text editor* yang digunakan dalam pembangunan *widget*.

3. Windows 10 Pro 64 bit

Windows 10 merupakan sistem operasi yang digunakan dalam pembangunan *widget*.

Pembuatan aplikasi *print server* dibuat menggunakan perangkat lunak dan keras. Perangkat lunak yang digunakan adalah sebagai berikut.

1. Microsoft Visual Studio 2010

Microsoft Visual Studio 2010 merupakan program yang digunakan dalam pembangunan aplikasi *print server*.

2. *Framework* Json.NET

Json.NET merupakan *framework* untuk mengolah JSON di dalam .NET yang paling banyak digunakan karena kemudahannya dalam penggunaan dan

memiliki performa yang lebih tinggi dibanding *framework* untuk mengolah JSON di dalam .NET seperti DataContractJsonSerializer dan JavaScriptSerializer.

3. Windows 10 Pro 64 bit

Windows 10 merupakan sistem operasi yang digunakan dalam pembangunan aplikasi *print server*.

Selain itu, perangkat keras yang digunakan untuk membuat *widget* dan aplikasi *print server* adalah *laptop* Lenovo G400s dengan spesifikasi sebagai berikut.

1. Processor Intel® Core™ i5-3230M CPU @ 2.60 GHz
2. Memori RAM 8 GB
3. VGA NVIDIA GeForce 720M

Pembuatan *widget* dan aplikasi *print server* dilakukan dalam beberapa tahap, yaitu perancangan sistem, *user interface*, dan implementasi.

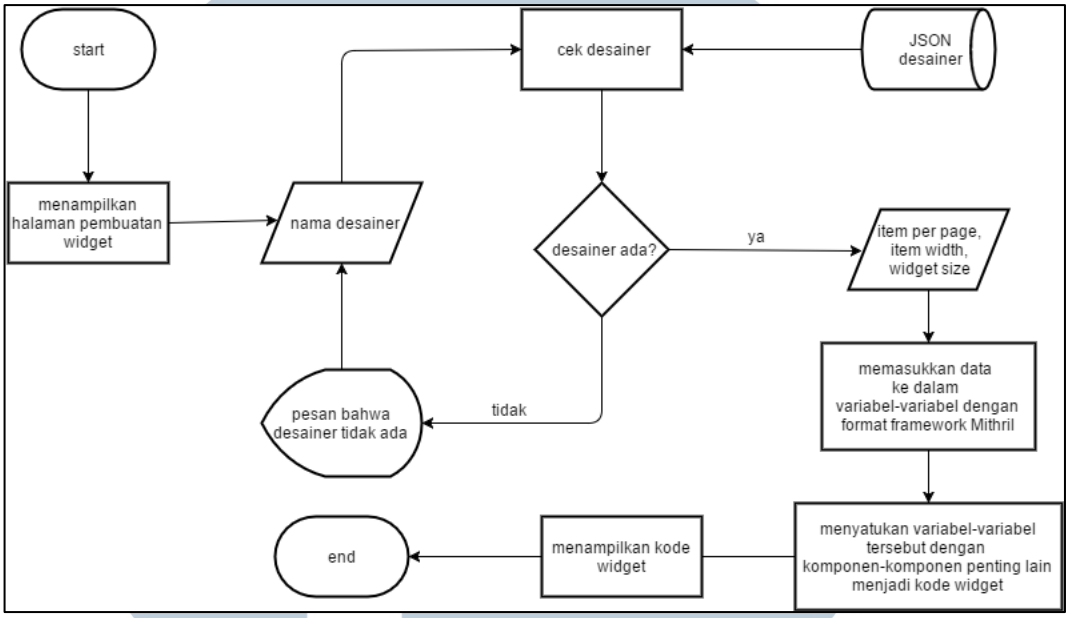
A. Perancangan Sistem

Pada tahap ini dilakukan pembuatan *flowchart*, perancangan format JSON, dan perancangan *user interface*.

A.1 Flowchart

Flowchart merupakan diagram yang digunakan untuk menggambarkan urutan kerja dari suatu sistem. Berikut *flowchart* dari aplikasi *widget* dan *print server* pada CV Ciptaloka Intermedia Nusantara.

