|  |  |
| --- | --- |
| logo unindra | |
| **SISTEM PEMESANAN MAKANAN DAN MINUMAN DI OSAKA RAMEN DEPOK BERBASIS JAVA** | |
| Tugas Akhir  diajukan untuk melengkapi  persyaratan mencapai  gelar sarjana | |
| NAMA  NPM | : ROBBY AWALDI  : 201543501022 |
| **PROGRAM STUDI INFORMATIKA**  **FAKULTAS TEKNIK DAN ILMU KOMPUTER**  **UNIVERSITAS INDRAPRASTA PGRI**  **2019** | |

**LEMBAR PERSETUJUAN UJIAN SKRIPSI/TUGAS AKHIR**

|  |  |  |
| --- | --- | --- |
| Nama | : | Robby Awaldi |
| NPM | : | 201543501022 |
| Fakultas | : | Teknik dan Ilmu Komputer |
| Program Studi | : | Informatika |
| Judul Skripsi/Tugas Akhir | : | Sistem Pemesanan Makanan dan Minuman Di Osaka Ramen Depok Berbasis Java |

Telah diperiksa dan disetujui untuk diujikan

|  |  |
| --- | --- |
| Pembimbing Materi | Pembimbing Teknik |
| (Harry Dhika, M.Kom.) | (Meri Chrismes Aruan, S.Pd., M.Kom.) |

**LEMBAR PENGESAHAN**

|  |  |  |
| --- | --- | --- |
| Nama | : | Robby Awaldi |
| NPM | : | 201543501022 |
| Program Studi | : | Informatika |
| Fakultas | : | Teknik dan Ilmu Komputer |
| Judul | : | Sistem Pemesanan Makanan dan Minuman Di Osaka Ramen Depok Berbasis Java |

Panitia Ujian

Ketua : Prof. Dr. H. Sumaryoto \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Sekretaris : Ir. H. Soepardi Harris, M.T \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Anggota :

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | No. | Nama | Tanda Tangan | | 1. | Ni Wayan Parwati Septiani, M.M., M.Kom |  | | 2. | Achmad Sarwandianto, M.Kom |  | | 3. | Abdul Mufti, M.Kom |  | |

**LEMBAR PERNYATAAN**

Yang bertanda tangan di bawah ini:

|  |  |  |
| --- | --- | --- |
| Nama | : | Robby Awaldi |
| NPM | : | 201543501022 |
| Program Studi | : | Informatika |

Dengan ini menyatakan bahwa skripsi/tugas akhir dengan judul Sistem Pemesanan Makanan dan Minuman Di Osaka Ramen Depok Berbasis Java beserta seluruh isinya adalah benar-benar karya saya sendiri. Saya tidak melakukan penjiplakan atau pengutipan dengan cara-cara yang tidak sesuai dengan etika ilmu yang berlaku dalam masyarakat keilmuan. Atas pernyataan ini, saya siap menanggung risiko/sanksi apabila di kemudian hari ditemukan adanya pelanggaran etika keilmuan atau ada klaim dari pihak lain terhadap keaslian karya saya ini sesuai dengan Undang-undang Republik Indonesia Nomor 20 Tahun 2003 tentang Sistem Pendidikan Nasional Bab VI Pasal 25 ayat 2 dan Bab XX Pasal 70. Demikian pernyataan ini saya buat untuk dimanfaatkan sesuai dengan keperluan.

Jakarta,

Yang menyatakan,

Robby Awaldi

**ABSTRAK**

1. Robby Awaldi, NPM : 201543501022
2. **Sistem Pemesanan Makanan dan Minuman Di Osaka Ramen Depok Berbasis Java**. Skripsi/Tugas Akhir : Jakarta : Fakultas Teknik dan Ilmu Komputer : Program Studi Informatika : Universitas Indraprasta Persatuan Guru Republik Indonesia, Juli, 2019
3. xv + 5 Bab + 91 halaman
4. Kata Kunci :Sistem, Pemesanan, Java
5. Tujuan penelitian adalah untuk memberikan solusi kepada tempat penelitian terkait untuk menerapkan media digital dalam proses pemesanan makanan dan minuman. Metode pengembangan sistem yang digunakan adalah metode *waterfall*. Sedangkan metode pengumpulan data yang digunakan yaitu observasi, studi literatur, dan wawancara. Hasil yang diperoleh dari penelitian ini adalah sistem dapat membantu restoran Osaka Ramen dalam melayani pemesanan dengan cepat, akurat, dan efisien terutama saat pengunjung sedang ramai sehingga pekerjaan menjadi lebih mudah.
6. Daftar Pustaka: 1. Buku 18 buah (Tahun 2010 – 2015)

2. 2 Jurnal

1. Pembimbing: (**Harry** **Dhika, M.Kom.**) Pembimbing Materi

(**Meri Chrismes Aruan, S.Pd., M.Kom.**) Pembimbing Teknik

**LEMBAR MOTO**

“Allah mencintai pekerjaan yang apabila bekerja ia menyelesaikannya dengan baik”

(HR. Thabrani)

“Skripsi ini

Penulis persembahkan

kepada Mama, Ayah, dan Ade”

**KATA PENGANTAR**

Penulis memanjatkan puji syukur ke hadirat Allah SWT yang telah melimpahkan rahmat dan karunia-Nya kepada penulis, sehingga akhirnya penulis dapat menyelesaikan skripsi/tugas akhir ini tepat pada waktunya.

Skripsi/tugas akhir yang berjudul “Sistem Pemesanan Makanan dan Minuman Di Osaka Ramen Depok Berbasis Java” ini ditulis untuk memenuhi salah satu syarat guna memperoleh gelar sarjana pada Universitas Indraprasta PGRI. Pada kesempatan yang baik ini, izinkanlah penulis menyampaikan rasa hormat dan ucapan terima kasih kepada semua pihak yang dengan tulus ikhlas telah memberikan bantuan dan dorongan kepada penulis dalam menyelesaikan skripsi/tugas akhir ini, terutama kepada:

1. Bapak Harry Dhika, M.Kom. selaku Dosen Pembimbing Materi Universitas Indraprasta PGRI.
2. Ibu Meri Chrismes Aruan, S.Pd., M.Kom. selaku Dosen Pembimbing Teknik Universitas Indraprasta PGRI.
3. Bapak Taufik Hidayat selaku pemilik Osaka Ramen Depok dan seluruh pegawai Osaka Ramen Depok yang telah memberikan kesempatan kepada penulis untuk melakukan penelitian serta membantu dalam proses pembuatan tugas akhir.
4. Bapak Prof. Dr. H. Sumaryoto selaku Rektor Universitas Indraprasta PGRI.
5. Ibu Mei Lestari, M.Kom. selaku Ketua Program Studi Informatika.
6. Seluruh Dosen dan Staff Informatika Universitas Indraprasta PGRI.
7. Kedua orang tua penulis yang senantiasa menyayangi, mendoakan, serta memberikan dukungan moral dan moril.
8. Seluruh kerabat dan kawan seperjuangan yang telah membantu dan memberikan dukungan yang luar biasa khususnya Nadia Rizky, Aditya Maulana Kahfi, Surianto, Syamsir Achmad Hidayat, Erlangga Ario Tejo, Duhan Ferdiansyah, Toharudin, Mochamad Rizki Apriyana.

Penulis menyadari bahwa skripsi/tugas akhir ini masih banyak kekurangan, baik bentuk, isi, maupun teknik penyajiannya. Oleh sebab itu, kritik dan saran yang bersifat membangun dari berbagai pihak akan penulis terima dengan tangan terbuka serta sangat diharapkan. Semoga kehadiran skripsi/tugas akhir ini memenuhi sasarannya.

Jakarta, Juli 2019

Penulis

**DAFTAR ISI**

[LEMBAR PERSETUJUAN i](#_Toc12805015)

[LEMBAR PENGESAHAN ii](#_Toc12805016)

[LEMBAR PERNYATAAN iii](#_Toc12805017)

[ABSTRAK iv](#_Toc12805018)

[LEMBAR MOTO v](#_Toc12805019)

[KATA PENGANTAR vi](#_Toc12805020)

[DAFTAR ISI viii](#_Toc12805021)

[DAFTAR TABEL x](#_Toc12805022)

[DAFTAR GAMBAR xi](#_Toc12805023)

[DAFTAR SIMBOL xiii](#_Toc12805024)

[DAFTAR LAMPIRAN xv](#_Toc12805025)

[BAB I PENDAHULUAN 1](#_Toc12805026)

[A. Latar Belakang Masalah 1](#_Toc12805027)

[B. Identifikasi Masalah 2](#_Toc12805028)

[C. Batasan Masalah 2](#_Toc12805029)

[D. Rumusan Masalah 3](#_Toc12805030)

[E. Tujuan Penelitian 3](#_Toc12805031)

[F. Kegunaan Penelitian 4](#_Toc12805032)

[G. Sistematika Penulisan 4](#_Toc12805033)

[BAB II LANDASAN TEORI, PENELITIAN YANG RELEVAN DAN KERANGKA BERPIKIR 7](#_Toc12805034)

[A. Landasan Teori 7](#_Toc12805035)

[B. Penelitian Yang Relevan 15](#_Toc12805036)

[C. Kerangka Berpikir 19](#_Toc12805037)

[BAB III METODE PENELITIAN 21](#_Toc12805038)

[A. Waktu dan Tempat Penelitian 21](#_Toc12805039)

[B. Desain Penelitian 22](#_Toc12805040)

[C. Metode Pengumpulan Data 23](#_Toc12805041)

[D. Langkah - Langkah Pengembangan Sistem 24](#_Toc12805042)

[BAB IV ANALISIS SISTEM BERJALAN DAN RANCANGAN SISTEM YANG DIUSULKAN 26](#_Toc12805043)

[A. Profil Perusahaan 26](#_Toc12805044)

[B. Struktur Organisasi Perusahaan 27](#_Toc12805045)

[C. Proses Bisnis Sistem Berjalan 28](#_Toc12805046)

[D. Aturan Bisnis Sistem Berjalan 29](#_Toc12805047)

[E. Dekomposisi Fungsi Sistem 30](#_Toc12805048)

[F. Analisis Masukan (*Input*), Proses dan Keluaran (*Output*) Sistem Berjalan 30](#_Toc12805049)

[G. Diagram Alir Data (DAD) Sistem Berjalan (Diagram Konteks, Nol, Rinci) 32](#_Toc12805050)

[H. Analisis Permasalahan 33](#_Toc12805051)

[I. Alternatif Penyelesaian Masalah 34](#_Toc12805052)

[J. Aturan Bisnis Sistem Diusulkan 35](#_Toc12805053)

[K. Dekomposisi Fungsi Sistem Diusulkan 36](#_Toc12805054)

[L. Rancangan Masukan, Proses, dan Keluaran 36](#_Toc12805055)

[M. Diagram Alir Data (DAD) Sistem yang Diusulkan (Diagram Konteks, Nol, Rinci) 41](#_Toc12805056)

[N. Kamus Data Sistem yang Diusulkan 44](#_Toc12805057)

[O. Spesifikasi Proses Sistem yang Diusulkan 48](#_Toc12805058)

[P. Bagan Terstruktur Sistem yang Diusulkan 54](#_Toc12805059)

[Q. Spesifikasi Modul yang Diusulkan 61](#_Toc12805060)

[R. Rancangan Basis Data Sistem yang Diusulkan 62](#_Toc12805061)

[S. Rancangan Layar, Rancangan *Form* Masukan Data, dan Rancangan Keluaran 67](#_Toc12805062)

[T. Tampilan dan Penjelasan Layar, Tampilan Format Masukan, dan Tampilan Keluaran 78](#_Toc12805063)

[BAB V SIMPULAN DAN SARAN 90](#_Toc12805064)

[A. Simpulan 90](#_Toc12805065)

[B. Saran 91](#_Toc12805066)

[DAFTAR PUSTAKA](#_Toc12805067)

**LAMPIRAN**

**DAFTAR TABEL**

[Tabel 2.1 Folder Penting XAMPP 12](#_Toc12629295)

[Tabel 2.2 Penelitian Yang Relevan 16](#_Toc12629296)

[Tabel 3.1 Jadwal Penelitian 21](#_Toc12629297)

[Tabel 4.1 Spesifikasi File Menu 65](#_Toc12629298)

[Tabel 4.2 Spesifikasi File Detail Ramen 66](#_Toc12629299)

[Tabel 4.3 Spesifikasi File Transaksi 66](#_Toc12629300)

[Tabel 4.4 Spesifikasi File Pesanan 67](#_Toc12629301)

[Tabel 4.5 Spesifikasi File Level 67](#_Toc12629302)

**DAFTAR GAMBAR**

[Gambar 2.1 IntelliJ IDEA Community 2019.1.1 11](#_Toc12804954)

[Gambar 2.2 Kerangka Berpikir 19](#_Toc12804955)

[Gambar 3.1 Model Waterfall 24](#_Toc12804956)

[Gambar 4.1 Struktur Organisasi 27](#_Toc12804957)

[Gambar 4.2 Dekomposisi Fungsi Sistem 30](#_Toc12804958)

[Gambar 4.3 Diagram Konteks Sistem Berjalan 32](#_Toc12804959)

[Gambar 4.4 Diagram Nol Sistem Berjalan 33](#_Toc12804960)

[Gambar 4.5 Dekomposisi Fungsi Sistem Diusulkan 36](#_Toc12804961)

[Gambar 4.6 Diagram Konteks Sistem yang Diusulkan 41](#_Toc12804962)

[Gambar 4.7 Diagram Nol Sistem yang Diusulkan 42](#_Toc12804963)

[Gambar 4.8 Diagram Rinci Level 1 Proses 1 Sistem yang Diusulkan 42](#_Toc12804964)

[Gambar 4.9 Diagram Rinci Level 1 Proses 2 43](#_Toc12804965)

[Gambar 4.10 Diagram Rinci Level 1 Proses 3 43](#_Toc12804966)

[Gambar 4.11 Diagram Rinci Level 1 Proses 4 44](#_Toc12804967)

[Gambar 4.12 Bagan Terstruktur Menampilkan Katalog 54](#_Toc12804968)

[Gambar 4.13 Bagan Terstruktur Mengirim Pesanan 54](#_Toc12804969)

[Gambar 4.14 Bagan Terstruktur Menerima Konfirmasi Pesanan 55](#_Toc12804970)

[Gambar 4.15 Bagan Terstruktur Request Pembayaran 55](#_Toc12804971)

[Gambar 4.16 Bagan Terstruktur Mencetak Bill 56](#_Toc12804972)

[Gambar 4.17 Bagan Terstruktur Melakukan Pembayaran 56](#_Toc12804973)

[Gambar 4.18 Bagan Terstruktur Menyimpan Transaksi 57](#_Toc12804974)

[Gambar 4.19 Bagan Terstruktur Mencetak Laporan Pemesanan 57](#_Toc12804975)

[Gambar 4.20 Bagan Terstruktur Mencetak Laporan Pemasukan 58](#_Toc12804976)

[Gambar 4.21 Bagan Terstruktur Mencetak Laporan Menu Favorit 58](#_Toc12804977)

[Gambar 4.22 Bagan Terstruktur Mencetak Laporan Kunjungan 59](#_Toc12804978)

[Gambar 4.23 Bagan Terstruktur Menampilkan Daftar Menu 59](#_Toc12804979)

[Gambar 4.24 Bagan Terstruktur Tambah Menu 60](#_Toc12804980)

[Gambar 4.25 Bagan Terstruktur Hapus Menu 60](#_Toc12804981)

[Gambar 4.26 Bagan Terstruktur Ubah Menu 61](#_Toc12804982)

[Gambar 4.27 Bentuk Tidak Normal 62](#_Toc12804983)

[Gambar 4.28 Normalisasi Pertama 63](#_Toc12804984)

[Gambar 4.29 Normalisasi Kedua 63](#_Toc12804985)

[Gambar 4.30 Normalisasi Ketiga 64](#_Toc12804986)

[Gambar 4.31 Diagram ERD 64](#_Toc12804987)

[Gambar 4.32 Rancangan Antarmuka 68](#_Toc12804988)

[Gambar 4.33 Rancangan Tampilan Navigasi 68](#_Toc12804989)

[Gambar 4.34 Rancangan Tampilan Menu Ramen 69](#_Toc12804990)

[Gambar 4.35 Rancangan Tampilan Menu Minuman, Cemilan, dan Lainnya 70](#_Toc12804991)

[Gambar 4.36 Rancangan Tampilan Daftar Pesanan 71](#_Toc12804992)

[Gambar 4.37 Rancangan Tampilan Setting 72](#_Toc12804993)

[Gambar 4.38 Rancangan Tampilan Sign in 73](#_Toc12804994)

[Gambar 4.39 Rancangan Tampilan Side Bar 74](#_Toc12804995)

[Gambar 4.40 Rancangan Tampilan Halaman Utama 75](#_Toc12804996)

[Gambar 4.41 Rancangan Tampilan Daftar Menu 76](#_Toc12804997)

[Gambar 4.42 Rancangan Tampilan Laporan 77](#_Toc12804998)

[Gambar 4.43 Tampilan Navigasi 78](#_Toc12804999)

[Gambar 4.44 Tampilan Daftar Ramen 78](#_Toc12805000)

[Gambar 4.45 Tampilan Daftar Minuman 79](#_Toc12805001)

[Gambar 4.46 Tampilan Daftar Cemilan 79](#_Toc12805002)

[Gambar 4.47 Tampilan Daftar Lainnya 80](#_Toc12805003)

[Gambar 4.48 Tampilan Daftar Pesanan 81](#_Toc12805004)

[Gambar 4.49 Tampilan Setting 82](#_Toc12805005)

[Gambar 4.50 Tampilan Sign in 83](#_Toc12805006)

[Gambar 4.51 Tampilan Side Bar 84](#_Toc12805007)

[Gambar 4.52 Tampilan Halaman Utama 84](#_Toc12805008)

[Gambar 4.53 Tampilan Daftar Menu 85](#_Toc12805009)

[Gambar 4.54 Tampilan Laporan 86](#_Toc12805010)

[Gambar 4.55 Tampilan Laporan Pemesanan 87](#_Toc12805011)

[Gambar 4.56 Tampilan Laporan Menu Favorit 87](#_Toc12805012)

[Gambar 4.57 Tampilan Laporan Pemasukan 88](#_Toc12805013)

[Gambar 4.58 Tampilan Laporan Kunjungan 89](#_Toc12805014)

**DAFTAR SIMBOL**

1. **Simbol Diagram Alir Data**

|  |  |  |
| --- | --- | --- |
| **No.** | **Simbol *Data Flow Diagram*** | **Keterangan** |
| 1. |  | **Proses**, adalah suatu langkah-langkah yang dilakukan komputer untuk mengelola data dari masukan sehingga mengeluarkan suatu keluaran data atau informasi. |
| 2. |  | ***Entitas***, adalah suatu keberadaan yang memiliki keunikan dan beda dari yang lain. Entitas mengirim data menuju proses dan menerima hasil dari proses. |
| 3. |  | **Simpanan Data**, merupakan basis data untuk menyimpan data hasil proses sistem sehingga data tersebut dapat digunakan kembali atau diambil kembali. |
| 4. |  | **Alur Data**, menunjukkan arah data yang dikirim dari suatu entitas ke proses atau sebaliknya dan dari proses ke basis data atau sebaliknya. |

1. **Simbol** **Kamus Data**

|  |  |  |
| --- | --- | --- |
| **No.** | **Simbol** | **Arti** |
| 1. | = | Disusun atau terdiri atas |
| 2. | + | Dan |
| 3. | [|] | baik ...atau... |
| 4. | {}n | n kali diulang/bernilai banyak |
| 5. | () | Data operasional |
| 6. | \*...\* | Batas komentar |

1. **Simbol Bagan Terstruktur**

|  |  |  |
| --- | --- | --- |
| **No** | **Simbol** | **Keterangan** |
| 1. |  | ***Module***  Menunjukkan suatu modul. |
| 2. |  | ***Connection***  Untuk menghubungkan suatu modul dengan modul yang lainnya. |
| 3. |  | ***Loop***  Menunjukkan suatu perulangan di dalam modul. |
| 4. |  | ***Decision***  Menunjukkan suatu penyeleksian kondisi di dalam modul. |
| 5. |  | ***Couple***  Menunjukkan suatu data atau elemen kontrol yang dikirim dari suatu modul ke modul lainnya. Panah dengan lingkaran kosong menunjukkan data dikirim dan panah dengan lingkaran diblok menunjukkan elemen kontrol yang dikirim. |

**DAFTAR LAMPIRAN**

Lampiran 1 Daftar Riwayat Hidup Penulis

Lampiran 2 Surat Keterangan Mitra

Lampiran 3 Kartu Asistensi 1

Lampiran 4 Kartu Asistensi 2

Lampiran 5 Listing Program

# BAB I PENDAHULUAN

1. **Latar Belakang Masalah**

Pada saat ini perkembangan teknologi informasi berkembang dengan pesat dan banyak dimanfaatkan untuk memenuhi aktivitas atau pekerjaan manusia. Banyak pekerjaan manusia yang sebelumnya menggunakan sistem manual dapat diubah menjadi sistem yang terkomputerisasi. Meskipun teknologi informasi sudah berkembang pesat, tetapi masih saja terdapat sistem manual yang masih dipertahankan. Salah satu sistem manual yang masih dipertahankan sampai saat ini adalah sistem pemesanan makanan menggunakan cara konvensional.

Sistem pemesanan bersifat konvensional mengandalkan pelayan untuk melayani pemesanan setiap pelanggan dengan mencatat pesanan menggunakan alat tulis dan kertas. Sistem pemesanan yang bersifat konvensional ini sering kali terdapat permasalahan yang ditimbulkan, salah satunya adalah saat tempat makan tersebut sedang ramai, pelayan akan sangat sibuk melayani pelanggan, dan pelanggan yang baru datang harus menunggu pelayan agar dapat melakukan pemesanan. Masalah ini terjadi di Osaka Ramen Depok yang masih menggunakan sistem pemesanan bersifat konvensional.