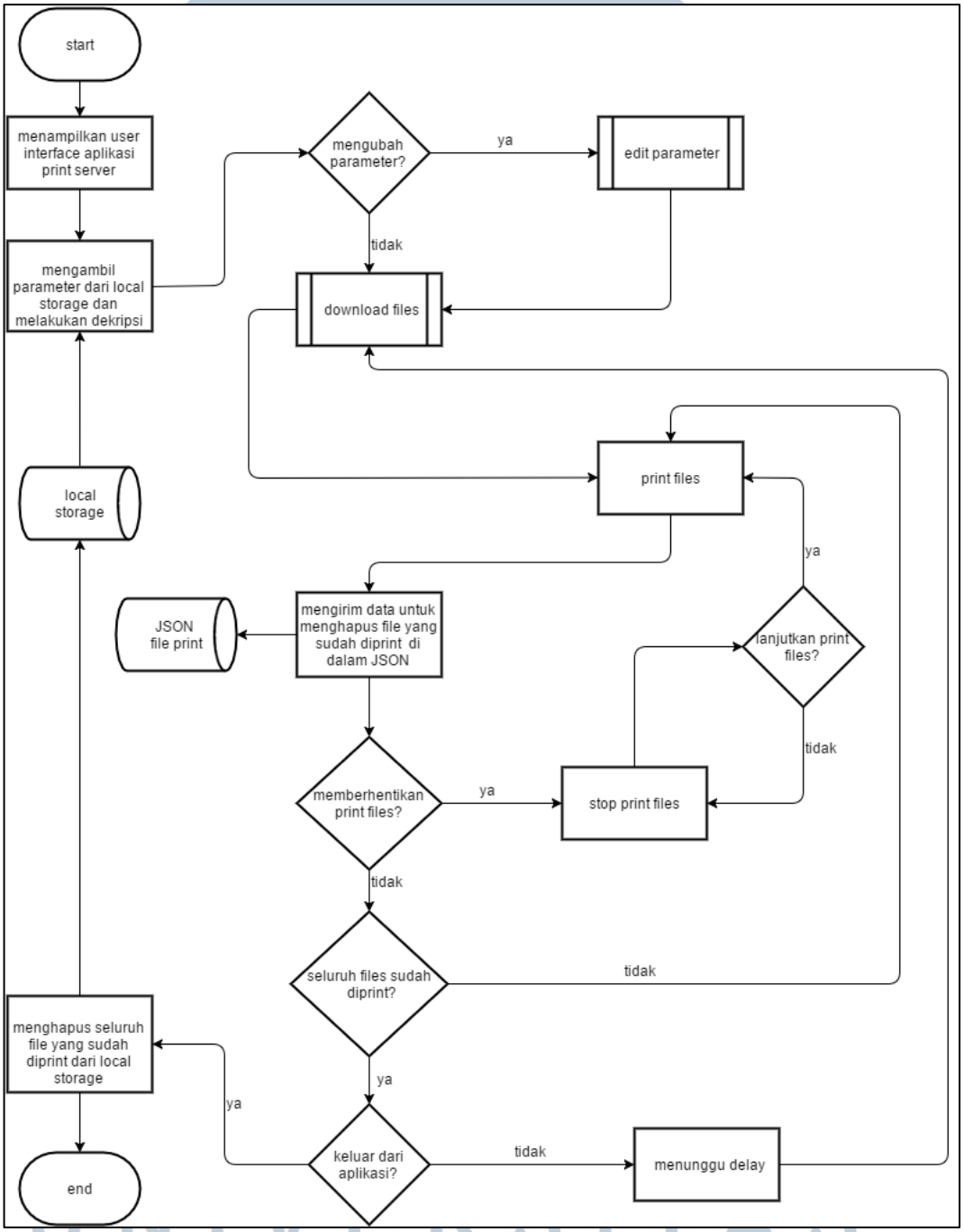
A.1.1Flowchart Widget



Gambar 3.2 Flowchart *Widget*

Gambar 3.2 merupakan *flowchart* dari *Widget.* Dimulai dengan tampilan halaman pembuatan *widget,* kemudian desainer memasukkan nama desainer. Jika nama desainer yang dimasukkan tidak ada di dalam JSON yang disediakan oleh API, akan ditampilkan pesan yang menyatakan bahwa desainer tidak ada sehingga kode tidak bisa disalin. Jika nama desainer yang dimasukkan ada di dalam JSON yang disediakan oleh API, data *item per page, item width,* dan *widget size* dimasukkan. Data tersebut dimasukkan ke dalam variabel-variabel dengan format *framework* Mithril. Setelah itu, variabel-variabel tersebut disatukan dengan komponen-komponen penting lain menjadi kode *widget* yang akan ditampilkan di halaman *web*.

A.1.2Flowchart Aplikasi Print Server

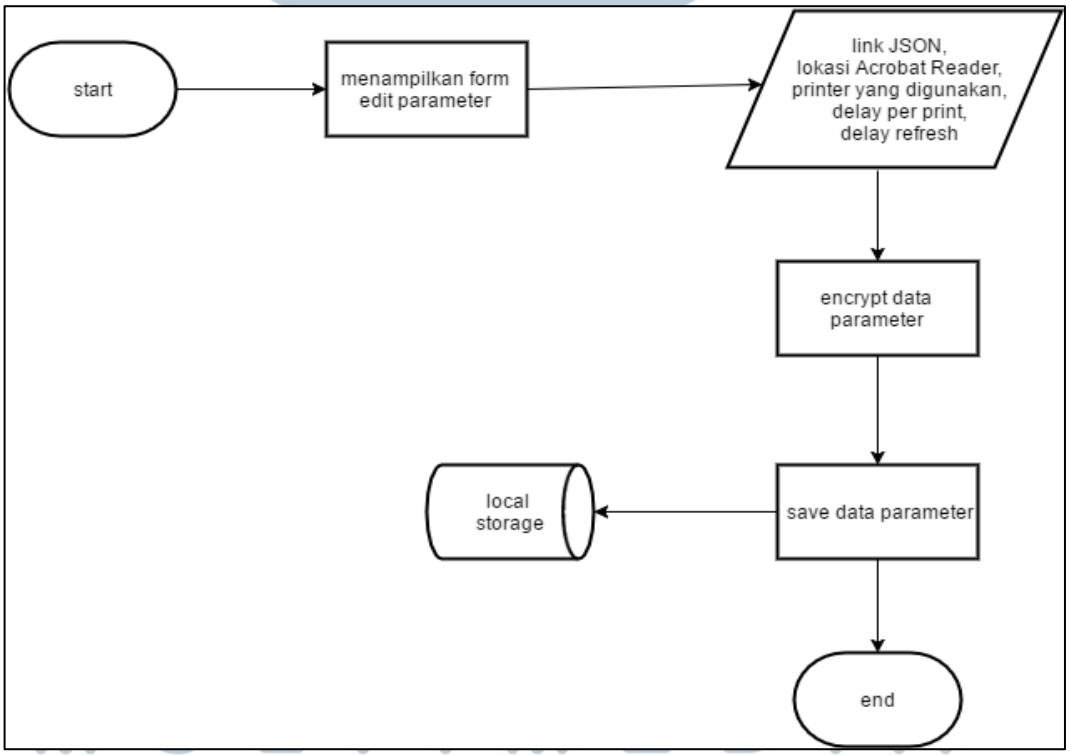


Gambar 3.3 Flowchart Aplikasi *Print Server*

Gambar 3.3 merupakan *flowchart* dari keseluruhan aplikasi *print server*. Dimulai dengan menampilkan *user interface* aplikasi. Lalu parameter yang

tersedia di *local storage* diambil dan didekripsi. Jika ingin mengubah parameter, parameter diubah lalu *files* yang ingin dicetak langsung diunduh. Jika tidak ingin mengubah parameter, *files* yang ingin dicetak langsung diunduh. Setelah *file-file* selesai diunduh, *file-file* tersebut dicetak. Setiap berhasil mencetak satu *file*, aplikasi akan mengirimkan data ke JSON yang disediakan oleh API untuk menghapus data *file* yang sudah dicetak. Jika tidak ingin memberhentikan pencetakan, proses mencetak akan dilakukan sampai semua *file* selesai dicetak. Jika sudah selesai dan keluar dari aplikasi, aplikasi akan menghapus seluruh *file* yang sudah diunduh di dalam *local storage.*

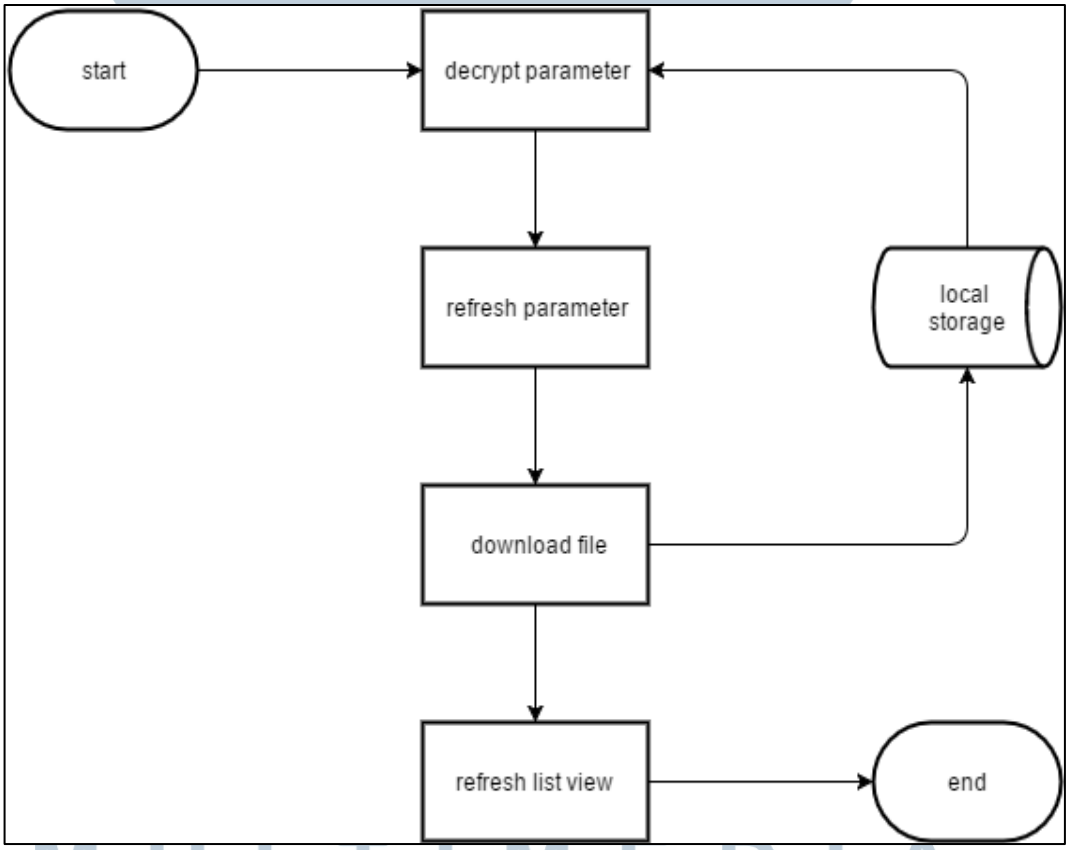
A.1.3 Flowchart Edit Parameter Pada Aplikasi Print Server



Gambar 3.4 Flowchart *Edit* Parameter Pada Aplikasi *Print Server*

Gambar 3.4 merupakan flowchart *edit parameter* pada aplikasi *print server*. Dimulai dengan menampilkan *form* untuk memasukkan data-data parameter. Kemudian data-data tersebut, yaitu *link* JSON untuk mengunduh *file*, lokasi dari *Acrobat Reader*, pilihan *printer* yang digunakan untuk mencetak, *delay per print*, dan *delay refresh* dimasukkan ke dalam *form* tersebut. Setelah itu data-data yang sudah dimasukkan akan dienkripsi dan disimpan ke dalam *file* teks di *local storage*.

A.1.4 Flowchart Download Files Pada Aplikasi Print Server



Gambar 3.5 Flowchart *Download Files* Pada Aplikasi *Print Server*

Gambar 3.5 merupakan flowchart download files. Pertama–tama, file parameter terenkripsi sebelumnya yang disimpan di dalam *local storage* akan diambil dan didekripsi. Setelah itu, parameter yang telah didekripsi akan di-*refresh* oleh aplikasi ke dalam variabel-variabel di dalam aplikasi. Lalu dengan menggunakan data dari parameter tersebut, aplikasi akan mengunduh seluruh *file* dan disimpan ke dalam *local storage*. Terakhir, setelah semua *file* berhasil diunduh, aplikasi akan memperbarui *list view* yang berada di halaman depan aplikasi berdasarkan *file-file* yang diunduh.

A.2 Perancangan Format JSON

JSON (*JavaScript Object Notation)* yang disediakan oleh API digunakan untuk pertukaran data pada *widget* dan *print server*. Berikut adalah format dari JSON yang digunakan.

A.2.1 Format JSON Pada Widget

```
{
    "_meta": {
        "status": "[SUCCESS / FAILED]",
        "count": [JUMLAH ITEM]
    },
    "result": [
        {
            "hash": "[HASH UNIK UNTUK SETIAP ITEM]",
            "title": "[NAMA ITEM]",
            "price": [HARGA ITEM],
            "url": "[LINK ITEM DARI WWW.CIPTALOKA.COM]",
            "image": "[LINK GAMBAR ITEM]"
        }
    ]
}
```

Gambar 3.6 Format JSON Pada *Widget*

Gambar 3.6 merupakan format dari JSON yang digunakan dalam *widget*. Terdapat objek "*_meta*" yang merupakan *array*. *Array* tersebut mempunyai dua data, yaitu "*status*" yang menandakan apakah JSON tersebut dapat diakses atau tidak dan "*count*" yang berisi jumlah objek dari *array* "*result*".