Osaka Ramen Depok adalah resto yang menjual bermacam-macam jenis ramen khas Jepang dan juga terdapat berbagai masakan Indonesia. Seperti yang dijelaskan di atas, Osaka Ramen Depok masih menggunakan sistem pemesanan yang bersifat konvensional. Oleh karena itu, perlu dibuat “**Sistem Pemesanan Makanan Dan Minuman Di Osaka Ramen Depok Berbasis Java**”. Sistem pemesanan yang dibuat diharapkan dapat membantu mempercepat proses pemesanan di Osaka Ramen Depok.

1. **Identifikasi Masalah**

Berdasarkan latar belakang yang dijelaskan di atas dapat dilihat permasalahan yang terjadi di Osaka Ramen Depok, maka penulis akan menyimpulkan beberapa identifikasi masalah yang ada, yaitu:

1. Pencatatan pesanan terkadang mengalami kesalahan karena menggunakan kertas.
2. Isi laporan penjualan sering mengalami kesalahan dikarenakan masih menghitung manual.
3. Pelanggan yang ingin memesan sering terabaikan di saat resto sedang ramai.
4. Proses pembayaran masih terbilang lama dan berisiko terdapat kesalahan karena dihitung secara manual menggunakan kalkulator.
5. **Batasan Masalah**

Berdasarkan latar belakang dan permasalahan yang diidentifikasi di atas, maka penulis melakukan pembatasan masalah, yaitu:

1. Sistem pemesanan yang dibuat hanya untuk mencatat pemesanan makanan dan minuman pelanggan yang ada di resto.
2. Laporan yang dibuat adalah laporan transaksi pemesanan setiap hari.
3. Fitur yang dapat digunakan pelanggan yaitu menampilkan daftar menu, melakukan pemesanan menu, dan melakukan pembayaran.
4. Pembayaran yang dapat dilakukan hanya melalui pembayaran tunai/*cash*.
5. **Rumusan Masalah**

Berdasarkan pembatasan masalah, maka perumusan masalah dalam penelitian ini yaitu:

1. Bagaimana caranya membuat sistem pemesanan makanan dan minuman yang terkomputerisasi?
2. Bagaimana cara mengelola data transaksi menjadi laporan yang bermanfaat dan akurat?
3. Bagaimana cara membuat *user interface* yang mudah digunakan dan tidak membingungkan pelanggan?
4. Seperti apakah proses pembayaran apabila menggunakan sistem yang terkomputerisasi?
5. **Tujuan Penelitian**

Tujuan dari penelitian tugas akhir ini adalah untuk menghasilkan sebagai berikut:

1. Membantu proses pencatatan makanan dan minuman yang selama ini dalam bentuk manual ke dalam komputerisasi.
2. Dapat membuat laporan yang lebih akurat untuk setiap pembukuan.
3. Dapat mempermudah pelanggan untuk memesan makanan dan minuman.
4. Dapat mempercepat proses pembayaran dan menghindari kesalahan saat menghitung total pembayaran.
5. **Kegunaan Penelitian**

Adapun kegunaan dari penelitian ini dibagi menjadi beberapa aspek, yaitu sebagai berikut:

1. **Aspek Sistem**
2. Untuk memudahkan proses pemesanan yang sebelumnya masih menggunakan manual menjadi lebih cepat dan efisien.
3. Sistem tersebut membuat proses pemesanan lebih mudah tanpa harus memanggil pelayan.
4. **Aspek Manajerial**
5. Lebih efisien kinerja karyawan karena tidak ada yang merangkap tugas kerja.
6. Lebih terkoordinasi dalam sistem pelayanan di Osaka Ramen Depok.
7. **Aspek Penelitian Lanjutan**
8. Membuat sistem yang terhubung dengan cabang-cabang Osaka Ramen lainnya.
9. Membuat sistem yang tidak hanya bekerja di jaringan lokal tetapi dapat diakses melalui *online*.
10. **Sistematika Penulisan**

Dalam usaha pemberian gambaran secara singkat, isi skripsi ini akan terbagi menjadi lima bab yang selanjutnya akan dijabarkan dalam beberapa sub-bab. Sistematika pembahasan adalah sebagai berikut:

|  |  |
| --- | --- |
| **BAB I** | **PENDAHULUAN**  Dalam bab ini penulis memberikan gambaran awal tentang latar belakang, identifikasi masalah, batasan masalah, rumusan masalah, tujuan penelitian, kegunaan penelitian, dan sistematika penulisan. |
| **BAB II** | **LANDASAN TEORI, PENELITIAN YANG RELEVAN DAN KERANGKA BERPIKIR**  Pada bab ini penulis menguraikan teori-teori atas konsep-konsep yang melandasi pembahasan dalam tugas akhir. Penelitian yang relevan berisi deskripsi singkat penelitian lain yang masih berkaitan dengan penelitian ini. Sedangkan kerangka berpikir menggambarkan bagan pola pikir penelitian dari awal sampai akhir kegiatan. |
| **BAB III** | **METODE PENELITIAN**  Dalam bab ini menjelaskan tentang waktu dan tempat penelitian, desain penelitian, serta metode pengumpulan data dan langkah-langkah pengembangan sistem. |
| **BAB IV** | **ANALISIS SISTEM BERJALAN DAN RANCANGAN SISTEM YANG DIUSULKAN**  Pada bab ini penulis membahas mengenai profil perusahaan, struktur organisasi perusahaan, analisis sistem berjalan, analisis permasalahan, rancangan sistem yang diusulkan dan rancangan layar beserta penjelasannya. |
| **BAB V** | **SIMPULAN DAN SARAN**  Pada bab ini berisi mengenai simpulan dan saran yang berkaitan dengan sistem pemesanan makanan dan minuman di Osaka Ramen Depok. |

# BAB II LANDASAN TEORI, PENELITIAN YANG RELEVAN DAN KERANGKA BERPIKIR

1. **Landasan Teori**

Sebagai landasan pembahasan pada bab berikutnya diperlukan beberapa teori yang mendukung di antaranya:

1. **Sistem**

“Sistem adalah kumpulan dari elemen yang saling berhubungan dan bekerja sama untuk mencapai suatu tujuan” (Sujarweni, 2015). “Sistem adalah kumpulan dari sub sistem atau komponen apa pun baik berupa fisik yang berhubungan satu sama lain dan bekerja sama secara sistematis untuk mencapai satu tujuan tertentu” (Susanto, 2013). “Sistem sebagai kumpulan atau himpunan dari unsur, komponen, atau variabel yang terorganisir, saling berinteraksi satu sama lain, dan terpadu” (Sutabri, 2012).

1. **Pengertian Pemesanan**

“Pemesanan adalah suatu aktivitas yang dilakukan oleh konsumen sebelum membeli” (Rahman, 2015). Pemesanan memiliki arti yang luas tergantung objek apa yang dituju. Dalam penelitian ini pengertian pemesanan adalah suatu proses yang dilakukan pelanggan restoran untuk memesan makanan dan minuman yang tersedia di daftar menu.

1. **Java**

“Bahasa pemrograman Java merupakan salah satu dari sekian banyak bahasa pemrograman yang dapat dijalankan di berbagai sistem operasi termasuk telepon genggam” (Nofriadi, 2015).

“Java merupakan bahasa pemrograman berorientasi objek yang dikembangkan oleh Sun Microsystem yang dimulai oleh James Gosling dan dirilis pada tahun 1995, saat ini Sun Microsystem telah di akuisisi oleh Oracle Corporation” (Enterprise, 2015).

“Bahasa Java memberi harapan menjadi perekat universal yang menghubungkan pemakai dengan informasi dari *web server*, basis data, penyedia informasi dan sumber-sumber lain” (Hariyanto, 2014).

Java dapat digunakan di berbagai *platform* dikarenakan program Java berjalan menggunakan *Java virtual machine* (*JVM*). JVM inilah yang membuat program Java hanya perlu ditulis sekali dan dapat digunakan di berbagai sistem operasi dan *platform*. Java termasuk bahasa pemrograman berorientasi objek dikarenakan dalam bahasa Java semua sintaksis atau kode harus berada di dalam suatu *class* yang nantinya akan diimplementasikan menjadi sebuah objek.

1. **FXML**

FXML adalah sebuah bahasa *markup* berbasis XML yang digunakan untuk membuat tampilan atau antarmuka pada aplikasi berbasis JavaFX. FXML dapat dihasilkan secara otomatis menggunakan aplikasi *scene builder* sehingga dapat mempermudah dalam merancang tampilan antarmuka. Penggunaan FXML juga dimaksud untuk memisahkan kode rancangan tampilan dengan kode logika bisnis yang bertujuan membuat kode program lebih rapih dan mudah untuk dilakukan pengembangan lebih lanjut.

1. **MySQL**

“MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL (bahasa inggris: *database management system*) atau DBMS yang *multithread*, *multi-user*, dengan sekitar 6 (enam) juta instalasi di seluruh dunia” (Faizal & Irnawati, 2015).

“MySQL adalah perangkat lunak basis data. MySQL merupakan tipe data relasional yang artinya MySQL menyimpan datanya dalam bentuk tabel-tabel yang saling berhubungan” (Winarno, Zaki, & SmitDev Community, 2014).

“MySQL adalah sistem yang berguna untuk melakukan proses pengaturan koleksi-koleksi struktur data (*database*) baik yang meliputi proses pembuatan atau proses pengelolaan *database*” (Ahmar, 2013).

1. **Jaringan Komputer**

“Jaringan komputer adalah suatu himpunan interkoneksi sejumlah komputer. Dalam bahasa populer dapat dijelaskan bahwa jaringan komputer adalah kumpulan beberapa komputer, dan perangkat lain seperti *router*, *switch*, dan sebagainya” (Sofana, 2013).

“Jaringan komputer merupakan sebuah sistem yang terdiri atas komputer dan perangkat jaringan lainnya yang bekerja sama untuk mencapai suatu tujuan yang sama” (Andi, 2015).

Berdasarkan penjelasan yang dikemukakan para ahli di atas dapat disimpulkan bahwa jaringan komputer adalah suatu sistem yang berupa sejumlah komputer dan perangkat jaringan lain yang saling terkoneksi satu sama lain, saling mengirim informasi dan berkomunikasi sehingga dapat mencapai suatu tujuan yang sama.

1. **JSON**

“JSON (JavaScript Object Notation) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (*generate*) oleh komputer” (Juansyah Andi, 2015). Pertukaran data dengan menggunakan format JSON sangat ideal karena format JSON berbasis teks yang independen tetapi menggunakan konvensi yang akrab bagi programmer. JSON dibangun di atas dua struktur, yaitu:

1. Koleksi pasangan nama atau nilai. Dalam berbagai bahasa pemrograman, ini direalisasikan sebagai objek.
2. Daftar dari nilai. Dalam kebanyakan bahasa pemrograman, ini direalisasikan sebagai *array*.
3. **IntelliJ IDEA**



Gambar 2.1  
IntelliJ IDEA Community 2019.1.1

Sumber : Dokumen Pribadi

IntelliJ IDEA adalah IDE (*Integrated Development Environment*) yang digunakan untuk mengembangkan program atau aplikasi menggunakan bahasa pemrograman Java. Intellij IDEA adalah perangkat lunak yang mirip seperti Netbeans, akan tetapi Intellij IDEA memiliki beberapa kelebihan seperti memberikan saran yang lebih lengkap saat proses penulisan kode berlangsung.

1. **XAMPP**

“XAMPP adalah perangkat lunak *open source* yang diunggah secara gratis dan bisa dijalankan di semua sistem operasi seperti Windows, Linux, Solaris, dan Mac” (Buana, 2014).

Di dalam folder utama XAMPP terdapat beberapa folder penting yang perlu diketahui yaitu sebagai berikut:

Tabel 2.1  
Folder Penting XAMPP

|  |  |
| --- | --- |
| **Folder** | **Keterangan** |
| *Apache* | Folder utama dari *Apache Webserver* |
| *Htdocs* | Folder utama untuk menyimpan data-data latihan web, baik *PHP* maupun *HTML* biasa |
| *Manual* | Berisi *sub folder* yang di dalamnya terdapat manual program dan *database*, termasuk manual *PHP* dan *MySQL* |
| *MySQL* | Folder utama untuk *database MySQL Server* |
| *PHP* | Folder utama untuk program *PHP* |

Sumber : (Nugroho, 2014)

1. **ERD**

“ERD adalah suatu model jaringan yang menggunakan susunan data yang disimpan dalam *system* secara abstrak. ERD mendokumentasikan data dengan mengidentifikasi jenis entitas dan hubungannya” (Ladjamudin, 2013). ERD terdiri dari 3(tiga) tingkatan yaitu:

1. Entitas (*entity*)

Entitas adalah suatu objek yang memiliki ciri khas yang membedakan dari objek-objek lainnya. Entitas dapat berupa apa saja dan tidak harus berbentuk fisik, dapat juga berupa sesuatu yang abstrak.

1. Identifikasi (*attribute*)

Identifikasi adalah penjelasan dari suatu entitas sehingga entitas memiliki suatu karakteristik tertentu. Suatu entitas diidentifikasikan menjadi sekumpulan atribut yang menjelaskan detail dari entitas tersebut.

1. Hubungan (*relation*)

Relasi adalah hubungan antara satu entitas dengan entitas lain. Relasi menjelaskan bagaimana satu entitas memiliki peran dengan entitas lain sehingga setiap entitas saling terhubung. Hubungan antar dua entitas dapat dikategorikan menjadi tiga macam yaitu:

1. Hubungan satu dengan satu (1:1), yaitu satu entitas hanya memiliki satu hubungan saja dengan entitas lain dan begitu sebaliknya.
2. Hubungan satu dengan banyak (1:M) atau (M:1), yaitu satu entitas hanya memiliki satu hubungan dengan entitas lain tetapi entitas lain memiliki banyak hubungan.
3. Hubungan banyak dengan banyak (M:M), yaitu satu entitas memiliki banyak hubungan dengan entitas lain, begitu juga sebaliknya.
4. **Normalisasi**

“Normalisasi adalah proses untuk mengevaluasi dan memperbaiki struktur tabel untuk meminimalkan kesamaan data, sehingga mengurangi kemungkinan anomali data” (Coronel, Morris, & Rob, 2013).

Untuk melakukan normalisasi ada beberapa tahapan yang harus dilakukan, yaitu:

1. Bentuk Normal Pertama (*1NF*/*First Normal Form*)

Bentuk normal pertama dimulai dengan menyajikan data dalam format *tabular*, di mana setiap sel memiliki nilai tunggal dan tidak ada grup berulang. Untuk menghilangkan grup berulang, menghilangkan nol dengan memastikan bahwa setiap atribut grup berulang berisi

nilai data yang sesuai

1. Bentuk Normal Kedua (*2NF/Second Normal Form*)

Konversi ke 2NF dilakukan hanya ketika 1NF memiliki kunci primer komposit. Jika 1NF memiliki atribut primer tunggal kunci, maka tabel secara otomatis dalam 2NF.

1. Bentuk Normal Ketiga (*3NF/* *Third Normal Form*)

Untuk setiap ketergantungan transitif, tulis determinannya sebagai kunci primer untuk tabel baru. Penentunya adalah atribut apa pun yang

nilainya menentukan nilai lain dalam satu baris.

1. Bentuk Normal Boyce-Codd (*BCNF/Boyce Codd Normal Form*)

Setiap penentu dalam tabel adalah kunci kandidat. BCNF tidak boleh berisi lebih dari satu kunci kandidat.

1. Bentuk Normal Keempat (*4NF/Fourth Normal Form*)

Tidak ada ketergantungan bernilai banyak selain kunci kandidat.

1. Bentuk Normal Kelima (*5NF/Fifth Normal Form*)

Tidak kehilangan dekomposisi ke dalam tabel yang lebih kecil.

1. **DFD**

“*Data Flow Diagram* (DFD) merupakan alat untuk membuat diagram yang serbaguna” (Yakub, 2012). “*Data Flow Diagram* atau dalam Bahasa Indonesia menjadi Diagram Alir Data (DAD) adalah representasi grafik yang menggambarkan aliran informasi dan transformasi informasi yang diaplikasikan sebagai data yang mengatur dari masukan (*input*) dan keluaran (*output*)” (Sukamto & Shalahuddin, 2014).

1. **Kamus Data *(Data Dictionary*)**

Kamus data berfungsi sebagai tempat informasi suatu data yang mengenai definisi, struktur dan pemakaian dari sistem elemen. Elemen adalah unit data yang terkecil yang terdapat pada suatu sistem informasi. Kamus data berperan menyajikan suatu data yang ada pada sistem informasi tersebut.

“Kamus data adalah kumpulan daftar elemen data yang mengalir pada sistem perangkat lunak sehingga masukkan (*input*) dan keluaran (*output*) dapat dipahami secara umum (memiliki standar cara penulisan)” (Sukamto & Shalahuddin, 2014).

1. **Penelitian Yang Relevan**

Hasil penelitian yang bisa dijadikan acuan atau pembanding dalam kajian penelitian masalah sebagai berikut :

Tabel 2.2  
Penelitian Yang Relevan

|  |  |  |  |
| --- | --- | --- | --- |
| **Identitas Penulis** | **Judul** | **Tujuan Penelitian** | **Hasil Penelitian** |
| Reni Rosmitalia 11540088 (2016) di Universitas Islam Negeri Raden Fatah Palembang | Sistem Pemesanan Makanan di Rumah Makan Palapa Indah Berbasis *Web Service* Menggunakan Mobile Android | Membuat sistem pemesanan makanan berbasis *web service* dengan *platform* android pada Rumah Makan Palapa Indah | Setelah melakukan *requirement planning,* *user design* dan berakhir dengan pembuatan program yang sesungguhnya, maka hasil yang dicapai oleh penulisan adalah sebuah sistem pemesanan makanan di rumah makan palapa indah berbasis *web service* menggunakan *mobile* android dengan menggunakan bahasa pemrograman Java dan XML. |
| Abdul Haris 10510321 (2014) di Universitas Komputer Indonesia Bandung | Sistem Informasi Pemesanan Makanan dan Minuman di R.M Pondok Kapau Berbasis Android | Mengetahui sistem informasi yang sedang berjalan di R.M Pondok Kapau, merancang sistem informasi pemesanan makanan dan minuman berbasis android di R.M Pondok Kapau agar dapat mendukung dan meningkatkan pelayanannya. Melakukan pengujian sistem informasi pemesanan makanan dan minuman berbasis android yang diusulkan di R.M Pondok Kapau. | Mempercepat dalam pelayanan pemesanan makanan dan minuman karena pemesanan telah menggunakan aplikasi *mobile* android, mengurangi kesalahan dalam pemesanan makanan dan minuman karena telah tertata dengan baik pada aplikasi ini. |
| Dian Galih Tegar di Universitas Dian Nuswantoro | Sistem Informasi Pemesanan makanan dan Minuman Pada Omahe Cafe *and* Resto Berbasis *Client Server* Dengan *Platform* Android | Merancang aplikasi pemesanan makanan dan minuman berbasis *client server* dengan *platform android* pada Omahe Cafe *and* Resto, Menghasilkan aplikasi yang dapat memberitahu pesanan pelanggan ke bagian dapur dan kasir. | Aplikasi dapat mengatasi pemesanan makanan dan minuman dengan *platform* android berbasis *client server*. Aplikasi *mobile* android ini bisa memberitahu pesanan pelanggan ke bagian dapur dan kasir, cepat, selain itu juga keamanannya bisa lebih terjamin. |
| Liliany Candra dan Ari Amir Alkodri pada jurnal SISFOKOM, volume 03, nomor 02, September 2014 di STMIK Atma Luhur Pangkalpinang | Aplikasi Pemesanan Makanan Pada Bangka Original Cafe Berbasis *Client* Server Dengan Platform Android | Menghasilkan aplikasi yang dapat memberitahu pesanan pelanggan kebagian dapur, bar, kasir dan pelayan tidak perlu mencatat pesanan sehingga membantu mempercepat sistem pemesanan makanan pada restoran. | Sebuah aplikasi pemesanan makanan pada Bangka Original Cafe yang terintegrasi di mana pemesanan makanan dari pelanggan akan dimasukkan di *mobile* android oleh pelayan, kemudian *view* di bagian dapur, bar untuk diproses dan dicetak di bagian kasir. |

Sumber : Dokumen Pribadi

1. **Kerangka Berpikir**

SISTEM PEMESANAN MAKANAN DAN MINUMAN DI OSAKA RAMEN DEPOK

Kondisi saat ini:

1. Sistem pemesanan masih menggunakan cara konvensional.
2. Proses pemesanan dapat terjadi kesalahan karena data pesanan masih ditulis secara manual.
3. Penghitungan total biaya pesanan masih dihitung secara manual. menggunakan kalkulator.
4. Data laporan transaksi pesanan kurang akurat.

Faktor pendukung:

1. Komputer.
2. Bahasa pemrograman Java.
3. Menggunakan IntelliJ IDEA Community Edition 2019.1.1.
4. Menggunakan *database* *MySQL* XAMPP.

Aturan yang dipakai:

1. Pemesanan hanya dapat dilakukan di tempat / Restoran Osaka Ramen.
2. Pemesanan dapat dilakukan untuk makan di tempat atau dibawa pulang.
3. Pembayaran hanya dapat dilakukan menggunakan uang tunai.

Kondisi yang diharapkan:

1. Mengubah sistem pemesanan yang sebelumnya masih menggunakan cara konvensional menjadi sistem yang terkomputerisasi.
2. Mengurangi kesalahan yang terjadi saat proses pemesanan.
3. Mempercepat proses pembayaran.
4. Membantu dalam proses pembuatan laporan yang akurat.

Gambar 2.2  
Kerangka Berpikir

Sumber : Dokumen Pribadi

1. **Penjelasan Dari Gambar Kerangka Berpikir**

Pada kondisi saat ini pemesanan di Osaka Ramen Depok terdapat beberapa permasalahan karena sistem pemesanan yang digunakan di Osaka Ramen Depok masih dengan cara konvensional. Cara ini terbilang kuno karena masih menggunakan kertas untuk mencatat pesanan pelanggan. Saat melakukan pembayaran, kasir harus menghitung terlebih dahulu menggunakan kalkulator untuk mengetahui total pembayaran. Setiap harinya karyawan atau kasir Osaka Ramen harus membuat laporan penjualan secara manual kemudian dicatat ke buku laporan. Pembuatan laporan seperti itu tidaklah akurat dan dapat mengalami kesalahan saat menghitung transaksi pembayaran.

Dengan bantuan beberapa faktor pendukung seperti komputer, bahasa pemrograman Java, IntelliJ IDEA, dan XAMPP, serta mengikuti aturan yang dipakai seperti pemesanan hanya dapat dilakukan di tempat/restoran, pemesanan dapat dilakukan untuk makan di tempat atau dibawa pulang, dan pembayaran hanya dapat dilakukan menggunakan uang tuna, maka dapat dibuat sebuah sistem pemesanan berbasis Java sehingga diharapkan dapat mengubah sistem pemesanan yang sebelumnya masih menggunakan cara konvensional menjadi sistem yang terkomputerisasi, mengurangi kesalahan yang terjadi saat proses pemesanan, mempercepat proses pembayaran, dan membantu dalam proses pembuatan laporan sehingga lebih akurat.

# BAB III METODE PENELITIAN

1. **Waktu dan Tempat Penelitian**
2. **Waktu Penelitian**

Waktu penelitian berlangsung dari bulan Februari 2019 sampai dengan bulan Juni 2019, dengan perincian jadwal sebagai berikut:

Tabel 3.1  
Jadwal Penelitian

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| No. | Kegiatan | Februari | | | | Maret | | | | April | | | | Mei | | | | Juni | | | |
| 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| 1. | Identifikasi |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2. | Analisis dan Pengumpulan Data |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3. | Perancangan Sistem |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4. | Desain |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5. | Pengkodean dan Pengujian |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 6. | Implementasi dan Evaluasi |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 7. | Penulisan Laporan |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Sumber : Dokumen Pribadi

1. **Tempat Penelitian**

Tempat penelitian penulis dilakukan di Osaka Ramen di Jl. Keadilan No. 23G, Rangkapan Jaya Baru, Pancoran Mas, Kota Depok, Jawa Barat.

1. **Desain Penelitian**

Desain penelitian yang dilakukan menggunakan metode deskriptif, yaitu metode untuk berusaha mengumpulkan, menyajikan, serta menganalisis data sehingga dapat memberikan gambaran yang jelas. Dalam penelitian ini terdapat beberapa jenis data dan sumber data yang digunakan.

1. **Jenis Data yang Dikumpulkan**
2. Data Kuantitatif

Data yang diperoleh dari mitra yang berhubungan dengan penelitian seperti tanggapan pegawai mengenai proses pemesanan yang digunakan selama ini di Osaka Ramen Depok.

1. Data Kualitatif

Data yang berupa penjelasan langsung dari pemilik Osaka Ramen Depok atau pegawai yang bersangkutan dengan permasalahan penelitian.

1. **Sumber Data yang Digunakan**
2. Data primer

Data yang didapatkan dari sumber penelitian, dengan cara wawancara langsung. Dalam penelitian ini penulis memperoleh data dari pemilik/*owner* dari Osaka Ramen Depok.

1. Data Sekunder

Data yang diperoleh dari laporan-laporan, dokumentasi, dan sumber-sumber lainnya yang berhubungan dengan permasalahan yang sedang dihadapi.

1. **Metode Pengumpulan Data**

Metode yang digunakan dalam pengumpulan data pada penelitian ini adalah observasi, studi literatur, dan wawancara.

1. **Observasi**

Observasi adalah pengamatan suatu objek untuk mendapatkan pemahaman yang mendalam dengan cara merasakan langsung atau melihat objek yang ingin diamati. Tujuan observasi yaitu untuk mengetahui secara langsung sistem atau metode pemesanan yang digunakan di Osaka Ramen Depok.

1. **Studi Literatur**

Pada tahap ini penulis mengumpulkan literatur dari buku-buku referensi dan jurnal yang berhubungan dengan permasalahan dalam penelitian ini.

1. **Wawancara**

Wawancara adalah suatu teknik pengumpulan data melalui tanya jawab antara narasumber dan pewawancara. Pada penelitian ini dilakukan wawancara di mana narasumber adalah pemilik atau *owner* dari Osaka Ramen Depok dan pewawancara adalah penulis sendiri.

1. **Langkah - Langkah Pengembangan Sistem**

Langkah-langkah pengembangan sistem yang digunakan penulis dalam menyelesaikan penelitian adalah dengan menggunakan model *waterfall*.

“Model *waterfall* menyediakan pendekatan dalam membangun perangkat lunak secara terurut dimulai dari analisa, desain, pengkodean, pengujian, dan tahap pendukung (*support*)” (Rosa & Shalahuddin, 2013).

Analisis

Desain

Pengkodean

Pengujian

Sistem/Rekayasa Informasi

Gambar 3.1  
Model Waterfall

Sumber : (Rosa & Shalahuddin, 2013)

1. **Tahapan Metode *Waterfall***
2. Analisis

Analisis dapat dilakukan dengan cara proses pengumpulan data seperti wawancara, studi literatur, atau observasi. Untuk membuat sistem yang dapat memenuhi kebutuhan pengguna, dibutuhkan informasi berupa kebutuhan-kebutuhan pengguna terhadap sistem. Maka dari itu sebaiknya perlu dipelajari apa saja yang dibutuhkan pengguna agar terciptanya sistem yang dapat bermanfaat.

1. Desain

Setelah kebutuhan sistem dianalisis, tahap selanjutnya adalah melakukan perancangan dari data hasil analisa menjadi desain sistem agar mempermudah dalam melakukan implementasi.

1. Pengkodean

Setelah desain sistem dilakukan, maka yang perlu dilakukan adalah mengubah desain sistem ke suatu bentuk bahasa yang dapat dimengerti oleh komputer. Pada tahap ini desain sistem akan diimplementasikan menggunakan bahasa pemrograman yang akan dikerjakan oleh programmer.

1. Pengujian

Pengujian dilakukan untuk memastikan apakah sistem yang dibuat sudah sesuai dengan yang diharapkan. Apabila masih terjadi kekurangan atau kesalahan, maka akan dilakukan perbaikan sampai program sesuai dengan harapan.

1. **Keunggulan dan Kelemahan Metode *Waterfall***
2. Keunggulan
3. Tahapan tidak membingungkan karena dilakukan secara berurut.
4. Mudah diterapkan dalam mengembangkan sistem yang tidak terlalu besar.
5. Kelemahan
6. Tidak cocok diterapkan untuk mengembangkan sistem yang rumit dan besar.

# BAB IV ANALISIS SISTEM BERJALAN DAN RANCANGAN SISTEM YANG DIUSULKAN

1. **Profil Perusahaan**
2. **Sejarah Osaka Ramen Depok**

Osaka Ramen adalah restoran ramen khas Jepang yang menyediakan beberapa variasi ramen dengan harga terjangkau. Osaka Ramen juga menyediakan berbagai makanan dan minuman lokal Indonesia sehingga menu yang dijual tidak hanya ramen saja. Osaka Ramen Depok yang beralamat di Jl. Keadilan No. 23G, Rangkapan Jaya Baru, Pancoran Mas, Depok mulai beroperasi tahun 2011.

1. **Visi dan Misi Osaka Ramen**
2. Visi

Mengenalkan masakan Jepang khususnya ramen kepada masyarakat kelas menengah ke bawah.

1. Misi

Memberikan cita rasa ramen yang berkualitas dengan harga terjangkau untuk masyarakat kelas menengah ke bawah.

1. **Struktur Organisasi Perusahaan**
2. **Gambar Struktur Organisasi**

Pemilik

*Chef*

Kasir

Pelayan

Gambar 4.1  
Struktur Organisasi

Sumber : Pemilik Osaka Ramen Depok

1. **Deskripsi Kerja**

Dari struktur organisasi yang ada di Osaka Ramen, akan diuraikan tugas dan tanggung jawabnya sebagai berikut:

1. Pemilik
2. Membuat perencanaan, strategi dan kebijakan uang menyangkut operasi Osaka Ramen.
3. Melakukan kontrol secara keseluruhan atas operasi Osaka Ramen.
4. Memegang kendali atas keputusan penting yang bersifat umum yang berkaitan dengan finansial.
5. *Chef*
6. Menyajikan makanan dan minuman sesuai pesanan pelanggan.
7. Mengawasi jalannya operasional dapur.
8. Kasir
9. Menjalankan proses penjualan dan pembayaran.
10. Melakukan pencatatan atas semua transaksi.
11. Melakukan pelaporan penjualan kepada pemilik Osaka Ramen.
12. Pelayan
13. Menyajikan makanan dan minuman kepada pelanggan dengan sopan, ramah, dan memberikan pelayanan terbaik demi kepuasan pelanggan.
14. Membersihkan dan mengatur semua meja, kursi, dan peralatan lainnya yang ada di restoran.
15. Memastikan bahwa semua minuman dan makanan yang disajikan sesuai dengan pesanan pelanggan.
16. **Proses Bisnis Sistem Berjalan**
17. **Proses Pemesanan**
18. Pelanggan datang ke Osaka Ramen dan duduk di tempat yang diinginkan.
19. Pelayan akan menghampiri pelanggan yang baru datang dan memberikan buku menu, kertas dan pulpen untuk menulis pesanan.
20. Pelanggan yang sudah selesai memilih menu dan menulisnya di kertas akan memanggil pelayan dan memberikan kertas pesanan.
21. Pelayan akan memberitahukan kepada bagian dapur/*chef* untuk menyajikan makanan dan minuman sesuai pesanan.
22. Setelah makanan dan minuman disajikan, pelayan akan membawakan makanan dan minuman ke meja pelanggan.
23. **Proses Pembayaran**
24. Setelah pelanggan selesai makan, pelanggan menghampiri kasir untuk melakukan pembayaran.
25. Kasir akan menanyakan nomor meja pelanggan tersebut.
26. Setelah kasir mengetahui nomor meja pelanggan, kasir akan mencari kertas pesanan dan menghitung total harga secara manual.
27. Kasir memberitahukan total harga pesanan, selanjutnya pelanggan membayar pesanan tersebut.
28. Kasir memberikan bukti pembayaran dan kembalian apabila uang pelanggan melebihi total harga pesanan.
29. **Aturan Bisnis Sistem Berjalan**

Dengan menganalisis proses-proses bisnis sistem berjalan maka dapat diketahui aturan bisnis sistem berjalan sebagai berikut:

1. **Pemesanan**

Pemesanan dilakukan secara manual menggunakan kertas dan pulpen untuk mencatat pesanan. Pelayan membiarkan pelanggan untuk mencatat pesanan sendiri.

1. **Pembayaran**

Pembayaran boleh dilakukan setelah pelanggan selesai makan atau saat makanan dan minuman tiba di meja.

1. **Dekomposisi Fungsi Sistem**

Sistem Pemesanan Makanan dan Minuman Di Osaka Ramen

Pemesanan

Pembayaran

Laporan

Buku Menu

Kertas Pesanan

Bukti Pembayaran

Laporan penjualan harian

Gambar 4.2  
Dekomposisi Fungsi Sistem

Sumber : Dokumen Pribadi

1. **Analisis Masukan (*Input*), Proses dan Keluaran (*Output*) Sistem Berjalan**
2. **Analisis Masukan (*Input*)**

|  |  |  |  |
| --- | --- | --- | --- |
| a. | Nama Masukan | : | Data pesanan |
|  | Fungsi | : | Untuk mengetahui pesanan pelanggan |
|  | Sumber | : | Pelanggan |
|  | Media | : | Kertas |
|  | Frekuensi | : | Setiap ada pemesanan |
|  | Keterangan | : | Berisi tentang menu yang dipilih dan jumlah pesanan |

1. **Analisis Proses**
2. Proses Pemesanan

Proses pemesanan terdiri dari proses penerimaan pesanan menggunakan kertas untuk mencatat pesanan sampai penyajian pesanan.

1. Proses Pembayaran

Proses pembayaran yaitu melakukan penghitungan harga dari kertas pesanan sampai mencatat data pesanan beserta harga ke bukti pembayaran.

1. Proses Pembuatan Laporan

Proses ini yaitu menghitung total transaksi setiap harinya dan menghitung secara manual pemasukan yang didapat kemudian dicatat ke buku laporan.

1. **Analisis Keluaran (*Output*)**

|  |  |  |  |
| --- | --- | --- | --- |
| a. | Nama Keluaran | : | Bukti pembayaran |
|  | Fungsi | : | Sebagai bukti transaksi pembayaran |
|  | Sumber | : | Kasir |
|  | Media | : | Kertas |
|  | Frekuensi | : | Setiap terjadi pembayaran |
|  | Keterangan | : | Berisi tentang daftar pesanan, harga tiap pesanan, dan total harga pesanan |
| b. | Nama Keluaran | : | Laporan harian |
|  | Fungsi | : | Untuk mengetahui total pemasukan setiap harinya |
|  | Sumber | : | Kasir |
|  | Media | : | Buku laporan |
|  | Frekuensi | : | Setiap hari |
|  | Keterangan | : | Berisi tentang data laporan berupa total pemasukan dalam sehari |

1. **Diagram Alir Data (DAD) Sistem Berjalan (Diagram Konteks, Nol, Rinci)**
2. **Diagram Konteks Sistem Berjalan**

Proses diagram konteks dalam sistem berjalan yang berlangsung pada Osaka Ramen adalah:



Gambar 4.3  
Diagram Konteks Sistem Berjalan

Sumber : Dokumen Pribadi

1. **Diagram Nol Sistem Berjalan**



Gambar 4.4  
Diagram Nol Sistem Berjalan

Sumber : Dokumen Pribadi

1. **Analisis Permasalahan**

Dari hasil penelitian yang penulis lakukan maka penulis dapat menganalisis permasalahan yang ada pada pemesanan di Osaka Ramen yaitu:

1. Proses pemesanan makanan dan minuman yang dilakukan masih menggunakan kertas dan pulpen dalam pencatatan pesanan sehingga terkadang terjadi kesalahan pesanan.
2. Perhitungan total pembayaran masih manual menggunakan kalkulator yang kemungkinan dapat terjadi kesalahan dan membutuhkan waktu saat proses menghitung total pembayaran setiap transaksi.
3. Proses pembuatan laporan masih manual dan harus dihitung semua transaksi setiap harinya.
4. **Alternatif Penyelesaian Masalah**

Alternatif penyelesaian masalah yang akan penulis buat adalah dengan membuat sistem pemesanan makanan dan minuman secara terkomputerisasi, yang bertujuan untuk mempermudah dalam proses pengolahan data pesanan dan data transaksi penjualan. Sistem ini akan dibuat dengan bahasa pemrograman Java yang dibagi menjadi dua jenis yaitu aplikasi *server* dan aplikasi *client* yang saling terhubung melalui jaringan komputer dan saling berinteraksi mengirimkan data melalui format *JSON*. Aplikasi *client* didesain untuk digunakan pelanggan yang berisi untuk menampilkan daftar menu makanan dan minuman serta mengelola pesanan pelanggan, sedangkan aplikasi *server* digunakan pelayan Osaka Ramen untuk mengelola pesanan masuk dan melakukan transaksi pembayaran.

Dengan menggunakan sistem ini diharapkan proses pemesanan dan pembayaran di Osaka Ramen menjadi lebih mudah, efektif, mengecilkan risiko kesalahan manusia, serta membuat daya tarik pengunjung atau pelanggan Osaka Ramen yang memberikan pengalaman dalam memesan makanan dan minuman di tempat makan dengan cara yang modern dan masa kini. Sistem ini baik dari sisi *aplikasi* server maupun *client* didesain dengan *interface* yang semudah mungkin dipahami sehingga pelanggan atau pelayan Osaka Ramen tidak kesulitan saat mengoperasikannya.

1. **Aturan Bisnis Sistem Diusulkan**
2. **Pemesanan**

Pelanggan memesan menggunakan komputer yang disediakan di setiap meja yang sudah terdapat aplikasi untuk memesan. Setelah memesan melalui aplikasi *client*, pelanggan hanya tinggal menunggu pesanan diantarkan ke meja tersebut. Jika pelanggan merasa kurang dengan pesanannya, pelanggan dapat memesan kembali menggunakan aplikasi yang sama. Tetapi jika pelanggan telah menekan tombol bayar di dalam aplikasi tersebut, pelanggan harus menyelesaikan pembayaran terlebih dahulu jika ingin kembali memesan.

1. **Pembayaran**

Untuk melakukan pembayaran pelanggan dapat menekan tombol bayar di aplikasi *client* dan menunggu pelayan untuk mengantarkan *bill* atau tagihan pembayaran. Setelah itu pelanggan dapat membayar tagihan kepada pelayan. Pelayan akan kembali ke komputer *server* untuk menyelesaikan pembayaran dan mencetak bukti pembayaran.

1. **Dekomposisi Fungsi Sistem Diusulkan**

Sistem Pemesanan Makanan dan Minuman Di Osaka Ramen

Pemesanan

Pembayaran

Laporan

Katalog Menu

Daftar Pesanan

*Bill*

Laporan Pemesanan

Bukti Pembayaran

Laporan Pemasukan

Laporan Kunjungan

Laporan Menu Favorit

Gambar 4.5  
Dekomposisi Fungsi Sistem Diusulkan

Sumber : Dokumen Pribadi

1. **Rancangan Masukan, Proses, dan Keluaran**
2. **Rancangan Masukan**

|  |  |  |  |
| --- | --- | --- | --- |
| a. | Nama Masukan | : | Data pesanan |
|  | Fungsi | : | Untuk mengetahui pesanan pelanggan |
|  | Sumber | : | Pelanggan |
|  | Frekuensi | : | Setiap ada pemesanan |
|  | Keterangan | : | Berisi tentang menu yang dipilih dan jumlah pesanan |
|  |  |  |  |
| b. | Nama Masukan | : | Data pembayaran |
|  | Fungsi | : | Untuk melakukan transaksi pembayaran |
|  | Sumber | : | Kasir |
|  | Frekuensi | : | Setiap ada transaksi pembayaran |
|  | Keterangan | : | Berisi tentang uang tunai yang dibayar pelanggan |
| c. | Nama Masukan | : | Data menu baru |
|  | Fungsi | : | Untuk mendata menu baru ke dalam sistem |
|  | Sumber | : | Pemilik/*Chef* |
|  | Frekuensi | : | Setiap ada menu baru |
|  | Keterangan | : | Berisi tentang data menu baru |

1. **Rancangan Proses**
2. Proses Pemesanan

Pelanggan akan memasukkan data pesanan dengan cara memilih menu di dalam katalog, memasukkan level apabila menu tersebut adalah ramen, dan memasukkan jumlah menu yang ingin dipesan, kemudian pelanggan akan masuk ke daftar pesanan untuk memastikan pesanan-pesanan yang dipilih pelanggan sudah benar. Setelah pelanggan merasa data sudah benar, selanjutnya pelanggan harus menekan tombol pesan di dalam daftar pesanan dan mengkonfirmasi jika data sudah benar.

Data tersebut akan dikirim ke komputer server melalui jaringan komputer lokal yang ada di Osaka Ramen. Di komputer *server*, pelayan atau *chef* akan mengkonfirmasi apakah pesanan dapat disajikan atau tidak. Apabila pesanan dapat disajikan, maka dikomputer *client* yang ada di meja pelanggan tersebut dapat melihat bahwa pesanannya sedang diproses.

1. Proses Pembayaran

Pelanggan dapat melalukan pembayaran apabila semua pesanan sudah dikonfirmasi saat melakukan pesanan. Pembayaran dapat dilakukan dengan cara masuk ke dalam daftar pesanan kemudian menekan tombol bayar. Apabila permintaan berhasil maka akan muncul sebuah *pop up* yang berisi pesan kepada pelanggan untuk menunggu pelayan mengantarkan tagihan atau *bill.* Data permintaan tersebut akan dikirim ke komputer *server* yang nantinya akan dikonfirmasi oleh kasir untuk mencetak *bill* pembayaran. Pembayaran dapat dilakukan di meja langsung saat pelayan mengantarkan tagihan. Pelayan akan kembali ke komputer *server* untuk memasukkan jumlah uang tunai pelanggan dan mencetak bukti pembayaran. Apabila transaksi sudah selesai, pelayan akan menyimpan data pesanan dengan menekan tombol simpan. Data tersebut akan disimpan ke *database* agar nantinya dapat diolah menjadi data laporan.

1. Proses Pembuatan Laporan

Data-data transaksi yang disimpan di *database* akan dibaca dan akan dikelola sistem untuk dibuat beberapa laporan. Terdapat empat laporan yang dapat dihasilkan yaitu laporan pemesanan, laporan pemasukan, laporan menu favorit, dan laporan kunjungan.