Pada *array* kedua, yaitu "*result*", terdapat objek yang mempunyai lima data, yaitu "*hash*" yang berisi *hash* unik untuk menandakan setiap *item*, "*title*" berisi nama dari *item*, "*price*" berisi harga dari *item*, "*url*" berisi *link* dari *item* yang jika ditekan dapat langsung mengarahkan ke *website* ciptaloka.com, dan "*image*" yang berisi *link* dari gambar item. Jumlah objek pada *array* "*result*" tergantung pada banyaknya *item* yang dimiliki oleh desainer yang diminta pada API.

A.2.2 Format JSON Pada Aplikasi Print Server

```
{
    "_meta": {
        "status": "[SUCCESS / FAILED]",
        "count": [JUMLAH ITEM]
    },
    "result": [
        {
            "id": [ID],
            "file": "[LINK FILE YANG INGIN DICETAK]"
        }
    ]
}
```

Gambar 3.7 Format JSON Pada Aplikasi *Print Server*

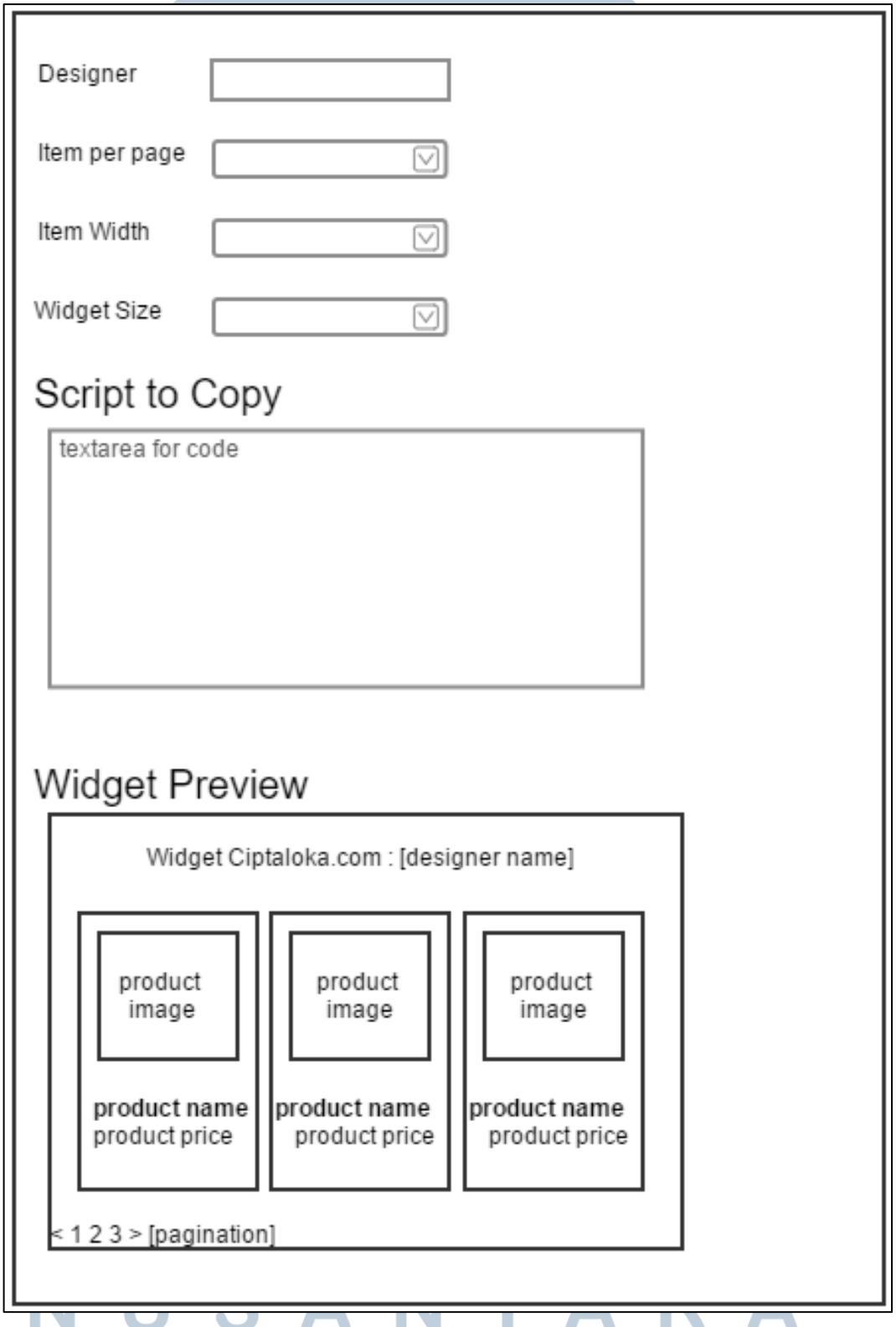
Gambar 3.7 merupakan format dari JSON yang digunakan dalam aplikasi *print server*. Terdapat objek "*_meta*" yang merupakan *array*. Sama seperti JSON yang digunakan oleh aplikasi *widget*, *Array* tersebut mempunyai dua data, yaitu "*status*" yang menandakan apakah JSON tersebut ada atau tidak dan "*count*" yang berisi jumlah objek dari *array* "*result*".

Pada *array* kedua, yaitu "*result*", terdapat objek yang mempunyai dua data, yaitu "*id*" yang berisi ID dari setiap *file* yang ingin dicetak dan "*file*" yang berisi *link* dari *file* yang ingin dicetak. Jumlah objek pada *array* "*result*" tergantung pada banyaknya *file* yang disediakan oleh API untuk dicetak.