1. **Rancangan Keluaran**

|  |  |  |  |
| --- | --- | --- | --- |
| a. | Nama Keluaran | : | *Bill* |
|  | Fungsi | : | Untuk memberitahukan total biaya pesanan ke pelanggan |
|  | Sumber | : | Kasir |
|  | Frekuensi | : | Setiap pelanggan meminta *bill* |
|  | Keterangan | : | Berisi tentang data pesanan dan total pembayaran |
| b. | Nama Keluaran | : | Bukti pembayaran |
|  | Fungsi | : | Untuk memberikan bukti pelanggan telah melakukan pembayaran |
|  | Sumber | : | Kasir |
|  | Frekuensi | : | Setiap pelanggan melakukan pembayaran |
|  | Keterangan | : | Berisi tentang data pesanan, total pembayaran, uang tunai pelanggan, data kembalian pelanggan. |
| c. | Nama Keluaran | : | Laporan pemesanan |
|  | Fungsi | : | Untuk merekam semua data pemesanan yang terjadi dalam sehari |
|  | Sumber | : | Kasir |
|  | Frekuensi | : | Setiap hari |
|  | Keterangan | : | Berisi tentang data pemesanan seperti nama menu, jumlah, harga, dan total harga |
| d. | Nama Keluaran | : | Laporan pemasukan |
|  | Fungsi | : | Untuk mengetahui total pemasukan dalam satu hari |
|  | Sumber | : | Kasir |
|  | Frekuensi | : | Setiap bulan |
|  | Keterangan | : | Berisi tentang total pemasukan dalam satu hari |
| e. | Nama Keluaran | : | Laporan menu favorit |
|  | Fungsi | : | Untuk mengetahui menu yang sering diminati pengunjung dan melakukan evaluasi terhadap daftar menu |
|  | Sumber | : | Kasir |
|  | Frekuensi | : | Setiap bulan |
|  | Keterangan | : | Berisi tentang daftar menu dan jumlah pesanan setiap menu |
|  |  |  |  |
| f. | Nama Keluaran | : | Laporan kunjungan |
|  | Fungsi | : | Untuk mengetahui jumlah pengunjung yang datang dalam satu bulan |
|  | Sumber | : | Kasir |
|  | Frekuensi | : | Setiap bulan |
|  | Keterangan | : | Berisi tentang jumlah pengunjung setiap bulan |

1. **Diagram Alir Data (DAD) Sistem yang Diusulkan (Diagram Konteks, Nol, Rinci)**
2. **Diagram Konteks Sistem yang Diusulkan**



Gambar 4.6  
Diagram Konteks Sistem yang Diusulkan

Sumber : Dokumen Pribadi

1. **Diagram Nol Sistem yang Diusulkan**



Gambar 4.7  
Diagram Nol Sistem yang Diusulkan

Sumber : Dokumen Pribadi

1. **Diagram Rinci Level 1 Proses 1**



Gambar 4.8  
Diagram Rinci Level 1 Proses 1 Sistem yang Diusulkan

Sumber : Dokumen Pribadi

1. **Diagram Rinci Level 1 Proses 2**



Gambar 4.9  
Diagram Rinci Level 1 Proses 2

Sumber : Dokumen Pribadi

1. **Diagram Rinci Level 1 Proses 3**



Gambar 4.10  
Diagram Rinci Level 1 Proses 3

Sumber : Dokumen Pribadi

1. **Diagram Rinci Level 1 Proses 4**



Gambar 4.11  
Diagram Rinci Level 1 Proses 4

Sumber : Dokumen Pribadi

1. **Kamus Data Sistem yang Diusulkan**

Kamus data merupakan data-data atau informasi yang menjelaskan atau merincikan suatu aliran data pada diagram alir data (DAD) yang diusulkan sebagai berikut:

|  |  |  |  |
| --- | --- | --- | --- |
| 1. | Nama arus data | : | Katalog |
|  | Alias | : | Buku menu, Daftar menu |
|  | Bentuk data | : | Data komputer |
|  | Arus data | : | Menu – proses pemesanan – pelanggan,  Menu – proses mengelola menu - *chef* |
|  | Penjelasan | : | Berisi data menu makanan dan minuman |
|  | Periode | : | Setiap pelanggan melihat menu |
|  | Volume | : | Rata – rata 200 pelanggan setiap hari |
|  | Struktur data | : | nama\_menu + tipe + harga\_menu |
| 2. | Nama arus data | : | Pesanan |
|  | Alias | : | Pesanan |
|  | Bentuk data | : | Data komputer |
|  | Arus data | : | Pelanggan – proses pemesanan - *chef* |
|  | Penjelasan | : | Mencatat pesanan pelanggan |
|  | Periode | : | Setiap pelanggan memesan |
|  | Volume | : | Rata – rata 400 makanan dan minuman |
|  | Struktur data | : | nama\_menu + jumlah + level + harga\_level |
| 3. | Nama arus data | : | Transaksi |
|  | Alias | : | *Bill,* bukti pembayaran |
|  | Bentuk data | : | Data komputer, kertas |
|  | Arus data | : | Proses pembayaran – kasir,  Proses pembayaran  *–* pelanggan,  Proses pembayaran - transaksi |
|  | Penjelasan | : | Mencatat transaksi pembayaran pelanggan |
|  | Periode | : | Setiap pelanggan melakukan transaksi pembayaran |
|  | Volume | : | Rata- rata 200 pengunjung sehari |
|  | Struktur data | : | id\_transaksi + no\_meja + tanggal + daftar\_pesanan |
| 4. | Nama arus data | : | Laporan pemesanan |
|  | Alias | : | Laporan pemesanan |
|  | Bentuk data | : | Data komputer, kertas |
|  | Arus data | : | Transaksi - proses mengelola laporan - *owner* |
|  | Penjelasan | : | Rekaman semua transaksi setiap hari |
|  | Periode | : | Setiap hari |
|  | Volume | : | 1 kali sehari |
|  | Struktur data | : | pukul + no\_meja + nama\_menu + jumlah + harga + total\_harga |
| 5. | Nama arus data | : | Laporan pemasukan |
|  | Alias | : | Laporan pemasukan |
|  | Bentuk data | : | Data komputer, kertas |
|  | Arus data | : | Transaksi - proses mengelola laporan – *owner* |
|  | Penjelasan | : | Rekaman semua transaksi setiap bulan |
|  | Periode | : | Setiap bulan |
|  | Volume | : | 1 kali sebulan |
|  | Struktur data | : | tanggal + total pemasukan |
| 6. | Nama arus data | : | Laporan menu favorit |
|  | Alias | : | Laporan menu favorit |
|  | Bentuk data | : | Data komputer, kertas |
|  | Arus data | : | Transaksi - proses mengelola laporan – *owner* |
|  | Penjelasan | : | Jumlah pesanan berdasarkan menu makanan dan minuman |
|  | Periode | : | Setiap bulan |
|  | Volume | : | 1 kali sebulan |
|  | Struktur data | : | nama\_menu + tipe + harga + total\_dipesan |
| 7. | Nama arus data | : | Laporan kunjungan |
|  | Alias | : | Laporan kunjungan |
|  | Bentuk data | : | Data komputer, kertas |
|  | Arus data | : | Transaksi - proses mengelola laporan – *owner* |
|  | Penjelasan | : | Jumlah kunjungan setiap bulan |
|  | Periode | : | Setiap bulan |
|  | Volume | : | 1 kali sebulan |
|  | Struktur data | : | tanggal + total\_kunjungan |

1. **Spesifikasi Proses Sistem yang Diusulkan**

Spesifikasi proses menjelaskan spesifikasi dari setiap proses pada diagram rinci sistem pemesanan makanan dan minuman di Osaka Ramen yang diusulkan sebagai berikut:

|  |  |  |  |
| --- | --- | --- | --- |
| 1. | Proses | : | 1.1 |
|  | Nama proses | : | Menampilkan katalog |
|  | Masukan | : | Daftar menu |
|  | Keluaran | : | Katalog |
|  | Uraian | : | Daftar menu diambil dari basis data untuk ditampilkan kepada pelanggan berdasarkan kategori menu tersebut |
| 2. | Proses | : | 1.2 |
|  | Nama proses | : | Mengirim pesanan |
|  | Masukan | : | Pesanan |
|  | Keluaran | : | Pesanan pelanggan |
|  | Uraian | : | Pesanan pelanggan dari komputer *client* dikirim ke komputer *server* dan ditampilkan ke *chef* sehingga pesanan dapat langsung dibuat |
| 3. | Proses | : | 1.3 |
|  | Nama proses | : | Menerima konfirmasi pesanan |
|  | Masukan | : | Konfirmasi pesanan |
|  | Keluaran | : | Status pesanan |
|  | Uraian | : | Pesanan akan dikonfirmasi oleh *chef* atau pelayan apakah pesanan tersebut dapat disajikan atau tidak, jika iya status pesanan akan diubah menjadi “diproses”, jika tidak pesanan akan dihapus |
| 4. | Proses | : | 2.1 |
|  | Nama proses | : | Buat transaksi |
|  | Masukan | : | *Request* pembayaran |
|  | Keluaran | : | Transaksi pembayaran |
|  | Uraian | : | Pelanggan yang ingin membayar akan dibuat data transaksi pembayaran dan data tersebut akan ditampilkan kepada kasir untuk diproses |
| 5. | Proses | : | 2.2 |
|  | Nama proses | : | Mencetak *bill* |
|  | Masukan | : | Konfirmasi cetak *bill* |
|  | Keluaran | : | *Bill* |
|  | Uraian | : | Data transaksi pembayaran pelanggan akan dicetak dalam bentuk *bill* dan akan diserahkan kepada pelanggan sebagai tagihan pembayaran |
| 6. | Proses | : | 2.3 |
|  | Nama proses | : | Melakukan pembayaran |
|  | Masukan | : | Uang pembayaran |
|  | Keluaran | : | Bukti pembayaran |
|  | Uraian | : | Pelanggan yang mendapatkan *bill* akan melakukan pembayaran dengan memberikan uang pembayaran kepada kasir, uang pembayaran tersebut dimasukkan oleh kasir ke dalam sistem, kemudian sistem akan mencetak bukti pembayaran untuk diserahkan kepada pelanggan |
| 7. | Proses | : | 2.4 |
|  | Nama proses | : | Menyimpan transaksi |
|  | Masukan | : | Konfirmasi simpan pesanan |
|  | Keluaran | : | Transaksi disimpan di *database* |
|  | Uraian | : | Setelah kasir melakukan pembayaran, kasir akan menyimpan data transaksi tersebut ke *database* |
| 8. | Proses | : | 3.1 |
|  | Nama proses | : | Mencetak laporan pemesanan |
|  | Masukan | : | Daftar transaksi |
|  | Keluaran | : | Laporan pemesanan |
|  | Uraian | : | Data transaksi diambil dari *database* kemudian dipilah berdasarkan tanggal. Data tersebut akan dibuat menjadi *file* pdf agar dapat dicetak |
| 9. | Proses | : | 3.2 |
|  | Nama proses | : | Mencetak laporan pemasukan |
|  | Masukan | : | Daftar transaksi |
|  | Keluaran | : | Laporan pemasukan |
|  | Uraian | : | Data transaksi diambil dari *database* kemudian dipilah berdasarkan tanggal. Data tersebut akan dibuat menjadi *file* pdf agar dapat dicetak |
|  |  |  |  |
| 10. | Proses | : | 3.3 |
|  | Nama proses | : | Mencetak laporan menu favorit |
|  | Masukan | : | Daftar transaksi |
|  | Keluaran | : | Laporan menu favorit |
|  | Uraian | : | Data transaksi diambil dari *database* kemudian dipilah berdasarkan tanggal. Data tersebut akan dibuat menjadi *file* pdf agar dapat dicetak |
| 11. | Proses | : | 3.4 |
|  | Nama proses | : | Mencetak laporan kunjungan |
|  | Masukan | : | Daftar transaksi |
|  | Keluaran | : | Laporan menu favorit |
|  | Uraian | : | Data transaksi diambil dari *database* kemudian dipilah berdasarkan tanggal. Data tersebut akan dibuat menjadi *file* pdf agar dapat dicetak |
| 12. | Proses | : | 4.1 |
|  | Nama proses | : | Menampilkan daftar menu |
|  | Masukan | : | Daftar menu |
|  | Keluaran | : | Daftar menu |
|  | Uraian | : | Daftar menu diambil dari *database* kemudian ditampilkan kepada *chef* melalui layar |
| 13. | Proses | : | 4.2 |
|  | Nama proses | : | Tambah menu |
|  | Masukan | : | Menu baru |
|  | Keluaran | : | Menu disimpan di *database* |
|  | Uraian | : | Data menu baru dimasukkan oleh *chef* kemudian data tersebut akan disimpan ke *database* |
| 14. | Proses | : | 4.3 |
|  | Nama proses | : | Hapus menu |
|  | Masukan | : | Menu yang dihapus |
|  | Keluaran | : | Menu dihapus dari *database* |
|  | Uraian | : | Data menu yang tidak ingin lagi dipasarkan akan dipilih oleh *chef* kemudian data tersebut akan dihapus dari *database* |
| 15. | Proses | : | 4.4 |
|  | Nama proses | : | Ubah menu |
|  | Masukan | : | Menu yang diubah |
|  | Keluaran | : | Menu diubah di *database* |
|  | Uraian | : | Data menu yang tidak sesuai akan diubah oleh *chef* kemudian data tersebut akan diubah di *database* |

1. **Bagan Terstruktur Sistem yang Diusulkan**



Gambar 4.12  
Bagan Terstruktur Menampilkan Katalog

Sumber : Dokumen Pribadi



Gambar 4.13  
Bagan Terstruktur Mengirim Pesanan

Sumber : Dokumen Pribadi



Gambar 4.14  
Bagan Terstruktur Menerima Konfirmasi Pesanan

Sumber : Dokumen Pribadi



Gambar 4.15  
Bagan Terstruktur Request Pembayaran

Sumber : Dokumen Pribadi



Gambar 4.16  
Bagan Terstruktur Mencetak Bill

Sumber : Dokumen Pribadi



Gambar 4.17  
Bagan Terstruktur Melakukan Pembayaran

Sumber : Dokumen Pribadi



Gambar 4.18  
Bagan Terstruktur Menyimpan Transaksi

Sumber : Dokumen Pribadi



Gambar 4.19  
Bagan Terstruktur Mencetak Laporan Pemesanan

Sumber : Dokumen Pribadi



Gambar 4.20  
Bagan Terstruktur Mencetak Laporan Pemasukan

Sumber : Dokumen Pribadi



Gambar 4.21  
Bagan Terstruktur Mencetak Laporan Menu Favorit

Sumber : Dokumen Pribadi



Gambar 4.22  
Bagan Terstruktur Mencetak Laporan Kunjungan

Sumber : Dokumen Pribadi



Gambar 4.23  
Bagan Terstruktur Menampilkan Daftar Menu

Sumber : Dokumen Pribadi



Gambar 4.24  
Bagan Terstruktur Tambah Menu

Sumber : Dokumen Pribadi



Gambar 4.25  
Bagan Terstruktur Hapus Menu

Sumber : Dokumen Pribadi



Gambar 4.26  
Bagan Terstruktur Ubah Menu

Sumber : Dokumen Pribadi

1. **Spesifikasi Modul yang Diusulkan**
2. **Modul pemesanan**

Tampilkan daftar menu

Ambil data menu yang dipilih

Ambil masukan jumlah dan level menu yang dipilih

Buat data pesanan dari menu yang dipilih, jumlah, dan level

Mengubah data pesanan menjadi format *JSON*

Kirim data pesanan dalam format *JSON* tersebut ke *server*

Konfirmasi data pesanan

1. **Modul *request bill***

Kirim *request* ke *server*

Buat transaksi berdasarkan nomor meja

Ambil data pesanan berdasarkan nomor meja

Hitung total pembayaran

Membuat *bill* dari data transaksi, data pesanan, dan total pembayaran

Cetak *bill*

1. **Modul pembayaran**

Ambil masukan nominal uang pembayaran

Ambil data transaksi berdasarkan nomor meja

Ambil data total pembayaran

Hitung kembalian dari uang kembalian dikurang total pembayaran

Membuat bukti pembayaran dari data transaksi, data pesanan, total pembayaran, uang pembayaran, dan kembalian

Cetak bukti pembayaran

1. **Rancangan Basis Data Sistem yang Diusulkan**
2. **Normalisasi**
3. Bentuk tidak normal (*Unnormalized*)

nama\_menu

tipe

harga\_menu

foto

deskripsi

level

harga\_level

id\_transaksi

no\_meja

tanggal

id\_pesanan

jumlah

Gambar 4.27  
Bentuk Tidak Normal

Sumber : Dokumen Pribadi

1. Normalisasi pertama (*First Normal Form*)

menu

id\_transaksi \*

no\_meja

tanggal

nama\_menu

jumlah

level

harga\_level

transaksi

nama\_menu\*

tipe

harga\_menu

foto

deskripsi

Gambar 4.28  
Normalisasi Pertama

Sumber : Dokumen Pribadi

1. Normalisasi kedua (*Second Normal Form*)

id\_pesanan\*

id\_transaksi

nama\_menu

jumlah

level

harga\_level

pesanan

menu

id\_transaksi \*

no\_meja

tanggal

transaksi

nama\_menu\*

tipe

harga\_menu

detail\_ramen

nama\_menu\*

foto

deskripsi

Gambar 4.29  
Normalisasi Kedua

Sumber : Dokumen Pribadi

1. Normalisasi ketiga (*Third Normal Form*)

level\*

harga\_level

level

id\_pesanan\*

id\_transaksi

nama\_menu

jumlah

level

pesanan

menu

id\_transaksi \*

no\_meja

tanggal

transaksi

nama\_menu\*

tipe

harga\_menu

detail\_ramen

nama\_menu\*

foto

deskripsi

Gambar 4.30  
Normalisasi Ketiga

Sumber : Dokumen Pribadi

1. **ERD (*Entity Relationship Diagram*)**



Gambar 4.31  
Diagram ERD

Sumber : Dokumen Pribadi

1. **Spesifikasi *File***

|  |  |  |  |
| --- | --- | --- | --- |
| a. | Nama *file* | : | menu |
|  | Media | : | *Harddisk* |
|  | *Primary key* | : | nama\_menu |
|  | Panjang *record* | : | 41 |
|  | Jumlah *record* | : | 41 x 31 (menu) = 1271 |
|  | Struktur | : |  |

Tabel 4.1  
Spesifikasi File Menu

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Nama *Field* | *Type* | *Size* | Keterangan |
| 1. | nama\_menu | *varchar* | 25 | Nama menu |
| 2. | tipe | *varchar* | 10 | Tipe menu |
| 3. | harga\_menu | *Int* | 6 | Harga menu |

Sumber : Dokumen Pribadi

|  |  |  |  |
| --- | --- | --- | --- |
| b. | Nama *file* | : | detail\_ramen |
|  | Media | : | *Harddisk* |
|  | *Primary key* | : | nama\_menu |
|  | Panjang *record* | : | 16777390 |
|  | Jumlah *record* | : | 16777390 x 4 (menu) = 67109560 |
|  | Struktur | : |  |

Tabel 4.2  
Spesifikasi File Detail Ramen

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Nama *Field* | *Type* | *Size* | Keterangan |
| 1. | nama\_menu | *varchar* | 25 | Nama menu |
| 2. | foto | *mediumblob* | 16777215 | Foto menu |
| 3. | deskripsi | *varchar* | 150 | Deskripsi menu |

Sumber : Dokumen Pribadi

|  |  |  |  |
| --- | --- | --- | --- |
| c. | Nama *file* | : | transaksi |
|  | Media | : | *Harddisk* |
|  | *Primary key* | : | id\_transaksi |
|  | Panjang *record* | : | 30 |
|  | Jumlah *record* | : | 30 x 100 (pesanan) x 12 (bulan) = 36000 |
|  | Struktur | : |  |

Tabel 4.3  
Spesifikasi File Transaksi

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Nama *Field* | *Type* | *Size* | Keterangan |
| 1. | id\_transaksi | *varchar* | 20 | Id transaksi |
| 2. | no\_meja | *varchar* | 2 | Nomor meja |
| 3. | Tanggal | *datatime* | 8 | Tanggal transaksi |

Sumber : Dokumen Pribadi

|  |  |  |  |
| --- | --- | --- | --- |
| d. | Nama *file* | : | pesanan |
|  | Media | : | *Harddisk* |
|  | *Primary key* | : | id\_pesanan |
|  | Panjang *record* | : | 62 |
|  | Jumlah *record* | : | 62 x 5 (menu) x 100 (pesanan) x 12 (bulan) = 372000 |
|  | Struktur | : |  |

Tabel 4.4  
Spesifikasi File Pesanan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Nama *Field* | *Type* | *Size* | Keterangan |
| 1. | id\_transaksi | *varchar* | 20 | Id transaksi |
| 2. | id\_pesanan | *varchar* | 12 | Id Pesanan |
| 3. | nama\_menu | *varchar* | 25 | Nama menu |
| 4. | jumlah | *Int* | 3 | Jumlah pesanan |
| 5. | level | *varchar* | 2 | Level pesanan |

Sumber : Dokumen Pribadi

|  |  |  |  |
| --- | --- | --- | --- |
| e. | Nama *file* | : | Level |
|  | Media | : | *Harddisk* |
|  | *Primary key* | : | level |
|  | Panjang *record* | : | 8 |
|  | Jumlah *record* | : | 8 x 11 (level) = 88 |
|  | Struktur | : |  |

Tabel 4.5  
Spesifikasi File Level

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Nama *Field* | *Type* | *Size* | Keterangan |
| 1. | level | *varchar* | 2 | Level pesanan |
| 2. | harga\_level | *Int* | 6 | Harga level |

Sumber : Dokumen Pribadi

1. **Rancangan Layar, Rancangan *Form* Masukan Data, dan Rancangan Keluaran**

Rancangan antar muka atau *user* *interface* adalah rancangan tampilan grafis untuk dilihat pengguna dan dapat dimengerti dan digunakan pengguna sehingga dapat terjadi adanya komunikasi antara pengguna dengan komputer. Rancangan tersebut dapat diuraikan sebagai berikut:

Gambar 4.32  
Rancangan Antarmuka

Sumber : Dokumen Pribadi

1. **Rancangan Tampilan Navigasi**

Ramen

Minuman

Cemilan

Lainnya

Daftar Pesanan

Gambar 4.33  
Rancangan Tampilan Navigasi

Sumber : Dokumen Pribadi

Tampilan ini terdapat pada aplikasi *client* yang digunakan pelanggan untuk memesan menu makanan dan minuman Osaka Ramen. Pada tampilan navigasi terdapat logo Osaka Ramen, empat kategori menu dan daftar pesanan. Tombol kategori-kategori ini digunakan untuk mengarahkan ke daftar menu sesuai kategori yaitu ramen, minuman, cemilan, dan lainnya. Daftar menu akan ditampilkan di bawah navigasi dalam satu jendela (*window*) yang sama. Sedangkan tombol daftar pesanan digunakan untuk memunculkan sebuah *pop up* atau dialog yang berisi daftar pesanan.

1. **Rancangan Tampilan Menu Ramen**

Nama Ramen

Deskripsi

Harga

Pesan

+

-

1

Level

Gambar 4.34  
Rancangan Tampilan Menu Ramen

Sumber : Dokumen Pribadi

Tampilan menu ramen dapat ditampilkan jika tombol kategori ramen di navigasi ditekan. Pada tampilan menu ramen terdapat data mengenai ramen seperti gambar ramen, nama ramen, deskripsi mengenai ramen, dan harga. Selain informasi mengenai ramen, juga terdapat sebuah masukan seperti masukan tingkat level kepedasan ramen dan jumlah ramen yang ingin dipesan, dan terdapat tombol pesan untuk mengeksekusi pesanan tersebut. Pada rancangan di atas, digunakan untuk menampung satu data ramen. Apabila data ramen lebih dari satu, maka rancangan tersebut akan diulang sebanyak jumlah data pada daftar ramen.