A.3 Perancangan Antarmuka

Dalam pembuatan *widget* dan aplikasi *print server* dilakukan perancangan antarmuka terlebih dahulu. Terdapat masing-masing satu rancangan antarmuka yang digunakan.

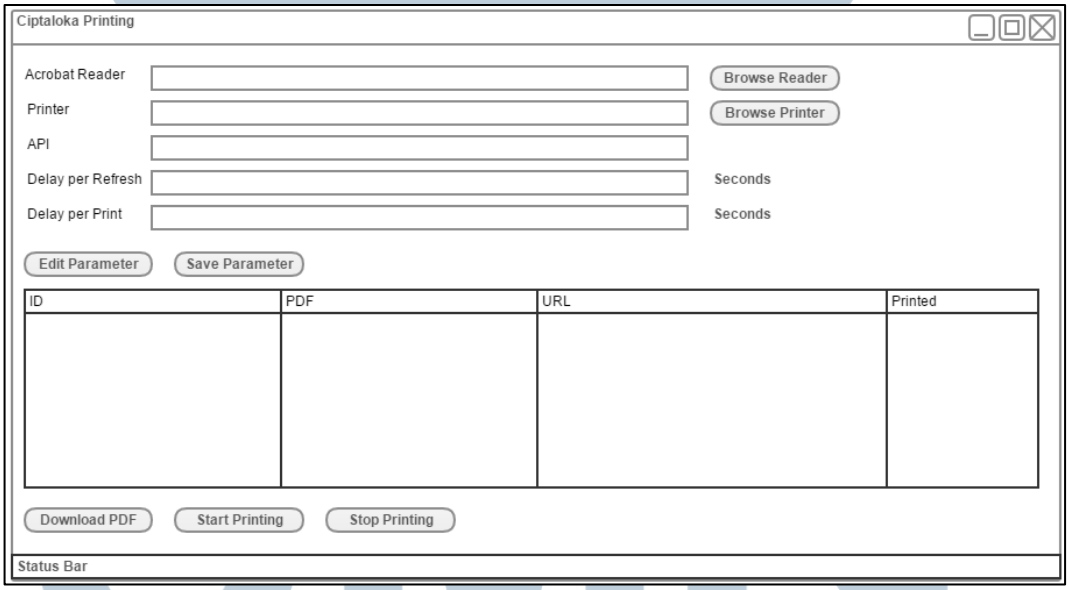
A.3.1 Perancangan Antarmuka Widget



Gambar 3.8 Rancangan Antarmuka *Widget*

Gambar 3.8 merupakan rancangan untuk halaman pembuatan *widget*. Terdapat *form* untuk memasukkan data. Untuk nama desainer menggunakan *textbox*. Untuk *item per page*, *item width*, dan *widget size* menggunakan *combobox* dengan nilai yang sudah ditentukan. Lalu untuk *script* kode *widget* menggunakan *textarea*. Terakhir untuk tampilan *widget* menggunakan tabel yang berisi gambar produk desainer, nama produk dan harga produk, serta tombol untuk *pagination*.

A.3.2 Perancangan Antarmuka Aplikasi Print Server



Gambar 3.9 Rancangan Antarmuka Aplikasi *Print Server*

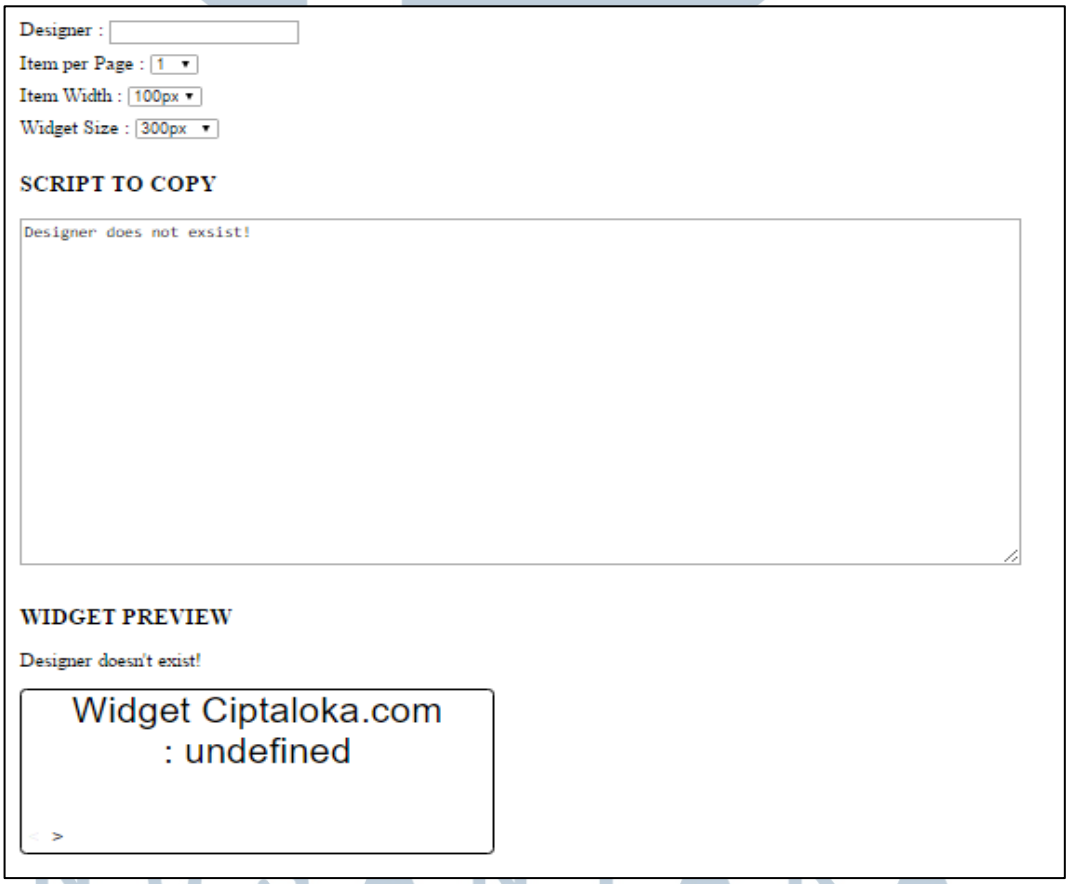
Gambar 3.9 merupakan rancangan untuk aplikasi *print server*. Terdapat *header* dan *button* untuk navigasi. Terdapat *form* dengan menggunakan *textbox* untuk mengisi data-data parameter. Untuk parameter Acrobat Reader dan *printer* harus menggunakan *button browse* yang terdapat di sebelah kanan *textbox* untuk