1. **Rancangan Tampilan Menu Minuman, Cemilan, dan Lainnya**

Nama Menu

Harga

Pesan

+

-

1

Gambar 4.35  
Rancangan Tampilan Menu Minuman, Cemilan, dan Lainnya

Sumber : Dokumen Pribadi

Tampilan ini dapat ditampilkan jika tombol kategori menu selain ramen ditekan. Tampilan menu minuman, cemilan, dan lainnya digunakan untuk menampilkan daftar menu selain ramen yaitu minuman, cemilan, dan lainnya. Pada tampilan ini terdapat informasi mengenai menu yaitu nama menu, dan harga. Selain informasi mengenai menu juga terdapat masukan yaitu jumlah pesanan menu dan tombol pesanan untuk mengeksekusi pesanan. Pada rancangan di atas, digunakan untuk menampung satu data menu. Apabila data menu lebih dari satu, maka rancangan tersebut akan diulang sebanyak jumlah data pada daftar menu.

1. **Rancangan Tampilan Daftar Pesanan**

Daftar Pesanan

Nama

Jumlah

Harga

Total

Status

Total harga:

Rp.0,00

Pesan

Bayar

Kembali

Gambar 4.36  
Rancangan Tampilan Daftar Pesanan

Sumber : Dokumen Pribadi

Tampilan daftar pesanan akan tampil jika tombol daftar pesanan yang ada di navigasi ditekan. Pada tampilan daftar pesanan terdapat tabel daftar pesanan, total harga pembayaran dan tiga tombol di bawah yaitu tombol pesan, bayar, dan kembali.

1. **Rancangan Tampilan *Setting***

No Meja

Host

Setting

Simpan

Port

Keluar

Gambar 4.37  
Rancangan Tampilan Setting

Sumber : Dokumen Pribadi

Tampilan *setting* akan muncul saat logo pada aplikasi *client* ditekan tiga kali. Hal ini dimaksud agar pelanggan tidak mengetahui cara mengakses tampilan *setting*. Pada tampilan *setting* terdapat konfigurasi untuk mengubah nomor meja, alamat *host* *server*, dan *port* *server*.

1. **Rancangan Tampilan *Sign In***

Username

Password

Sign in

Sign in

Gambar 4.38  
Rancangan Tampilan Sign in

Sumber : Dokumen Pribadi

Tampilan *sign in* akan muncul saat program atau aplikasi *server* di jalankan. Pada tampilan *sign in* terdapat logo Osaka Ramen, masukan *username, password* dan tombol untuk melakukan *sign in*.

1. **Rancangan Tampilan *Side Bar***

Halaman Utama

Daftar Menu

Laporan

Keluar

Gambar 4.39  
Rancangan Tampilan Side Bar

Sumber : Dokumen Pribadi

Tampilan ini akan tampil jika pengguna berhasil melakukan *sign in*. Pada tampilan *side bar* terdapat logo, dan 3 menu yaitu halaman utama, daftar menu, dan laporan. Pada tampilan ini juga terdapat tombol untuk keluar dari aplikasi.

1. **Rancangan Tampilan Halaman Utama**

Pesanan Masuk

Pembayaran

No meja

Nama

Jumlah

Terima

Tolak

No meja

Total harga

Bill

Struk

Simpan

Gambar 4.40  
Rancangan Tampilan Halaman Utama

Sumber : Dokumen Pribadi

Tampilan halaman utama dapat diakses pada awal tampilan atau saat tombol halaman utama pada *side bar* ditekan. Pada tampilan ini terdapat dua tabel yaitu tabel pesanan masuk untuk memantau pesanan yang masuk dan mengkonfirmasi pesanan tersebut dan terdapat tabel pembayaran untuk memantau permintaan pelanggan untuk melakukan pembayaran.

1. **Rancangan Tampilan Daftar Menu**

Hapus

Daftar Menu

Level

Nama

Tipe

Harga

Level

Harga

Tambah Menu

Ubah Level

Nama

Tipe

Harga

Deskripsi

Tambah/Ubah

Harga

Level

Ubah

Gambar

Pilih gambar...

Gambar 4.41  
Rancangan Tampilan Daftar Menu

Sumber : Dokumen Pribadi

Tampilan daftar menu dapat ditampilkan saat tombol daftar menu pada *side bar* di tekan. Pada tampilan daftar menu terdapat tabel daftar menu dan *form* menu untuk mengelola data menu seperti menambah, menghapus, dan mengubah data menu.

1. **Rancangan Tampilan Laporan**

Laporan Pemesanan

Laporan Pemasukan

Diagram Garis

Menu Favorit

Laporan Kunjungan

Diagram Garis

Pukul

No Meja

Nama

Jumlah

Harga

Total Harga

Dari

Sampai

Semua

Cetak

Gambar 4.42  
Rancangan Tampilan Laporan

Sumber : Dokumen Pribadi

Tampilan laporan dapat ditampilkan saat tombol laporan pada *side bar* di tekan. Tampilan laporan terdapat empat bagian yaitu laporan pemesanan, laporan menu favorit, laporan pemasukan, dan laporan kunjungan.

1. **Tampilan dan Penjelasan Layar, Tampilan Format Masukan, dan Tampilan Keluaran**
2. **Tampilan Layar**
3. Tampilan Navigasi



Gambar 4.43  
Tampilan Navigasi

Sumber : Dokumen Pribadi

Pada gambar di atas adalah tampilan navigasi pada aplikasi c*lient.* Navigasi digunakan sebagai petunjuk arah untuk menuju ke sebuah tampilan seperti tampilan daftar ramen, daftar minuman, daftar cemilan, daftar lainnya, dan daftar pesanan.

1. Tampilan Daftar Ramen



Gambar 4.44  
Tampilan Daftar Ramen

Sumber : Dokumen Pribadi

Pada gambar di atas merupakan daftar menu ramen yang ada di Osaka Ramen. Untuk melakukan pesanan, pelanggan dapat memilih level kepedasan, menentukan jumlah pesanan yang ada di setiap bawah menu. Setelah menentukan level dan jumlah, pelanggan dapat menekan tombol tambah. Pesanan akan disimpan ke dalam daftar pesanan.

1. Tampilan Daftar Menu Minuman, Cemilan, dan Lainnya



Gambar 4.45  
Tampilan Daftar Minuman

Sumber : Dokumen Pribadi



Gambar 4.46  
Tampilan Daftar Cemilan

Sumber : Dokumen Pribadi



Gambar 4.47  
Tampilan Daftar Lainnya

Sumber : Dokumen Pribadi

Pada tiga gambar di atas merupakan tampilan daftar menu minuman, cemilan, dan lainnya. Untuk melakukan pesanan, pelanggan hanya tinggal menentukan jumlah pesanan dan menekan tombol tambah. Pesanan akan disimpan di dalam daftar pesanan.

1. Tampilan Daftar Pesanan

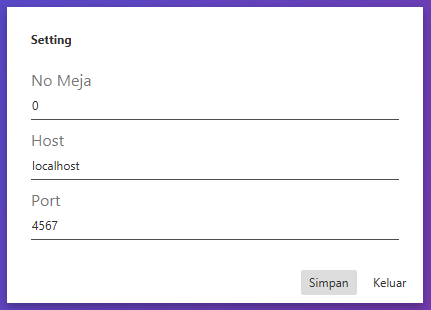


Gambar 4.48  
Tampilan Daftar Pesanan

Sumber : Dokumen Pribadi

Gambar di atas merupakan tampilan daftar pesanan. Daftar pesanan berfungsi untuk menampilkan daftar pesanan yang sudah ditambahkan. Untuk melakukan pemesanan, pelanggan dapat menekan tombol pesan, kemudian akan muncul dialog konfirmasi untuk memastikan bahwa data pesanan sudah benar. Apabila konfirmasi diterima, pesanan akan dikirim ke komputer *server* dan status pesanan akan berubah menjadi dipesan. Untuk melakukan pembayaran dapat menekan tombol bayar. Pembayaran dapat dilakukan setelah semua pesanan diproses di dapur. Untuk keluar dari daftar pesanan, pelanggan dapat menekan tombol keluar.

1. Tampilan *Setting*



Gambar 4.49  
Tampilan Setting

Sumber : Dokumen Pribadi

Pada gambar di atas merupakan tampilan dari *form setting* untuk melakukan konfigurasi seperti mengatur nomor meja, mengatur alamat *server*, dan *port* *server*. Pengaturan tersebut akan disimpan ke sebuah *file* berformat JSON.

1. Tampilan *Sign In*



Gambar 4.50  
Tampilan Sign in

Sumber : Dokumen Pribadi

Pada gambar di atas merupakan tampilan dari *form sign in* untuk pegawai Osaka Ramen dan pemilik Osaka Ramen masuk ke aplikasi *server*. Pada tampilan terdapat dua buah *text field* untuk memasukkan *username* dan *password*. *Username* dan *password* tersebut akan diperiksa di *database* apakah data yang dimasukkan sesuai dengan data yang ada di *database*.

1. Tampilan *Side Bar*

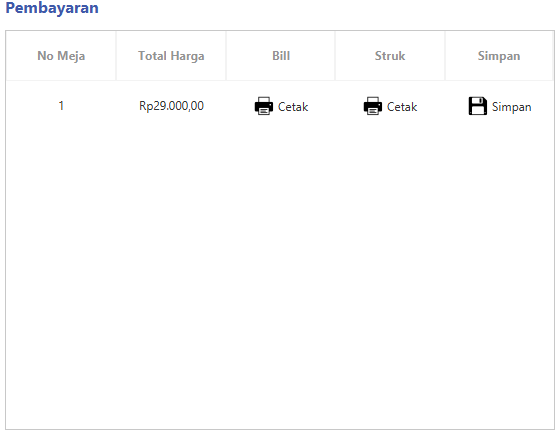


Gambar 4.51  
Tampilan Side Bar

Sumber : Dokumen Pribadi

Pada gambar di atas merupakan *side bar* dari aplikasi *server*. Menu pada *side bar* digunakan untuk mengarahkan pengguna untuk menuju ke tampilan yang diinginkan seperti halaman utama, daftar menu, laporan, dan tombol keluar untuk kembali ke *form sign in*.

1. Tampilan Halaman Utama



Gambar 4.52  
Tampilan Halaman Utama

Sumber : Dokumen Pribadi

Pada gambar di atas merupakan tampilan dari halaman utama. Pada halaman utama terdapat dua fitur yaitu *monitoring* pesanan masuk dan *monitoring* pembayaran. Apabila terdapat pesanan masuk maka akan muncul pada tabel pesanan masuk yang berisi data pesanan seperti nomor meja, nama pesanan, jumlah pesanan. Kemudian data tersebut dapat dikonfirmasi apakah pesanan dapat diterima karena stok di dapur tersedia atau dapat ditolak karena tidak ada stok atau alasan lain. Pada tabel pembayaran, data pembayaran atau transaksi akan muncul apabila ada pelanggan yang melakukan permintaan untuk melakukan pembayaran. Data pada tabel pembayaran berupa nomor meja, total pembayaran. Pelayan atau kasir dapat melakukan pencetakan *bill* untuk diantarkan ke meja pelanggan. Setelah pelanggan membayar, pelayan atau kasir dapat memasukkan nominal uang pembayaran dengan menekan terlebih dahulu tombol cetak struk. Setelah transaksi selesai, transaksi dapat disimpan ke *database* dengan menekan tombol simpan.

1. **Tampilan Masukan**
2. Tampilan Daftar Menu

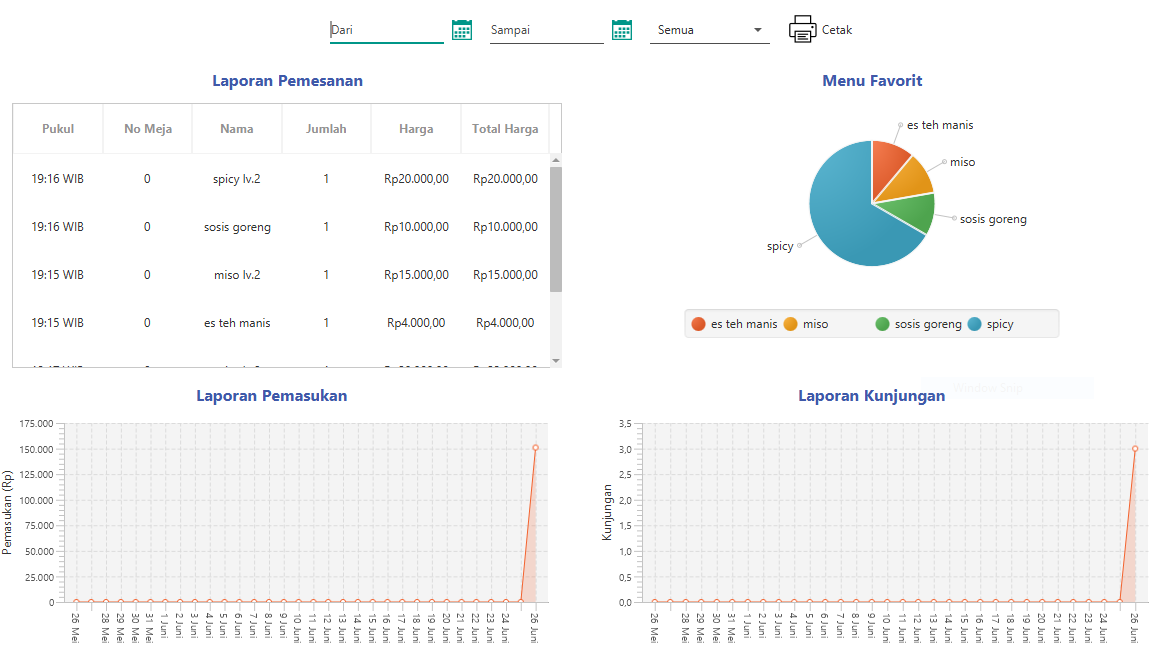


Gambar 4.53  
Tampilan Daftar Menu

Sumber : Dokumen Pribadi

Pada gambar di atas merupakan tampilan untuk mengelola daftar menu. Tampilan ini hanya dapat diakses oleh pemilik Osaka Ramen. Tampilan ini digunakan untuk mengelola data menu seperti menambah data menu, mengubah data menu, dan menghapus data menu. Selain mengelola daftar menu, juga terdapat fitur untuk mengelola harga level.

1. **Tampilan Keluaran**
2. Tampilan Laporan



Gambar 4.54  
Tampilan Laporan

Sumber : Dokumen Pribadi

Gambar di atas merupakan tampilan laporan-laporan yaitu laporan pemesanan, laporan menu favorit, laporan pemasukan, dan laporan kunjungan. Tampilan ini hanya dapat diakses oleh pemilik Osaka Ramen. Setiap laporan terdapat tombol untuk menampilkan laporan yang lebih rinci dalam bentuk pdf.

1. Tampilan Laporan Pemesanan



Gambar 4.55  
Tampilan Laporan Pemesanan

Sumber : Dokumen Pribadi

Pada gambar di atas merupakan tampilan laporan pemesanan di dalam *file* pdf. Laporan ini dirincikan menampilkan daftar transaksi pemesanan yang dicatat di dalam sistem.

1. Tampilan Laporan Menu Favorit



Gambar 4.56  
Tampilan Laporan Menu Favorit

Sumber : Dokumen Pribadi

Pada gambar di atas merupakan tampilan laporan menu favorit di dalam *file* pdf. Laporan ini dirincikan seperti menampilkan daftar menu dan total menu yang dipesan.

1. Tampilan Laporan Pemasukan



Gambar 4.57  
Tampilan Laporan Pemasukan

Sumber : Dokumen Pribadi

Pada gambar di atas merupakan tampilan laporan pemasukan di dalam *file* pdf. Laporan ini dirincikan seperti menampilkan daftar pemasukan setiap hari.

1. Tampilan Laporan Kunjungan



Gambar 4.58  
Tampilan Laporan Kunjungan

Sumber : Dokumen Pribadi

Pada gambar di atas merupakan tampilan laporan kunjungan di dalam *file* pdf. Laporan ini dirincikan seperti menampilkan daftar kunjungan setiap hari.

# BAB V SIMPULAN DAN SARAN

1. **Simpulan**

Berdasarkan permasalahan dan pembahasan yang dijelaskan di bab-bab sebelumnya, maka dapat disimpulkan bahwa:

1. Proses pencatatan secara komputerisasi dapat memberikan suatu daya tarik yang berbeda kepada pelanggan, mengurangi risiko kesalahan data pesanan, serta mengurangi pekerjaan pelayan karena proses pencatatan pesanan sudah ditangani oleh komputer.
2. Data-data pesanan yang terkumpul langsung diolah dalam bentuk laporan sehingga pelayan Osaka Ramen tidak perlu menghitung semua rangkap bukti pembayaran secara manual. Terdapat laporan menu favorit yang bermanfaat sebagai bahan untuk pengambil keputusan dalam menjalankan bisnis restoran.
3. Sistem pemesanan dapat diimplementasikan menggunakan jaringan *server client.* Data-data menu yang dipesan pelanggan melalui komputer *client* diubah dalam format *JSON* kemudian dikirim melalui *HTTP method* untuk dikirim ke komputer *server*. Kemudian di komputer *server* data-data tersebut diubah kembali ke bentuk *object* dan ditampilkan ke tabel pesanan masuk.
4. **Saran**

Berdasarkan hasil dari penelitian yang telah dilakukan, saran yang dapat penulis sampaikan untuk pengembangan sistem pemesanan Osaka Ramen berikutnya antara lain:

1. Sistem dapat dibuat lebih kompleks lagi dengan menambahkan fitur penghitungan profit yang didapat dalam setiap transaksi.
2. Aplikasi *client* yang digunakan dapat diubah menjadi aplikasi berbasis *platform mobile* sehingga lebih praktis dan lebih mudah digunakan pelanggan saat mengoperasikannya.
3. Pemesanan tidak hanya dapat dilakukan secara *offline* di Osaka Ramen, tetapi dapat dilakukan secara *online* dan terdapat layanan *delivery* ke tempat tujuan.

**DAFTAR PUSTAKA**

Ahmar, A. S. (2013). *Modifikasi Template CMS Lokomedia Cara Cepat dan Mudah Membuat Website Elegan Secara Gratis*. Yogyakarta: Garudhawaca.

Andi. (2015). *Membangun Sendiri Sistem Jaringan Komputer*. Yogyakarta: MADCOMS.

Buana, I. K. S. (2014). *Jago Pemrograman PHP*. Jakarta: Dunia Komputer.

Coronel, C., Morris, S., & Rob, P. (2013). *Database Systems: Design, Implementation and Management*. *Management*.

Enterprise, J. (2015). *Pengenalan Pemrograman Komputer*. Jakarta: PT Elex Media Komputindo.

Faizal, E., & Irnawati. (2015). *Pemrograman Java Web (JSP, JSTL & SERVLET) tentang Pembuatan Sistem Informasi Klinik Diimplementasikan dengan Netbeans IDE 7.2 dan MySQL*. Yogyakarta: Gava Media.

Hariyanto, B. (2014). *Esensi-Esensi Bahasa Pemrograman Java: Disertai Lebih Dari 100 Contoh Program*. Bandung: Informatika.

Juansyah Andi. (2015). PEMBANGUNAN APLIKASI CHILD TRACKER BERBASIS ASSISTED – GLOBAL POSITIONING SYSTEM (A-GPS) DENGAN PLATFORM ANDROID. *Jurnal Ilmiah Komputer Dan Informatika (KOMPUTA)*, *1*(1), 1–8. Retrieved from elib.unikom.ac.id/download.php?id=300375

Ladjamudin, A.-B. (2013). *Analisis dan Desain Sistem Informasi*. Yogyakarta: Graha Ilmu.

Nofriadi. (2015). *Java Fundamental Dengan Netbeans 8.0.2*. Yogyakarta: DeePublish.

Nugroho, B. (2014). *Dasar Pemrograman Web PHP-MySQL dengan Dreamweaver*. Yogyakarta: Gava Media.

Rahman, F. (2015). Aplikasi pemesanan undangan online. *Jurnal Sains Dan Informatika*, *1*(2), 78–87.

Rosa, & Shalahuddin. (2013). *Rekayasa Perangkat Lunak (Terstrukur dan Berorientasi Objek)*. Bandung: Informatika.

Sofana, I. (2013). *Membangun Jaringan Komputer: Mudah Membuat Jaringan Komputer (Wire & Wireless) Untuk Pengguna Windows dan Linux*. Bandung: Informatika.

Sujarweni, V. W. (2015). *Sistem Akuntansi*. Yogyakarta: Pustaka Baru Press.

Sukamto, & Shalahuddin. (2014). *Rekayasa Perangkat Lunak Terstruktur dan Berorientasi Objek*. Bandung: Informatika.

Susanto, A. (2013). *Sistem Informasi Akuntansi*. Bandung: Lingga jaya.

Sutabri, T. (2012). *Analisis Sistem Informasi*. Yogyakarta: Andi.

Winarno, E., Zaki, A., & SmitDev Community. (2014). *Pemrograman Web Berbasis HTML5, PHP, & JavaScript*. Jakarta: PT Elex Media Komputindo.

Yakub. (2012). *Pengantar Sistem Informasi*. Yogyakarta: Graha Ilmu.

**DAFTAR RIWAYAT HIDUP PENULIS**

****

1. **DATA PRIBADI**

|  |  |  |
| --- | --- | --- |
| Nama | : | Robby Awaldi |
| Tempat/Tanggal Lahir | : | Depok, 17 Juli 1997 |
| NPM | : | 201543501022 |
| Program Studi | : | Informatika |
| Jenis Kelamin | : | Laki-Laki |
| Agama | : | Islam |
| Kebangsaan | : | WNI |
| No. Handphone | : | 0896-5262-9124 |
| Alamat | : | Jl. Kembang Beji RT 05 RW 03 No.62B Kelurahan Beji Kecamatan Beji, Depok, Jawa Barat |

1. **RIWAYAT PENDIDIKAN**

|  |  |  |
| --- | --- | --- |
| 2003 - 2009 | : | SDN Beji 1 Depok |
| 2009 - 2012 | : | SMPN 5 Depok |
| 2012 - 2015 | : | SMK Prisma Depok |

Demikian riwayat hidup ini saya buat dengan sebenar – benarnya tanpa adanya pemalsuan data dan untuk dapat dipergunakan sebagaimana mestinya.