mengisi nilai. Lalu terdapat *button* untuk *edit* dan *save* parameter. Terdapat *list view* untuk menampilkan *file-file* yang sudah diunduh dan status apakah sudah dicetak atau belum. Kemudian ada tiga *button* untuk mengunduh PDF, memulai mencetak, dan memberhentikan proses mencetak. Di bagian bawah terdapat *status bar* untuk menunjukkan keterangan dari kegiatan yang sedang dilakukan oleh aplikasi *print server*.

B. Implementasi

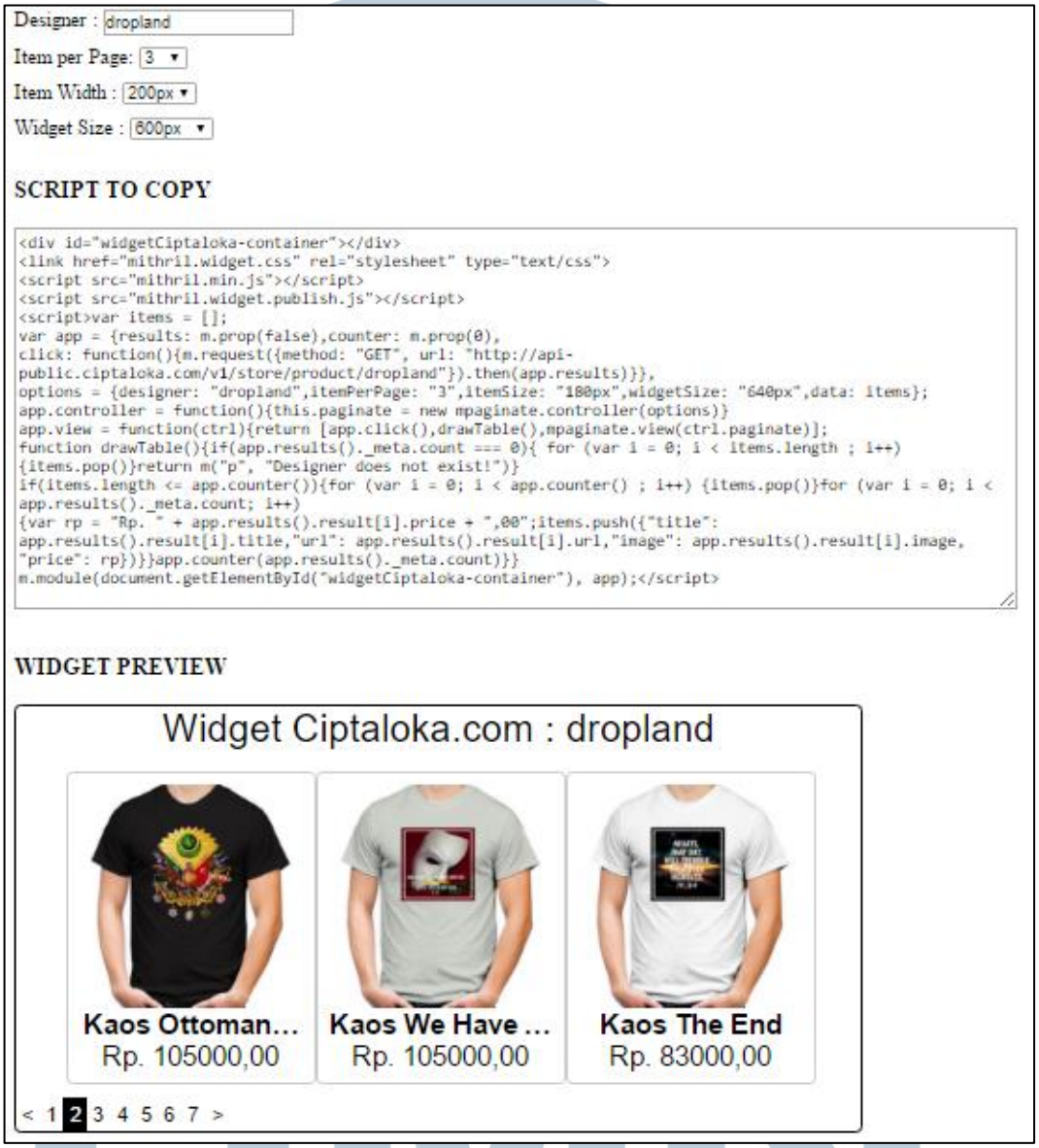
Berikut merupakan hasil implementasi *widget* dan aplikasi *print server*.

B.1 Implementasi Widget



Gambar 3.10 Halaman Awal *Widget*

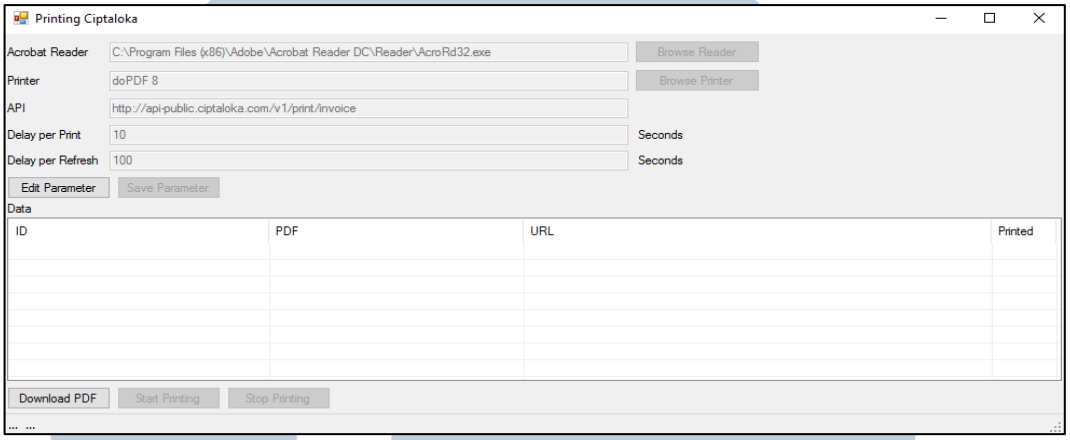
Implementasi *widget* masih polos hanya dengan Javascript karena ada divisi tersendiri, yaitu divisi *graphic designer* yang memiliki tugas untuk memperindah tampilan dari halaman *web* menggunakan CSS. Oleh karena itu, saat ini halaman pembuatan *widget* ini belum dimasukkan ke dalam *website* utama ciptaloka.com karena masih dilakukan penyesuaian dengan kode yang dibuat. Gambar 3.10 merupakan halaman awal sementara dari *widget.* Desainer harus mengisi data nama desainer, *item per page*, *item width*, dan *widget size* untuk mendapatkan kode *widget*. Jika nama desainer tidak tersedia pada JSON yang disediakan oleh API, pada bagian kode akan memunculkan pesan bahwa desainer tidak ada.



Gambar 3.11 Halaman Final *Widget*

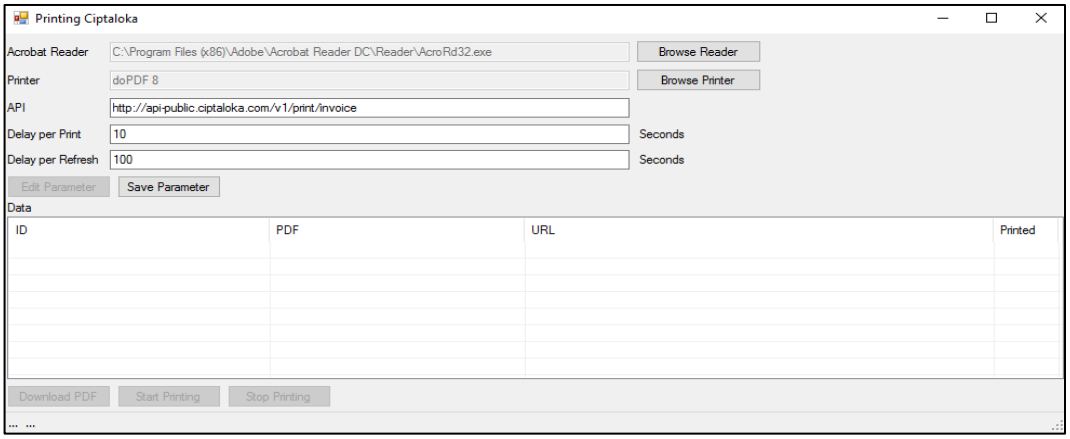
Jika nama desainer diisi sesuai dengan nama desainer yang tersedia di JSON yang disediakan oleh API, kode untuk *widget* akan muncul beserta *preview* dari *widget* tersebut seperti yang terlihat pada Gambar 3.11.

B.2 Implementasi Aplikasi Print Server



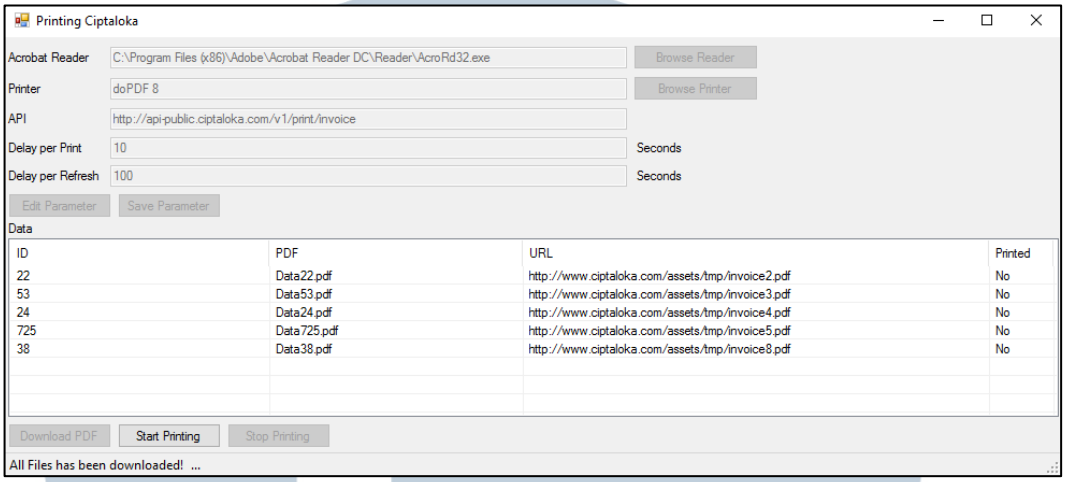
Gambar 3.12 Tampilan Awal Aplikasi *Print Server*

Gambar 3.12 merupakan tampilan awal dari aplikasi *print server*. Data parameter secara otomatis langsung diambil dari *local storage* dan dimasukkan ke dalam *form* parameter.