Jakarta, 02 Juli 2019

Robby Awaldi

**LISTING PROGRAM**

1. **Client.java**
2. package com.unindra.restoclient;
3. import com.google.gson.ExclusionStrategy;
4. import com.google.gson.FieldAttributes;
5. import com.google.gson.Gson;
6. import com.google.gson.GsonBuilder;
7. import com.jfoenix.controls.datamodels.treetable.RecursiveTreeObject;
8. import com.unindra.restoclient.models.StandardResponse;
9. import com.unindra.restoclient.models.StatusResponse;
10. import org.apache.commons.io.IOUtils;
11. import java.io.\*;
12. import java.net.HttpURLConnection;
13. import java.net.URL;
14. import java.nio.charset.StandardCharsets;
15. import static com.unindra.restoclient.models.Setting.setting;
16. public class Client {
17. public static StandardResponse get(String paramUrl) throws IOException {
18. URL url = new URL(setting().getBaseUrl() + paramUrl);
19. HttpURLConnection connection = (HttpURLConnection) url.openConnection();
20. connection.setRequestMethod("GET");
21. if (connection.getResponseCode() == HttpURLConnection.HTTP\_OK) {
22. BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(connection.getInputStream()));
23. String inputLine;
24. StringBuilder stringBuilder = new StringBuilder();
25. while ((inputLine = bufferedReader.readLine()) != null) {stringBuilder.append(inputLine);}
26. bufferedReader.close();
27. connection.disconnect();
28. return gson().fromJson(stringBuilder.toString(), StandardResponse.class);
29. } else return new StandardResponse(StatusResponse.ERROR);}
30. public static StandardResponse send(String paramUrl, String requestMethod, String json){
31. try {
32. URL url = new URL(setting().getBaseUrl() + paramUrl);
33. HttpURLConnection connection = (HttpURLConnection) url.openConnection();
34. connection.setConnectTimeout(5000);
35. connection.setRequestProperty("Content-Type", "application/json; charset=UTF-8");
36. connection.setRequestMethod(requestMethod);
37. connection.setDoInput(true);
38. connection.setDoOutput(true);
39. OutputStream outputStream = connection.getOutputStream();
40. connection.disconnect();
41. outputStream.write(json.getBytes(StandardCharsets.UTF\_8));
42. outputStream.close();
43. InputStream inputStream = new BufferedInputStream(connection.getInputStream());
44. String result = IOUtils.toString(inputStream, StandardCharsets.UTF\_8);
45. return gson().fromJson(result, StandardResponse.class);
46. } catch (IOException e) {
47. return new StandardResponse(StatusResponse.ERROR);}}
48. public static Gson gson() {
49. return new GsonBuilder().addSerializationExclusionStrategy(new ExclusionStrategy() {
50. @Override
51. public boolean shouldSkipField(FieldAttributes fieldAttributes) {return fieldAttributes.getDeclaringClass().equals(RecursiveTreeObject.class);}
52. @Override
53. public boolean shouldSkipClass(Class<?> aClass) {

return false;}

1. }).addDeserializationExclusionStrategy(new ExclusionStrategy() {
2. @Override
3. public boolean shouldSkipField(FieldAttributes fieldAttributes) {

return fieldAttributes.getDeclaringClass().equals(RecursiveTreeObject.class);}

1. @Override
2. public boolean shouldSkipClass(Class<?> aClass) {

return false;}

1. }).create();}}
2. **Main.java**
3. package com.unindra.restoclient;
4. import javafx.application.Application;
5. import javafx.fxml.FXMLLoader;
6. import javafx.scene.Parent;
7. import javafx.scene.Scene;
8. import javafx.scene.image.Image;
9. import javafx.stage.Stage;
10. public class Main extends Application {
11. @Override
12. public void start(Stage stage) throws Exception {
13. Parent root = FXMLLoader.load(getClass().getResource("/fxml/app.fxml"));
14. stage.setOnCloseRequest(event -> System.exit(0));
15. stage.getIcons().add(new Image("/icons/logo-ramen-bulet-merah-copy20x20.png"));
16. stage.setTitle("Osaka Ramen");
17. stage.setScene(new Scene(root));
18. stage.show();}
19. public static void main(String[] args) {
20. launch(args);}}
21. **AllMenuController.java**
22. package com.unindra.restoclient.controllers;
23. import com.jfoenix.controls.JFXButton;
24. import com.unindra.restoclient.Dialog;
25. import com.unindra.restoclient.Rupiah;
26. import com.unindra.restoclient.models.Pesanan;
27. import com.unindra.restoclient.models.Menu;
28. import com.unindra.restoclient.models.StandardResponse;
29. import com.unindra.restoclient.models.StatusResponse;
30. import javafx.scene.control.Label;
31. import javafx.stage.Stage;
32. import java.util.concurrent.atomic.AtomicInteger;
33. import static com.unindra.restoclient.models.Pesanan.getPesananList;
34. public class AllMenuController {
35. public Label namaLabel;
36. public Label hargaLabel;
37. public JFXButton tambahButton;
38. public Label jumlahLabel;
39. private AtomicInteger jumlah = new AtomicInteger(1);
40. void setMenu(Menu menu) {
41. namaLabel.setText(menu.getNama\_menu());
42. hargaLabel.setText(Rupiah.rupiah(menu.getHarga\_menu()));
43. tambahButton.setOnAction(event -> {
44. Dialog alert = new Dialog((Stage) tambahButton.getScene().getWindow());
45. if (getPesananList("dibayar").isEmpty()) {
46. Pesanan pesanan = new Pesanan(menu, jumlah.get());
47. StandardResponse standardResponse = pesanan.post();
48. if (standardResponse.getStatus() == StatusResponse.SUCCESS)
49. alert.information("Berhasil","Pesanan anda disimpan ke daftar pesanan");
50. else alert.information("Gagal","Pesanan anda gagal diproses");
51. } else alert.information("Gagal","Proses pembayaran belum selesai");
52. reset();});}
53. private void reset() {
54. jumlah.set(1);
55. jumlahLabel.setText(String.valueOf(jumlah.get()));
56. namaLabel.requestFocus();}
57. public void kurangJmlHandle() {
58. if (jumlah.decrementAndGet() > 0) {
59. jumlahLabel.setText(String.valueOf(jumlah.get()));
60. } else jumlah.incrementAndGet();}
61. public void tambahJmlHandle() {
62. jumlahLabel.setText(String.valueOf(jumlah.incrementAndGet()));}}
63. **AppController.java**
64. package com.unindra.restoclient.controllers;
65. import com.jfoenix.controls.JFXButton;
66. import com.unindra.restoclient.Dialog;
67. import com.unindra.restoclient.models.Pesanan;
68. import com.unindra.restoclient.models.Menu;
69. import com.unindra.restoclient.models.Setting;
70. import javafx.application.Platform;
71. import javafx.event.ActionEvent;
72. import javafx.fxml.FXMLLoader;
73. import javafx.fxml.Initializable;
74. import javafx.geometry.Insets;
75. import javafx.scene.Parent;
76. import javafx.scene.control.Label;
77. import javafx.scene.control.ScrollPane;
78. import javafx.scene.input.MouseEvent;
79. import javafx.scene.layout.FlowPane;
80. import javafx.scene.layout.VBox;
81. import javafx.stage.Stage;
82. import java.io.IOException;
83. import java.net.URL;
84. import java.util.List;
85. import java.util.ResourceBundle;
86. import static com.unindra.restoclient.Dialog.getDialogLayout;
87. import static com.unindra.restoclient.models.Pesanan.getPesananList;
88. import static com.unindra.restoclient.models.Setting.setting;
89. public class AppController implements Initializable {
90. public JFXButton ramenButton;
91. public JFXButton minumanButton;
92. public JFXButton cemilanButton;
93. public JFXButton lainnyaButton;
94. public JFXButton pesananButton;
95. public ScrollPane mainPane;
96. private FlowPane ramenPane;
97. private FlowPane minumanPane;
98. private FlowPane cemilanPane;
99. private FlowPane lainnyaPane;
100. private VBox pesananPane;
101. private Dialog pesananDialog;
102. @Override
103. public void initialize(URL location, ResourceBundle resources) {
104. ramenPane = new FlowPane();
105. minumanPane = new FlowPane();
106. cemilanPane = new FlowPane();
107. lainnyaPane = new FlowPane();
108. setRamenPane(Menu.menus("ramen"));
109. setAllMenuPane(Menu.menus("minuman"), minumanPane);
110. setAllMenuPane(Menu.menus("cemilan"), cemilanPane);
111. setAllMenuPane(Menu.menus("lainnya"), lainnyaPane);
112. mainPane.setContent(ramenPane);
113. try {
114. pesananPane = FXMLLoader.load(getClass().getResource("/fxml/pesanan.fxml"));
115. } catch (IOException e) {
116. e.printStackTrace();}
117. Thread thread = new Thread(() -> {
118. while (!Thread.interrupted()) {
119. try {
120. Pesanan.updatePesanan();
121. Thread.sleep(1000);
122. } catch (IOException | InterruptedException e) {
123. break;}}
124. ramenButton.setDisable(true);
125. minumanButton.setDisable(true);
126. cemilanButton.setDisable(true);
127. lainnyaButton.setDisable(true);
128. pesananButton.setDisable(true);
129. Platform.runLater(() ->
130. getDialog().information("Koneksi Terputus","Buka setting untuk mengubah alamat host atau port"));});
131. thread.start();
132. Platform.runLater(() -> pesananDialog = getDialog());
133. JFXButton pesanButton = new JFXButton("Pesan");
134. JFXButton bayarButton = new JFXButton("Bayar");
135. JFXButton keluarButton = new JFXButton("Keluar");
136. Platform.runLater(() -> pesananDialog.getDialog()
137. .setContent(getDialogLayout(new Label("Daftar Pesanan"),
138. pesananPane,pesanButton,bayarButton,keluarButton)));
139. pesanButton.setOnAction(event -> {
140. if (!getPesananList("belum dipesan").isEmpty()) {
141. Dialog dialog = getDialog();
142. dialog.confirmation("Pesanan tidak dapat dibatalkan setelah proses pemesanan berhasil",
143. e -> {
144. if (Pesanan.pesan()) {
145. dialog.information("Berhasil","Pesanan anda berhasil! mohon tunggu pesanan disajikan");}});}});
146. bayarButton.setOnAction(event -> {
147. if (getPesananList("diproses").size() == Pesanan.getPesananList().size())
148. if (Pesanan.getPesananList().size() != 0) try {
149. if (Pesanan.bayar())
150. getDialog().information("Mohon tunggu","Kasir akan mengantarkan bill ke meja anda");
151. } catch (IOException e) {
152. e.printStackTrace();}});
153. keluarButton.setOnAction(event -> pesananDialog.getDialog().hide());}
154. public void menuHandle(ActionEvent actionEvent) {
155. ramenButton.getStyleClass().set(2, "ramen");
156. minumanButton.getStyleClass().set(2, "minuman");
157. cemilanButton.getStyleClass().set(2, "cemilan");
158. lainnyaButton.getStyleClass().set(2, "lainnya");
159. Object source = actionEvent.getSource();
160. if (ramenButton.equals(source)) {
161. ramenButton.getStyleClass().set(2, "ramen-pressed");
162. mainPane.setContent(ramenPane);
163. } else if (minumanButton.equals(source)) {
164. minumanButton.getStyleClass().set(2, "minuman-pressed");
165. mainPane.setContent(minumanPane);
166. } else if (cemilanButton.equals(source)) {
167. cemilanButton.getStyleClass().set(2, "cemilan-pressed");
168. mainPane.setContent(cemilanPane);
169. } else if (lainnyaButton.equals(source)) {
170. lainnyaButton.getStyleClass().set(2, "lainnya-pressed");
171. mainPane.setContent(lainnyaPane);}}
172. public void daftarPesananHandle() {
173. Platform.runLater(() -> pesananDialog.getDialog().show());}
174. public void settingHandle(MouseEvent mouseEvent) {
175. if (mouseEvent.getClickCount() == 3) {
176. FXMLLoader fxmlLoader = new FXMLLoader();
177. fxmlLoader.setLocation(getClass().getResource("/fxml/setting.fxml"));
178. JFXButton keluarButton = new JFXButton("Keluar");
179. JFXButton simpanButton = new JFXButton("Simpan");
180. Dialog settingDialog = getDialog();
181. try {
182. settingDialog.getDialog().setContent(
183. getDialogLayout(new Label("Setting"),fxmlLoader.load(),simpanButton,keluarButton));
184. } catch (IOException e) {
185. e.printStackTrace();}
186. SettingController settingController = fxmlLoader.getController();
187. simpanButton.setOnAction(event -> {
188. Setting setting = setting();
189. setting.setNo\_meja(settingController.mejaField.getText());
190. setting.setHost(settingController.hostField.getText());
191. setting.setPort(settingController.portField.getText());
192. setting.simpan();
193. settingDialog.confirmation("Aplikasi akan dimatikan",
194. e -> System.exit(0));});
195. simpanButton.requestFocus();
196. keluarButton.setOnAction(event -> settingDialog.getDialog().hide());
197. settingDialog.getDialog().show();}}
198. private void setRamenPane(List<Menu> menuList) {
199. ramenPane.setPrefWidth(800);
200. ramenPane.setPrefHeight(500);
201. ramenPane.getStyleClass().add("body-pane");
202. menuList.forEach(menu -> {
203. try {
204. FXMLLoader fxmlLoader = new
205. FXMLLoader(getClass().getResource("/fxml/ramen.fxml"));
206. Parent root = fxmlLoader.load();
207. RamenController c = fxmlLoader.getController();
208. c.setMenu(menu);
209. ramenPane.getChildren().add(root);
210. } catch (IOException e) {
211. e.printStackTrace();}});}
212. private void setAllMenuPane(List<Menu> menuList, FlowPane allMenuPane) {
213. allMenuPane.setPrefWidth(800);
214. allMenuPane.setPrefHeight(500);
215. allMenuPane.setHgap(30);
216. allMenuPane.setVgap(30);
217. allMenuPane.setPadding(new Insets(30,20,30,36));
218. allMenuPane.getStyleClass().add("body-pane");
219. menuList.forEach(menu -> {
220. try {
221. FXMLLoader fxmlLoader = new FXMLLoader(getClass().getResource("/fxml/allmenu.fxml"));
222. Parent root = fxmlLoader.load();
223. AllMenuController c = fxmlLoader.getController();
224. c.setMenu(menu);
225. allMenuPane.getChildren().add(root);
226. } catch (IOException e) {
227. e.printStackTrace();}});}
228. private Dialog getDialog() {
229. return new Dialog((Stage) mainPane.getScene().getWindow());}}
230. **PesananController.java**
231. package com.unindra.restoclient.controllers;
232. import com.jfoenix.controls.JFXButton;
233. import com.jfoenix.controls.JFXTreeTableView;
234. import com.jfoenix.controls.RecursiveTreeItem;
235. import com.jfoenix.controls.datamodels.treetable.RecursiveTreeObject;
236. import com.unindra.restoclient.Dialog;
237. import com.unindra.restoclient.models.Pesanan;
238. import com.unindra.restoclient.models.StatusResponse;
239. import javafx.application.Platform;
240. import javafx.beans.property.SimpleStringProperty;
241. import javafx.collections.ListChangeListener;
242. import javafx.fxml.Initializable;
243. import javafx.scene.control.Label;
244. import javafx.scene.control.TreeTableCell;
245. import javafx.scene.control.TreeTableColumn;
246. import javafx.scene.control.TreeTableView;
247. import javafx.stage.Stage;
248. import javafx.util.Callback;
249. import java.net.URL;
250. import java.util.ResourceBundle;
251. import static com.unindra.restoclient.Rupiah.rupiah;
252. import static com.unindra.restoclient.models.Menu.menu;
253. public class PesananController implements Initializable {
254. public JFXTreeTableView<Pesanan> pesananTableView;
255. public Label totalLabel;
256. @Override
257. public void initialize(URL location, ResourceBundle resources) {
258. TreeTableColumn<Pesanan, String> namaCol = new TreeTableColumn<>("Nama");
259. TreeTableColumn<Pesanan, Integer> jumlahCol = new TreeTableColumn<>("Jumlah");
260. TreeTableColumn<Pesanan, String> hargaCol = new TreeTableColumn<>("Harga");
261. TreeTableColumn<Pesanan, String> totalCol = new TreeTableColumn<>("Total");
262. TreeTableColumn<Pesanan, String> hapusCol = new TreeTableColumn<>("Status");
263. namaCol.setCellValueFactory(param -> menu(param.getValue().getValue()).nama\_menuProperty());
264. jumlahCol.setCellValueFactory(param -> param.getValue().getValue().jumlahProperty());
265. hargaCol.setCellValueFactory(param -> menu(param.getValue().getValue()).harga\_menuProperty());
266. totalCol.setCellValueFactory(param -> param.getValue().getValue().totalProperty());
267. hapusCol.setCellValueFactory(param -> new SimpleStringProperty(""));
268. namaCol.setCellFactory(new Callback<TreeTableColumn<Pesanan, String>, TreeTableCell<Pesanan, String>>() {
269. @Override
270. public TreeTableCell<Pesanan, String> call(TreeTableColumn<Pesanan, String> param) {
271. return new TreeTableCell<Pesanan, String>() {
272. @Override
273. protected void updateItem(String item, boolean empty) {
274. super.updateItem(item, empty);
275. if (item == null) {
276. setText(null);
277. } else {
278. Pesanan i = Pesanan.getPesananList().get(getIndex());
279. if (menu(i).getTipe().equals("ramen"))
280. setText(item + " lv." + i.getLevel());
281. else setText(item);}}};}});
282. hapusCol.setCellFactory(new Callback<TreeTableColumn<Pesanan, String>, TreeTableCell<Pesanan, String>>() {
283. @Override
284. public TreeTableCell<Pesanan, String> call(TreeTableColumn<Pesanan, String> param) {
285. return new TreeTableCell<Pesanan, String>() {
286. final JFXButton button = new JFXButton("hapus");
287. @Override
288. protected void updateItem(String item, boolean empty) {
289. super.updateItem(item, empty);
290. if (item == null) {
291. setText(null);
292. setGraphic(null);
293. } else {
294. Pesanan thisPesanan = Pesanan.getPesananList().get(getIndex());
295. button.setFocusTraversable(false);
296. button.getStyleClass().add("hapus");
297. button.setOnAction(event -> {
298. Dialog alert = new Dialog((Stage) pesananTableView.getScene().getWindow());
299. alert.confirmation(
300. "Anda yakin ingin menghapus pesanan ini?",
301. e -> {
302. if (thisPesanan.delete().getStatus() == StatusResponse.SUCCESS)
303. alert.getDialog().hide();});});
304. if (!thisPesanan.getStatus\_item().equals("belum dipesan")) {
305. setText(thisPesanan.getStatus\_item());
306. setGraphic(null);
307. } else {
308. setText(null);
309. setGraphic(button);}}}};}});
310. Pesanan.getPesananList().addListener((ListChangeListener<Pesanan>) c ->
311. Platform.runLater(() -> totalLabel.setText(rupiah(Pesanan.getGrandTotal()))));
312. pesananTableView.setRoot(new RecursiveTreeItem<>(Pesanan.getPesananList(), RecursiveTreeObject::getChildren));
313. pesananTableView.getColumns().add(namaCol);
314. pesananTableView.getColumns().add(jumlahCol);
315. pesananTableView.getColumns().add(hargaCol);
316. pesananTableView.getColumns().add(totalCol);
317. pesananTableView.getColumns().add(hapusCol);
318. pesananTableView.setColumnResizePolicy(TreeTableView.CONSTRAINED\_RESIZE\_POLICY);}}
319. **RamenController.java**
320. package com.unindra.restoclient.controllers;
321. import com.jfoenix.controls.JFXButton;
322. import com.jfoenix.controls.JFXComboBox;
323. import com.unindra.restoclient.Dialog;
324. import com.unindra.restoclient.models.\*;
325. import javafx.collections.FXCollections;
326. import javafx.scene.control.Label;
327. import javafx.scene.control.ListCell;
328. import javafx.scene.control.ListView;
329. import javafx.scene.image.Image;
330. import javafx.scene.layout.VBox;
331. import javafx.scene.paint.ImagePattern;
332. import javafx.scene.shape.Circle;
333. import javafx.stage.Stage;
334. import javafx.util.Callback;
335. import java.io.ByteArrayInputStream;
336. import java.io.IOException;
337. import java.util.concurrent.atomic.AtomicInteger;
338. import static com.unindra.restoclient.Rupiah.rupiah;
339. import static com.unindra.restoclient.models.Pesanan.getPesananList;
340. public class RamenController {
341. public VBox rootPane;
342. public JFXComboBox<String> levelCombo;
343. public Label namaLabel;
344. public Label keteranganLabel;
345. public Label hargaLabel;
346. public Circle circle;
347. public JFXButton tambahButton;
348. public Label jumlahLabel;
349. private AtomicInteger jumlah = new AtomicInteger(1);
350. void setMenu(Menu menu) {
351. namaLabel.setText(menu.getNama\_menu().toUpperCase());
352. hargaLabel.setText(rupiah(menu.getHarga\_menu()));
353. try {
354. DetailRamen detailRamen = DetailRamen.detailRamen(menu.getNama\_menu());
355. if (detailRamen != null) {
356. keteranganLabel.setText(detailRamen.getDeskripsi());
357. Image image = new Image(new ByteArrayInputStream(detailRamen.getFoto()));
358. circle.setFill(new ImagePattern(image));}
359. } catch (IOException e) {
360. e.printStackTrace();}
361. levelCombo.setItems(
362. FXCollections.observableArrayList("0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "10"));
363. levelCombo.setCellFactory(new Callback<ListView<String>, ListCell<String>>() {
364. @Override
365. public ListCell<String> call(ListView<String> param) {
366. return new ListCell<String>() {
367. @Override
368. protected void updateItem(String item, boolean empty) {
369. super.updateItem(item, empty);
370. if (item == null || empty) {
371. setGraphic(null);
372. setText(null);
373. } else {
374. try {
375. if (Level.level(Integer.parseInt(item)).getHarga\_level() > 0) {
376. setText(item+" (+"+rupiah(Level.level(Integer.parseInt(item)).getHarga\_level())+")");
377. } else {
378. setText(item);}} catch (IOException e) {e.printStackTrace();}}}};}});
379. tambahButton.setOnAction(event -> {
380. Dialog alert = new Dialog((Stage) rootPane.getScene().getWindow());
381. if (getPesananList("dibayar").isEmpty()) {
382. if (!levelCombo.getSelectionModel().isEmpty()) {
383. Pesanan pesanan = new Pesanan(menu, jumlah.get(),
384. Integer.parseInt(levelCombo.getValue()));
385. StandardResponse standardResponse = pesanan.post();
386. if (standardResponse.getStatus() == StatusResponse.SUCCESS)
387. alert.information("Berhasil","Pesanan anda disimpan ke daftar pesanan");
388. else alert.information("Gagal","Pesanan anda gagal diproses");
389. } else alert.information("Gagal","Level belum dimasukkan");
390. } else alert.information("Gagal","Proses pembayaran belum selesai");
391. reset();});}
392. private void reset() {
393. jumlah.set(1);
394. jumlahLabel.setText(String.valueOf(jumlah.get()));
395. levelCombo.getSelectionModel().clearSelection();
396. rootPane.requestFocus();}
397. public void kurangJmlHandle() {
398. if (jumlah.decrementAndGet() > 0) {
399. jumlahLabel.setText(String.valueOf(jumlah.get()));
400. } else jumlah.incrementAndGet();}
401. public void tambahJmlHandle() {
402. jumlahLabel.setText(String.valueOf(jumlah.incrementAndGet()));}}
403. **DetailRamen.java**
404. package com.unindra.restoclient.models;
405. import java.io.IOException;
406. import java.util.Arrays;
407. import static com.unindra.restoclient.Client.get;
408. import static com.unindra.restoclient.Client.gson;
409. public class DetailRamen {
410. private String nama\_menu;
411. private byte[] foto;
412. private String deskripsi;
413. public DetailRamen(String nama\_menu, byte[] foto, String deskripsi) {
414. this.nama\_menu = nama\_menu;
415. this.foto = foto;
416. this.deskripsi = deskripsi;}
417. public static DetailRamen detailRamen(String nama\_menu) throws IOException {
418. StandardResponse standardResponse = get("/detail\_ramen/"+nama\_menu);
419. if (standardResponse.getStatus() == StatusResponse.SUCCESS) {
420. return gson().fromJson(standardResponse.getData(), DetailRamen.class);}
421. return null;}
422. public byte[] getFoto() {
423. return foto;}
424. public String getDeskripsi() {
425. return deskripsi;}
426. @Override
427. public String toString() {
428. return "DetailRamen{" +"nama\_menu='" + nama\_menu + '\'' +", foto=" + Arrays.toString(foto) +", deskripsi='" + deskripsi + '\'' +'}';}}
429. **Level.java**
430. package com.unindra.restoclient.models;
431. import javafx.collections.FXCollections;
432. import java.io.IOException;
433. import java.util.List;
434. import static com.unindra.restoclient.Client.get;
435. import static com.unindra.restoclient.Client.gson;
436. public class Level {
437. private int level;
438. private int harga\_level;
439. private static final String paramUrl = "/levels";
440. private Level(int level, int harga\_level) {
441. this.level = level;
442. this.harga\_level = harga\_level;}
443. private static List<Level> levelList() throws IOException {
444. StandardResponse standardResponse = get(paramUrl);
445. if (standardResponse.getStatus() == StatusResponse.SUCCESS) {
446. Level[] levels = gson().fromJson(standardResponse.getData(), Level[].class);
447. return FXCollections.observableArrayList(levels);}
448. return FXCollections.observableArrayList();}
449. public static Level level(int level) throws IOException {
450. return levelList().stream().filter(l -> l.level == level).findFirst().orElse(null);}
451. public int getHarga\_level() {
452. return harga\_level;}
453. @Override
454. public String toString() {
455. return "Level{" +"level=" + level +", harga\_level=" + harga\_level +'}';}}
456. **Menu.java**
457. package com.unindra.restoclient.models;
458. import javafx.beans.property.SimpleStringProperty;
459. import javafx.beans.property.StringProperty;
460. import javafx.collections.FXCollections;
461. import java.io.IOException;
462. import java.util.List;
463. import java.util.stream.Collectors;
464. import static com.unindra.restoclient.Client.get;
465. import static com.unindra.restoclient.Client.gson;
466. import static com.unindra.restoclient.Rupiah.rupiah;
467. public class Menu {
468. private String nama\_menu;
469. private int harga\_menu;
470. private String tipe;
471. public Menu(String nama\_menu, int harga\_menu, String tipe) {
472. this.nama\_menu = nama\_menu;
473. this.harga\_menu = harga\_menu;
474. this.tipe = tipe;}
475. private static List<Menu> menus() {
476. try {
477. Menu[] daftarMenus = gson().fromJson(get("/menus").getData(), Menu[].class);
478. return FXCollections.observableArrayList(daftarMenus);
479. } catch (IOException e) {
480. return FXCollections.observableArrayList();}}
481. public static List<Menu> menus(String tipe) {
482. return menus().stream().filter(menu -> menu.tipe.equals(tipe)).collect(Collectors.toList());}
483. public static Menu menu(Pesanan pesanan) {
484. return menus()
485. .stream()
486. .filter(menu -> menu.nama\_menu.equals(pesanan.getNama\_menu()))
487. .findFirst()
488. .orElse(null);}
489. public String getNama\_menu() {
490. return nama\_menu;}
491. public int getHarga\_menu() {
492. return harga\_menu;}
493. public String getTipe() {
494. return tipe;}
495. public StringProperty nama\_menuProperty() {
496. return new SimpleStringProperty(nama\_menu);}
497. public StringProperty harga\_menuProperty() {
498. return new SimpleStringProperty(rupiah(harga\_menu));}
499. @Override
500. public String toString() {
501. return "Menu{" +", nama\_menu='" + nama\_menu + '\'' +", harga\_menu=" + harga\_menu +'}';}}
502. **Pesanan.java**
503. package com.unindra.restoclient.models;
504. import com.google.gson.annotations.Expose;
505. import com.jfoenix.controls.datamodels.treetable.RecursiveTreeObject;
506. import javafx.beans.property.ObjectProperty;
507. import javafx.beans.property.SimpleObjectProperty;
508. import javafx.beans.property.SimpleStringProperty;
509. import javafx.beans.property.StringProperty;
510. import javafx.collections.FXCollections;
511. import javafx.collections.ObservableList;
512. import java.io.IOException;
513. import java.util.List;
514. import java.util.stream.Collectors;
515. import static com.unindra.restoclient.Client.\*;
516. import static com.unindra.restoclient.Rupiah.rupiah;
517. import static com.unindra.restoclient.models.Level.level;
518. import static com.unindra.restoclient.models.Menu.menu;
519. import static com.unindra.restoclient.models.Setting.setting;
520. public class Pesanan extends RecursiveTreeObject<Pesanan> {
521. private String id\_pesanan;
522. private String nama\_menu;
523. private int jumlah;
524. private int level;
525. private String no\_meja;
526. private String status\_item;
527. @Expose
528. private static final String paramUrl = "/pesanan";
529. @Expose
530. private static ObservableList<Pesanan> pesananList = FXCollections.observableArrayList();
531. private Pesanan(String nama\_menu, int jumlah, int lvl\_item, String no\_meja, String status\_item) {
532. this.id\_pesanan = "";
533. this.nama\_menu = nama\_menu;
534. this.jumlah = jumlah;
535. this.level = lvl\_item;
536. this.no\_meja = no\_meja;
537. this.status\_item = status\_item;}
538. public Pesanan(Menu menu, int jumlah) {
539. this(menu.getNama\_menu(), jumlah, 0, setting().getNo\_meja(), "belum dipesan");}
540. public Pesanan(Menu menu, int jumlah, int lvl\_item) {
541. this(menu.getNama\_menu(), jumlah, lvl\_item, setting().getNo\_meja(), "belum dipesan");}
542. public static void updatePesanan() throws IOException {
543. StandardResponse standardResponse = get(paramUrl + "/" + setting().getNo\_meja());
544. if (standardResponse.getStatus() == StatusResponse.SUCCESS) {
545. Pesanan[] pesananArrays = gson().fromJson(standardResponse.getData(), Pesanan[].class);
546. for (Pesanan pesanan : pesananArrays) pesanan.setChildren(FXCollections.observableArrayList());
547. Pesanan.pesananList.setAll(pesananArrays);}}
548. public static boolean pesan() {
549. getPesananList("belum dipesan").forEach(item -> item.status\_item = "dipesan");
550. return getPesananList("dipesan")
551. .stream()
552. .map(item -> item.put().getStatus() == StatusResponse.SUCCESS)
553. .reduce(true, (a, b) -> a && b);}
554. public static boolean bayar() throws IOException {
555. getPesananList("diproses").forEach(item -> item.status\_item = "dibayar");
556. boolean success = getPesananList("dibayar").stream()
557. .map(item -> item.put().getStatus() == StatusResponse.SUCCESS)
558. .reduce(true, (a, b) -> a && b);
559. StandardResponse standardResponse = get("/bayar/" + setting().getNo\_meja());
560. return success && standardResponse.getStatus() == StatusResponse.SUCCESS;}
561. public StandardResponse post() {
562. return send(paramUrl, "POST", gson().toJson(this));}
563. private StandardResponse put() {
564. return send(paramUrl, "PUT", gson().toJson(this));}
565. public StandardResponse delete() {
566. return send(paramUrl, "DELETE", gson().toJson(this));}
567. public static ObservableList<Pesanan> getPesananList() {
568. return pesananList;}
569. public static List<Pesanan> getPesananList(String status\_item) {
570. return pesananList
571. .stream()
572. .filter(pesanan -> pesanan.getStatus\_item().equals(status\_item))
573. .collect(Collectors.toList());}
574. private int getTotal() {
575. try {
576. return (menu(this)).getHarga\_menu() + level(level).getHarga\_level() \* jumlah;
577. } catch (IOException e) {
578. return 0;}}
579. public static int getGrandTotal() {
580. return getPesananList().stream().mapToInt(Pesanan::getTotal).sum();}
581. String getNama\_menu() {
582. return nama\_menu;}
583. public int getLevel() {
584. return level;}
585. public String getStatus\_item() {
586. return status\_item;}
587. public ObjectProperty<Integer> jumlahProperty() {
588. return new SimpleObjectProperty<>(jumlah);}
589. public StringProperty totalProperty() {
590. return new SimpleStringProperty(rupiah(getTotal()));}
591. @Override
592. public String toString() {
593. return "Pesanan{" +"id\_pesanan=" + id\_pesanan +", nama\_menu=" +nama\_menu +", jumlah=" + jumlah +", level=" + level +", no\_meja=" + no\_meja +", status\_item='" + status\_item + '\'' +'}';}}
594. **allmenu.fxml**
595. <?xml version="1.0" encoding="UTF-8"?>
596. <?import com.jfoenix.controls.JFXButton?>
597. <?import java.lang.String?>
598. <?import javafx.geometry.Insets?>
599. <?import javafx.scene.control.Label?>
600. <?import javafx.scene.image.ImageView?>
601. <?import javafx.scene.layout.ColumnConstraints?>
602. <?import javafx.scene.layout.GridPane?>
603. <?import javafx.scene.layout.HBox?>
604. <?import javafx.scene.layout.RowConstraints?>
605. <GridPane alignment="CENTER" hgap="10.0" prefHeight="80.0" styleClass="border" stylesheets="@../css/style.css" xmlns="http://javafx.com/javafx/8.0.172-ea" xmlns:fx="http://javafx.com/fxml/1" fx:controller="com.unindra.restoclient.controllers.AllMenuController">
606. <columnConstraints>
607. <ColumnConstraints hgrow="NEVER" maxWidth="173.0" minWidth="10.0" prefWidth="156.0" />
608. <ColumnConstraints hgrow="NEVER" maxWidth="298.0" minWidth="10.0" prefWidth="129.0" />
609. <ColumnConstraints hgrow="NEVER" maxWidth="298.0" minWidth="10.0" prefWidth="116.0" />
610. <ColumnConstraints hgrow="NEVER" maxWidth="177.0" minWidth="10.0" prefWidth="177.0" />
611. </columnConstraints>
612. <rowConstraints>
613. <RowConstraints minHeight="10.0" vgrow="SOMETIMES" />
614. </rowConstraints>
615. <padding>
616. <Insets left="10.0" />
617. </padding>
618. <Label fx:id="namaLabel" prefHeight="17.0" prefWidth="163.0" text="Nama Menu">
619. <styleClass>
620. <String fx:value="text-body" />
621. <String fx:value="text-white" />
622. <String fx:value="text-bold" />
623. </styleClass>
624. </Label>
625. <Label fx:id="hargaLabel" prefHeight="17.0" prefWidth="128.0" text="Harga" GridPane.columnIndex="1">
626. <styleClass>
627. <String fx:value="text-body" />
628. <String fx:value="text-white" />
629. <String fx:value="text-bold" />
630. </styleClass>
631. </Label>
632. <HBox alignment="CENTER" prefHeight="19.0" prefWidth="100.0" spacing="5.0" GridPane.columnIndex="2">
633. <JFXButton focusTraversable="false" mnemonicParsing="false" onAction="#kurangJmlHandle" styleClass="kurang" />
634. <Label fx:id="jumlahLabel" text="1">
635. <styleClass>
636. <String fx:value="text-body" />
637. <String fx:value="text-white" />
638. </styleClass>
639. </Label>
640. <JFXButton focusTraversable="false" layoutX="10.0" layoutY="10.0" mnemonicParsing="false" onAction="#tambahJmlHandle" styleClass="tambah" />
641. </HBox>
642. <JFXButton fx:id="tambahButton" focusTraversable="false" mnemonicParsing="false" prefHeight="25.0" prefWidth="160.0" styleClass="tambah-pesanan-button" text="Tambah" GridPane.columnIndex="3">
643. <graphic>
644. <ImageView fitHeight="30.0" fitWidth="30.0" pickOnBounds="true" preserveRatio="true" />
645. </graphic>
646. <padding>
647. <Insets bottom="-5.0" />
648. </padding>
649. </JFXButton>
650. </GridPane>
651. **app.fxml**
652. <?xml version="1.0" encoding="UTF-8"?>
653. <?import com.jfoenix.controls.JFXButton?>
654. <?import javafx.geometry.Insets?>
655. <?import javafx.scene.control.ScrollPane?>
656. <?import javafx.scene.image.Image?>
657. <?import javafx.scene.image.ImageView?>
658. <?import javafx.scene.layout.HBox?>
659. <?import javafx.scene.layout.Pane?>
660. <?import javafx.scene.layout.VBox?>
661. <?import javafx.scene.text.Font?>
662. <VBox prefHeight="470.0" prefWidth="690.0" stylesheets="@../css/style.css" xmlns="http://javafx.com/javafx/8.0.172-ea" xmlns:fx="http://javafx.com/fxml/1" fx:controller="com.unindra.restoclient.controllers.AppController">
663. <HBox alignment="BOTTOM\_LEFT" prefHeight="69.0" prefWidth="800.0" spacing="5.0" styleClass="body-white">
664. <ImageView fitHeight="50.0" fitWidth="50.0" onMousePressed="#settingHandle" pickOnBounds="true" preserveRatio="true">
665. <HBox.margin>
666. <Insets right="10.0" />
667. </HBox.margin>
668. <Image url="@../icons/logo-ramen-bulet-merah-copy50x50.png" />
669. </ImageView>
670. <JFXButton fx:id="ramenButton" focusTraversable="false" mnemonicParsing="false" onAction="#menuHandle" ripplerFill="WHITE" styleClass="ramen-pressed" text="Ramen" textOverrun="CLIP" HBox.hgrow="ALWAYS">
671. <font><Font name="System Bold" size="12.0" /></font>
672. <graphic>
673. <ImageView fitHeight="20.0" fitWidth="20.0" pickOnBounds="true" preserveRatio="true" />
674. </graphic>
675. </JFXButton>
676. <JFXButton fx:id="minumanButton" focusTraversable="false" layoutX="75.0" layoutY="45.0" mnemonicParsing="false" onAction="#menuHandle" ripplerFill="WHITE" styleClass="minuman" text="Minuman" textOverrun="CLIP" HBox.hgrow="ALWAYS">
677. <font>
678. <Font name="System Bold" size="12.0" />
679. </font>
680. <graphic>
681. <ImageView fitHeight="20.0" fitWidth="20.0" pickOnBounds="true" preserveRatio="true" />
682. </graphic>
683. </JFXButton>
684. <JFXButton fx:id="cemilanButton" focusTraversable="false" mnemonicParsing="false" onAction="#menuHandle" ripplerFill="WHITE" styleClass="cemilan" text="Cemilan" textOverrun="CLIP" HBox.hgrow="ALWAYS">
685. <font>
686. <Font name="System Bold" size="12.0" />
687. </font>
688. <graphic>
689. <ImageView fitHeight="20.0" fitWidth="20.0" pickOnBounds="true" preserveRatio="true" />
690. </graphic>
691. </JFXButton>
692. <JFXButton fx:id="lainnyaButton" focusTraversable="false" mnemonicParsing="false" onAction="#menuHandle" ripplerFill="WHITE" styleClass="lainnya" text="Lainnya" textOverrun="CLIP" HBox.hgrow="ALWAYS">
693. <font>
694. <Font name="System Bold" size="12.0" />
695. </font>
696. <graphic>
697. <ImageView fitHeight="20.0" fitWidth="20.0" pickOnBounds="true" preserveRatio="true" />
698. </graphic>
699. </JFXButton>
700. <Pane HBox.hgrow="SOMETIMES" />
701. <JFXButton fx:id="pesananButton" alignment="TOP\_CENTER" ellipsisString="" focusTraversable="false" mnemonicParsing="false" onAction="#daftarPesananHandle" prefHeight="28.0" prefWidth="128.0" styleClass="pesanan-button" text="Daftar Pesanan" textAlignment="CENTER" textOverrun="CLIP">
702. <graphic>
703. <ImageView fitHeight="20.0" fitWidth="20.0" pickOnBounds="true" preserveRatio="true" />
704. </graphic>
705. <font>
706. <Font name="System Bold" size="12.0" />
707. </font>
708. </JFXButton>
709. <padding>
710. <Insets bottom="10.0" left="50.0" right="50.0" top="10.0" />
711. </padding>
712. <VBox.margin>
713. <Insets />
714. </VBox.margin>
715. </HBox>
716. <ScrollPane fx:id="mainPane" fitToHeight="true" fitToWidth="true" stylesheets="@../css/scroll.css" VBox.vgrow="ALWAYS" />
717. </VBox>
718. **DB.java**
719. package com.unindra.restoserver;
720. import org.sql2o.Sql2o;
721. public class DB {
722. public static Sql2o sql2o;
723. static {
724. sql2o = new Sql2o("jdbc:mysql://localhost/osaka", "root", "zxcasdqwe123");
725. try {
726. sql2o.open();
727. } catch (Exception e) {
728. sql2o = new Sql2o("jdbc:mysql://localhost/osaka", "root", "");}}}
729. **Laporan.java**
730. package com.unindra.restoserver;
731. import com.itextpdf.io.image.ImageDataFactory;
732. import com.itextpdf.kernel.font.PdfFont;
733. import com.itextpdf.kernel.font.PdfFontFactory;
734. import com.itextpdf.kernel.geom.PageSize;
735. import com.itextpdf.kernel.geom.Rectangle;
736. import com.itextpdf.kernel.pdf.PdfDocument;
737. import com.itextpdf.kernel.pdf.PdfWriter;
738. import com.itextpdf.layout.Document;
739. import com.itextpdf.layout.borders.Border;
740. import com.itextpdf.layout.element.Image;
741. import com.itextpdf.layout.element.\*;
742. import com.itextpdf.layout.property.HorizontalAlignment;
743. import com.itextpdf.layout.property.TextAlignment;
744. import com.itextpdf.layout.property.UnitValue;
745. import com.itextpdf.layout.property.VerticalAlignment;
746. import com.unindra.restoserver.models.Menu;
747. import com.unindra.restoserver.models.Pesanan;
748. import com.unindra.restoserver.models.PesananService;
749. import com.unindra.restoserver.models.Transaksi;
750. import javafx.collections.FXCollections;
751. import org.joda.time.LocalDate;
752. import org.joda.time.LocalTime;
753. import javax.imageio.ImageIO;
754. import java.awt.\*;
755. import java.awt.image.BufferedImage;
756. import java.io.ByteArrayOutputStream;
757. import java.io.File;
758. import java.io.IOException;
759. import java.util.Arrays;
760. import java.util.Date;
761. import java.util.List;
762. import java.util.concurrent.atomic.AtomicInteger;
763. import static com.unindra.restoserver.Rupiah.rupiah;
764. import static com.unindra.restoserver.models.Menu.getMenus;
765. import static com.unindra.restoserver.models.Menu.menu;
766. import static com.unindra.restoserver.models.Pesanan.getPesanan;
767. import static com.unindra.restoserver.models.Transaksi.getTotalBayar;
768. import static com.unindra.restoserver.models.Transaksi.getTransaksiList;
769. public class Laporan {
770. private static final String bold = "fonts/OpenSans-Bold.ttf";
771. private static Table kop\_surat(String judul) throws IOException {
772. BufferedImage image = ImageIO.read(
773. Laporan.class.getResourceAsStream("/icons/logo-ramen-bulet-merah-copy50x50.png"));
774. ByteArrayOutputStream baos = new ByteArrayOutputStream();
775. ImageIO.write(image, "png", baos);
776. baos.flush();
777. byte[] imageInByte = baos.toByteArray();
778. baos.close();
779. Image img = new Image(ImageDataFactory.create(imageInByte));
780. return new Table(new UnitValue[]{
781. new UnitValue(UnitValue.PERCENT, 10),
782. new UnitValue(UnitValue.PERCENT, 90)}, true)
783. .setFontSize(12)
784. .addCell(cellNoBorder(img.setAutoScale(true)))
785. .addCell(cellNoBorder("Osaka Ramen\n" + judul)
786. .setTextAlignment(TextAlignment.CENTER)
787. .setHorizontalAlignment(HorizontalAlignment.CENTER)
788. .setVerticalAlignment(VerticalAlignment.MIDDLE));}
789. private static Table signature(LocalDate tgl) {
790. return new Table(1)
791. .setFontSize(10)
792. .setWidth(130)
793. .setHeight(80)
794. .setMarginTop(10)
795. .setHorizontalAlignment(HorizontalAlignment.RIGHT)
796. .addCell(cellNoBorder("Depok" + ", " +
797. hari().get(tgl.getDayOfWeek()) + ", " +
798. tgl.getDayOfMonth() + " " +
799. bulan().get(tgl.getMonthOfYear()) + " " +
800. tgl.getYear()).setTextAlignment(TextAlignment.CENTER))
801. .addCell(cellNoBorder("Pemilik\nTaufik Hidayat")
802. .setTextAlignment(TextAlignment.CENTER)
803. .setVerticalAlignment(VerticalAlignment.BOTTOM));}
804. private static Cell cellNoBorder(String text) {
805. return new Cell()
806. .setBorder(Border.NO\_BORDER)
807. .add(new Paragraph(text));}
808. private static Cell cellNoBorder(Image image) {
809. return new Cell()
810. .setBorder(Border.NO\_BORDER)
811. .add(image);}
812. private static Cell cell(String text) {
813. return new Cell().add(new Paragraph(text));}
814. private static List<String> hari() {
815. return Arrays.asList("","Senin","Selasa","Rabu","Kamis","Jumat","Sabtu","Minggu");}
816. private static List<String> bulan() {
817. return Arrays.asList("","Januari","Februari","Maret","April","Mei","Juni","Juli","Agustus","September","Oktober","November","Desember");}
818. private static void showReport(String fileName) {
819. File file = new File(fileName);
820. Desktop desktop = Desktop.getDesktop();
821. try {
822. desktop.open(file);
823. } catch (IOException e) {
824. e.printStackTrace();}}
825. private static boolean makeDir() {
826. File file = new File(System.getProperty("user.home") + "\\Documents\\LaporanResto");
827. if (!file.exists()) return file.mkdir();
828. else return true;}
829. public static void pemesanan(LocalDate dari, LocalDate sampai) throws IOException {
830. PdfFont boldFont = PdfFontFactory.createFont(bold, true);
831. LocalDate localDate = new LocalDate(new Date());
832. String fileName = String.format("%s\\Documents\\LaporanResto\\laporan-pemesanan-%s.pdf",System.getProperty("user.home"),localDate.toString());
833. if (makeDir()) {
834. PdfWriter writer = new PdfWriter(fileName);
835. PdfDocument pdf = new PdfDocument(writer);
836. Document document = new Document(pdf, PageSize.A4);
837. document.add(kop\_surat("Laporan Pemesanan"));
838. Table table = new Table(6)
839. .setWidth(520)
840. .setMarginTop(0)
841. .setFontSize(10)
842. .addHeaderCell(cell("Pukul").setFont(boldFont))
843. .addHeaderCell(cell("No Meja").setFont(boldFont))
844. .addHeaderCell(cell("Nama Menu").setFont(boldFont))
845. .addHeaderCell(cell("Jumlah").setFont(boldFont))
846. .addHeaderCell(cell("Harga").setFont(boldFont))
847. .addHeaderCell(cell("Total Harga").setFont(boldFont));
848. while (dari.isBefore(sampai.plusDays(1))) {
849. getTransaksiList(dari).forEach(transaksi -> Pesanan.getPesanan(transaksi).forEach(pesanan -> {
850. LocalTime t = new LocalTime(transaksi.getTanggal());
851. table.addCell(cell(String.format("%d:%d WIB", t.getHourOfDay(),
852. t.getMinuteOfHour())));
853. table.addCell(cell(transaksi.getNo\_meja()));
854. table.addCell(cell(pesanan.getNama\_menu()));
855. table.addCell(cell(String.valueOf(pesanan.getJumlah())));
856. table.addCell(cell(rupiah(menu(pesanan).getHarga\_menu())));
857. table.addCell(cell(rupiah(pesanan.getTotal())));}));
858. dari = dari.plusDays(1);}
859. document.add(table);
860. document.add(signature(localDate));
861. document.close();
862. showReport(fileName);}}
863. public static void pemasukan(LocalDate dari, LocalDate sampai) throws IOException {
864. PdfFont boldFont = PdfFontFactory.createFont(bold, true);
865. LocalDate localDate = new LocalDate(new Date());
866. String fileName = String.format(
867. "%s\\Documents\\LaporanResto\\laporan-pemasukan-%s.pdf",
868. System.getProperty("user.home"),
869. localDate.toString());
870. if (makeDir()) {
871. PdfWriter writer = new PdfWriter(fileName);
872. PdfDocument pdf = new PdfDocument(writer);
873. Document document = new Document(pdf, PageSize.A4);
874. document.add(kop\_surat("Laporan Pemasukan"));
875. Table table = new Table(2)
876. .setWidth(520)
877. .setMarginTop(0)
878. .setFontSize(10)
879. .addHeaderCell(cell("Tanggal").setFont(boldFont))
880. .addHeaderCell(cell("Total Pemasukan").setFont(boldFont));
881. while (dari.isBefore(sampai.plusDays(1))) {
882. table.addCell(cell(dari.toString()));
883. table.addCell(cell(rupiah(getTotalBayar(dari.getYear(), dari.getMonthOfYear()))));
884. dari = dari.plusDays(1);}
885. document.add(table);
886. document.add(signature(localDate));
887. document.close();
888. showReport(fileName);}}
889. public static void menuFavorit(LocalDate dari, LocalDate sampai) throws IOException{
890. PdfFont boldFont = PdfFontFactory.createFont(bold, true);
891. LocalDate localDate = new LocalDate(new Date());
892. String fileName = String.format(
893. "%s\\Documents\\LaporanResto\\laporan-menu-favorit-%s.pdf",
894. System.getProperty("user.home"),
895. localDate.toString());
896. if (makeDir()) {
897. PdfWriter writer = new PdfWriter(fileName);
898. PdfDocument pdf = new PdfDocument(writer);
899. Document document = new Document(pdf, PageSize.A4);
900. document.add(kop\_surat("Laporan Menu Favorit"));
901. Table table = new Table(4)
902. .setWidth(520)
903. .setMarginTop(0)
904. .setFontSize(10)
905. .addHeaderCell(cell("Nama Menu").setFont(boldFont))
906. .addHeaderCell(cell("Tipe").setFont(boldFont))
907. .addHeaderCell(cell("Harga").setFont(boldFont))
908. .addHeaderCell(cell("Total Dipesan").setFont(boldFont));
909. List<Menu> menus = FXCollections.observableArrayList(getMenus());
910. menus.sort((menu1, menu2) -> {
911. List<Pesanan> items1 = getPesanan(menu1);
912. List<Pesanan> items2 = getPesanan(menu2);
913. return items2.size() - items1.size();});
914. for (Menu menu : getMenus()) {
915. AtomicInteger jumlahMenu = new AtomicInteger();
916. LocalDate tgl = dari;
917. while (tgl.isBefore(sampai.plusDays(1))) {
918. for (Transaksi transaksi : getTransaksiList(tgl)) {
919. jumlahMenu.addAndGet(getPesanan(transaksi, menu).stream().mapToInt(Pesanan::getJumlah).sum());}
920. tgl = tgl.plusDays(1);}
921. if (jumlahMenu.get() > 0) {
922. table.addCell(cell(menu.getNama\_menu()));
923. table.addCell(cell(menu.getTipe()));
924. table.addCell(cell(rupiah(menu.getHarga\_menu())));
925. table.addCell(cell(String.valueOf(getPesanan(menu).size())));}}
926. document.add(table);
927. document.add(signature(localDate));
928. document.close();
929. showReport(fileName);}}
930. public static void kunjungan(LocalDate dari, LocalDate sampai) throws IOException {
931. PdfFont boldFont = PdfFontFactory.createFont(bold, true);
932. LocalDate localDate = new LocalDate(new Date());
933. String fileName = String.format("%s\\Documents\\LaporanResto\\laporan-kunjungan-%s.pdf",
934. System.getProperty("user.home"),
935. localDate.toString());
936. if (makeDir()) {
937. PdfWriter writer = new PdfWriter(fileName);
938. PdfDocument pdf = new PdfDocument(writer);
939. Document document = new Document(pdf, PageSize.A4);
940. document.add(kop\_surat("Laporan Kunjungan"));
941. Table table = new Table(2)
942. .setWidth(520)
943. .setMarginTop(0)
944. .setFontSize(10)
945. .addHeaderCell(cell("Tanggal").setFont(boldFont))
946. .addHeaderCell(cell("Total Kunjungan").setFont(boldFont));
947. while (dari.isBefore(sampai.plusDays(1))) {
948. int totalKunjungan = getTransaksiList(dari.getYear(), dari.getMonthOfYear()).size();
949. table.addCell(cell(dari.toString()));
950. table.addCell(cell(String.valueOf(totalKunjungan)));
951. dari = dari.plusDays(1);}
952. document.add(table);
953. document.add(signature(localDate));
954. document.close();
955. showReport(fileName);}}
956. public static void bill(Transaksi transaksi) throws IOException {
957. List<Pesanan> pesanans = PesananService.getItems(transaksi);
958. String fileName = String.format("%s\\Documents\\LaporanResto\\bill.pdf", System.getProperty("user.home"));
959. PdfFont boldFont = PdfFontFactory.createFont(bold, true);
960. LocalDate localDate = new LocalDate(new Date());
961. if (makeDir()) {
962. PdfWriter writer = new PdfWriter(fileName);
963. PdfDocument pdf = new PdfDocument(writer);
964. Document document = new Document(pdf, new PageSize(new Rectangle(226.8f, 600f)));
965. document.add(
966. new Paragraph()
967. .setTextAlignment(TextAlignment.CENTER)
968. .setFontSize(5)
969. .add(new Text("OSAKA RAMEN").setFont(boldFont))
970. .add("\n-----------------------------------------------------------------------------------------\n")
971. .add("No Meja:")
972. .add(pesanans.get(0).getNo\_meja()).add("\tTanggal:")
973. .add(localDate + " " + new LocalTime().toString().substring(0, 8))
974. .add("\n-----------------------------------------------------------------------------------------\n"));
975. Table table = new Table(new UnitValue[]{
976. new UnitValue(UnitValue.PERCENT, 30),
977. new UnitValue(UnitValue.PERCENT, 20),
978. new UnitValue(UnitValue.PERCENT, 50),}, true)
979. .setFontSize(6)
980. .setTextAlignment(TextAlignment.CENTER);
981. pesanans.forEach(item -> {
982. table.addCell(cellNoBorder(menu(item).getNama\_menu()));
983. table.addCell(cellNoBorder(item.getJumlah() + "x"));
984. table.addCell(cellNoBorder(rupiah(item.getTotal())));});
985. document.add(table);
986. document.add(
987. new Paragraph()
988. .setTextAlignment(TextAlignment.CENTER)
989. .setFontSize(5)
990. .add("-----------------------------------------------------------------------------------------"));
991. Table footerTable = new Table(new UnitValue[]{
992. new UnitValue(UnitValue.PERCENT, 50),
993. new UnitValue(UnitValue.PERCENT, 50),}, true)
994. .setTextAlignment(TextAlignment.RIGHT)
995. .addCell(cellNoBorder("Total").setFontSize(6))
996. .addCell(cellNoBorder(rupiah(transaksi.getTotalBayarFromService())).setFontSize(6));
997. document.add(footerTable);
998. document.close();
999. showReport(fileName);}}
1000. public static void struk(Transaksi transaksi, int tunai) throws IOException {
1001. List<Pesanan> pesanans = PesananService.getItems(transaksi);
1002. String fileName = String.format("%s\\Documents\\LaporanResto\\struk.pdf", System.getProperty("user.home"));
1003. PdfFont boldFont = PdfFontFactory.createFont(bold, true);
1004. LocalDate localDate = new LocalDate(new Date());
1005. if (makeDir()) {
1006. PdfWriter writer = new PdfWriter(fileName);
1007. PdfDocument pdf = new PdfDocument(writer);
1008. Document document = new Document(pdf, new PageSize(new Rectangle(226.8f, 600f)));
1009. document.add(
1010. new Paragraph()
1011. .setTextAlignment(TextAlignment.CENTER)
1012. .setFontSize(5)
1013. .add(new Text("OSAKA RAMEN").setFont(boldFont))
1014. .add("\nJl. Keadilan No. 23G, Rangkapan Jaya Baru, Pancoran Mas, Kota Depok Jawa Barat")
1015. .add("\n-----------------------------------------------------------------------------------------\n")
1016. .add("No Meja:")
1017. .add(pesanans.get(0).getNo\_meja())
1018. .add("\tTanggal:")
1019. .add(localDate + " " + new LocalTime().toString().substring(0, 8))
1020. .add("\n-----------------------------------------------------------------------------------------\n"));
1021. Table table = new Table(new UnitValue[]{
1022. new UnitValue(UnitValue.PERCENT, 30),
1023. new UnitValue(UnitValue.PERCENT, 20),
1024. new UnitValue(UnitValue.PERCENT, 50),}, true)
1025. .setFontSize(6)
1026. .setTextAlignment(TextAlignment.CENTER);
1027. pesanans.forEach(item -> {
1028. table.addCell(cellNoBorder(menu(item).getNama\_menu()));