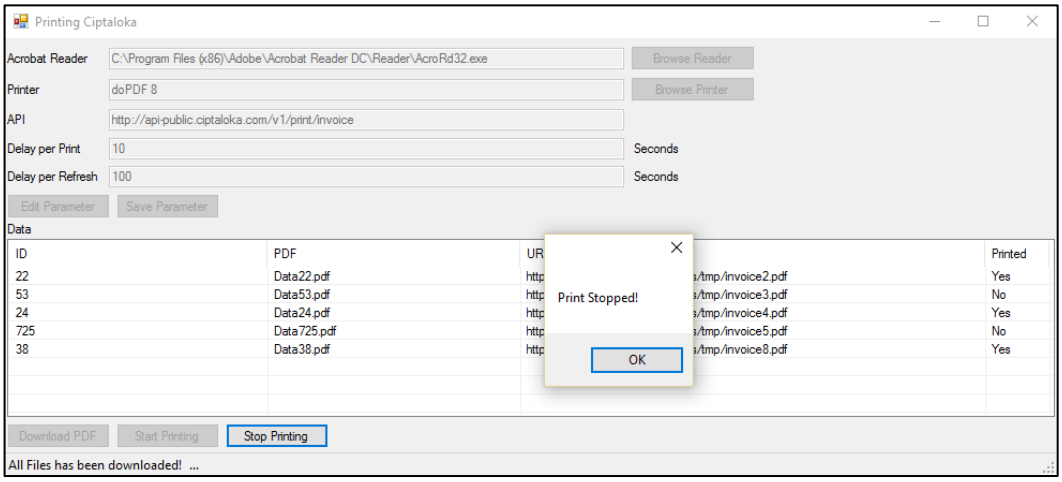
Gambar 3.13 Tampilan Edit Parameter Aplikasi *Print Server*

Untuk mengubah nilai dari parameter, tekan tombol Edit Parameter. Lalu *form* untuk mengubah parameter dapat diakses seperti yang terlihat pada Gambar 3.13. Jika sudah selesai melakukan perubahan parameter, tekan tombol Save Parameter untuk menyimpan perubahan.



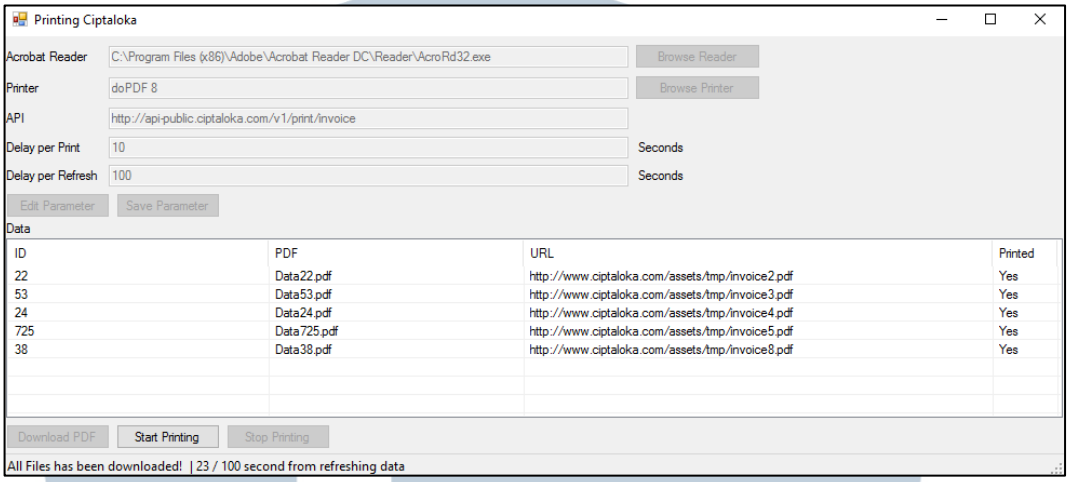
Gambar 3.14 Tampilan *Download* Aplikasi *Print Server*

Untuk mengunduh *file* yang akan dicetak, tekan tombol Download PDF. Setelah semua *file* berhasil diunduh, *list view* akan menampilkan seluruh informasi dari *file* yang sudah diunduh seperti pada Gambar 3.14.



Gambar 3.15 Tampilan *Stop* Aplikasi *Print Server*

Untuk mencetak *file-file* yang sudah diunduh, tekan tombol Start Printing. *File* yang sudah dicetak akan terlihat pada *list view* bagian *Printed* berubah menjadi *Yes*. Jika ingin memberhentikan proses mencetak, tekan tombol Stop Printing seperti pada Gambar 3.15.



Gambar 3.16 Tampilan *Refresh* Aplikasi *Print Server*

Jika seluruh *file* sudah berhasil dicetak, aplikasi akan menunggu selama nilai *Delay per Refresh* sesuai dengan yang dimasukkan pada *form* parameter untuk kembali mengunduh *file* dari JSON yang disediakan oleh API. Informasi *delay* bisa dilihat pada bagian *status bar* di bagian bawah aplikasi seperti pada Gambar 3.16. *Refresh* dapat langsung dilakukan tanpa menunggu waktu *delay* dengan menekan tombol Start Printing.

3.3.2 Kendala yang Ditemukan

Kendala yang ditemukan saat pengerjaan *widget* dan aplikasi *print server* adalah sebagai berikut.

1. *Framework* Mithril yang masih baru sehingga belum banyak dokumentasi dan tutorial yang tersedia di internet.
2. Kode *script* untuk hasil *widget* masih terbilang panjang karena belum menguasai framework Mithril dengan sempurna.

3. Sulit dalam *testing* untuk mencetak *file* pada aplikasi *print server* karena keterbatasan *printer*.

3.3.3 Solusi atas Kendala yang Ditemukan

Solusi atas kendala yang ditemukan dalam menjalankan kerja magang adalah sebagai berikut.

1. Membuat percobaan-percobaan kecil dan mempelajari lebih dalam tentang cara kerja maupun hal-hal yang perlu diketahui seputar *framework* Mithril.
2. Menghapus ruang kosong sebisa mungkin pada kode *script* untuk hasil widget sehingga terlihat lebih pendek.
3. Melakukan *testing* menggunakan *printer virtual*.