table.addCell(cellNoBorder(item.getJumlah() + "x"));

table.addCell(cellNoBorder(rupiah(item.getTotal())));});

1. document.add(table);
2. document.add(
3. new Paragraph()
4. .setTextAlignment(TextAlignment.CENTER)
5. .setFontSize(5)
6. .add("-----------------------------------------------------------------------------------------"));
7. Table footerTable = new Table(new UnitValue[]{
8. new UnitValue(UnitValue.PERCENT, 50),
9. new UnitValue(UnitValue.PERCENT, 50),}, true)
10. .setTextAlignment(TextAlignment.RIGHT)
11. .addCell(cellNoBorder("Total").setFontSize(6))
12. .addCell(cellNoBorder(rupiah(transaksi.getTotalBayarFromService())).setFontSize(6))
13. .addCell(cellNoBorder("Tunai").setFontSize(6))
14. .addCell(cellNoBorder(rupiah(tunai)).setFontSize(6))
15. .addCell(cellNoBorder("Kembali").setFontSize(6))
16. .addCell(cellNoBorder(rupiah(tunai - transaksi.getTotalBayarFromService())).setFontSize(6));
17. document.add(footerTable);
18. document.close();
19. showReport(fileName);}}}
20. **Router.java**
21. package com.unindra.restoserver;
22. import com.google.gson.\*;
23. import com.jfoenix.controls.datamodels.treetable.RecursiveTreeObject;
24. import com.unindra.restoserver.models.\*;
25. import javafx.collections.FXCollections;
26. import static com.unindra.restoserver.models.Level.getLevels;
27. import static com.unindra.restoserver.models.Menu.getMenus;
28. import static com.unindra.restoserver.models.PesananService.getItems;
29. import static spark.Spark.\*;
30. class Router {
31. private static Gson gson;
32. static {
33. gson = new GsonBuilder().addSerializationExclusionStrategy(new ExclusionStrategy(){
34. @Override
35. public boolean shouldSkipField(FieldAttributes fieldAttributes) {

return fieldAttributes.getDeclaringClass().equals(RecursiveTreeObject.class);}

1. @Override
2. public boolean shouldSkipClass(Class<?> aClass) {

return false;}

1. }).addDeserializationExclusionStrategy(new ExclusionStrategy() {
2. @Override
3. public boolean shouldSkipField(FieldAttributes fieldAttributes) {

return fieldAttributes.getDeclaringClass().equals(RecursiveTreeObject.class);}

1. @Override
2. public boolean shouldSkipClass(Class<?> aClass) {

return false;

1. }}).create();
2. post("/pesanan", (request, response) -> {
3. response.type("application/json");
4. Pesanan pesanan = gson.fromJson(request.body(), Pesanan.class);
5. PesananService.add(pesanan);
6. return gson.toJson(new StandardResponse(StatusResponse.SUCCESS));});
7. get("/pesanan/:no\_meja", (request, response) -> {
8. response.type("application/json");
9. JsonElement jsonElement = gson.toJsonTree(getItems(request.params(":no\_meja")));
10. return gson.toJson(new StandardResponse(StatusResponse.SUCCESS, jsonElement));});
11. put("/pesanan", (request, response) -> {
12. response.type("application/json");
13. Pesanan pesanan = gson.fromJson(request.body(), Pesanan.class);
14. pesanan.setChildren(FXCollections.observableArrayList());
15. if (PesananService.update(pesanan))
16. return gson.toJson(new StandardResponse(StatusResponse.SUCCESS, "Pesanan diedit"));
17. return gson.toJson(new StandardResponse(StatusResponse.ERROR, "Pesanan gagal dihapus"));});
18. delete("/pesanan", (request, response) -> {
19. response.type("application/json");
20. Pesanan pesanan = gson.fromJson(request.body(), Pesanan.class);
21. if (PesananService.delete(pesanan))
22. return gson.toJson(new StandardResponse(StatusResponse.SUCCESS, "Pesanan dihapus"));
23. return gson.toJson(new StandardResponse(StatusResponse.ERROR, "Pesanan gagal dihapus"));});
24. get("/menus", (request, response) -> {
25. response.type("application/json");
26. return gson.toJson(new StandardResponse(
27. StatusResponse.SUCCESS,
28. gson.toJsonTree(getMenus())));});
29. get("/levels", (request, response) -> {
30. response.type("application/json");
31. return gson.toJson(new StandardResponse(
32. StatusResponse.SUCCESS,
33. gson.toJsonTree(getLevels())));});
34. get("/bayar/:no\_meja", (request, response) -> {
35. response.type("application/json");
36. Transaksi transaksi = new Transaksi(request.params(":no\_meja"));
37. TransaksiService.add(transaksi);
38. return gson.toJson(new StandardResponse(StatusResponse.SUCCESS));});
39. get("/detail\_ramen/:nama\_menu", (request, response) -> {
40. response.type("application/json");
41. return gson.toJson(new StandardResponse(StatusResponse.SUCCESS,
42. gson.toJsonTree(DetailRamen.detailRamen(request.params(":nama\_menu")))));});}}
43. **DaftarMenuController.java**
44. package com.unindra.restoserver.controllers;
45. import com.jfoenix.controls.\*;
46. import com.jfoenix.controls.datamodels.treetable.RecursiveTreeObject;
47. import com.unindra.restoserver.Dialog;
48. import com.unindra.restoserver.models.DetailRamen;
49. import com.unindra.restoserver.models.Level;
50. import com.unindra.restoserver.models.Menu;
51. import javafx.collections.FXCollections;
52. import javafx.collections.ObservableList;
53. import javafx.fxml.Initializable;
54. import javafx.scene.control.Label;
55. import javafx.scene.control.TreeItem;
56. import javafx.scene.control.TreeTableColumn;
57. import javafx.scene.control.TreeTableView;
58. import javafx.scene.input.MouseEvent;
59. import javafx.scene.layout.HBox;
60. import javafx.stage.FileChooser;
61. import javafx.stage.Stage;
62. import java.io.File;
63. import java.io.IOException;
64. import java.net.URL;
65. import java.nio.file.Files;
66. import java.util.ResourceBundle;
67. import static com.unindra.restoserver.models.Level.getLevels;
68. import static com.unindra.restoserver.models.Menu.getMenus;
69. public class DaftarMenuController implements Initializable {
70. public JFXTreeTableView<Menu> menuTableView;
71. public JFXButton actionButton;
72. public JFXButton hapusButton;
73. public JFXTextField namaField;
74. public JFXTextField hargaField;
75. public JFXComboBox<String> tipeComboBox;
76. public JFXTextArea deskArea;
77. public Label titleLabel;
78. public JFXTreeTableView<Level> levelTableView;
79. public JFXTextField hargaLevelField;
80. public JFXTextField levelField;
81. public HBox formForRamenPane;
82. public JFXButton pilihGambarButton;
83. private ObservableList<String> tipeList;
84. private Menu menu;
85. private Level level;
86. private DetailRamen detailRamen;
87. @Override
88. public void initialize(URL location, ResourceBundle resources) {
89. TreeTableColumn<Menu, String> namaCol = new TreeTableColumn<>("Nama");
90. TreeTableColumn<Menu, String> tipeCol = new TreeTableColumn<>("Tipe");
91. TreeTableColumn<Menu, String> hargaCol = new TreeTableColumn<>("Harga");
92. namaCol.setCellValueFactory(param -> param.getValue().getValue().nama\_menuProperty());
93. tipeCol.setCellValueFactory(param -> param.getValue().getValue().tipeProperty());
94. hargaCol.setCellValueFactory(param -> param.getValue().getValue().harga\_menuProperty());
95. TreeItem<Menu> rootMenu = new RecursiveTreeItem<>(getMenus(), RecursiveTreeObject::getChildren);
96. menuTableView.setRoot(rootMenu);
97. menuTableView.getColumns().add(namaCol);
98. menuTableView.getColumns().add(tipeCol);
99. menuTableView.getColumns().add(hargaCol);
100. menuTableView.setColumnResizePolicy(TreeTableView.CONSTRAINED\_RESIZE\_POLICY);
101. tipeList = FXCollections.observableArrayList("ramen","minuman", "cemilan", "lainnya");
102. tipeComboBox.setItems(tipeList);
103. TreeTableColumn<Level, Integer> levelCol = new TreeTableColumn<>("Level");
104. TreeTableColumn<Level, String> hargaLevelCol = new TreeTableColumn<>("Harga");
105. levelCol.setCellValueFactory(param -> param.getValue().getValue().levelProperty());
106. hargaLevelCol.setCellValueFactory(param -> param.getValue().getValue().harga\_levelProperty());
107. TreeItem<Level> rootLevel = new RecursiveTreeItem<>(getLevels(), RecursiveTreeObject::getChildren);
108. levelTableView.setRoot(rootLevel);
109. levelTableView.getColumns().add(levelCol);
110. levelTableView.getColumns().add(hargaLevelCol);
111. levelTableView.setColumnResizePolicy(TreeTableView.CONSTRAINED\_RESIZE\_POLICY);
112. hargaField.textProperty().addListener((observable, oldValue, newValue) -> {
113. if (!newValue.matches("\\d\*")) {hargaField.setText(newValue.replaceAll("[^\\d]", ""));}});
114. hargaLevelField.textProperty().addListener((observable, oldValue, newValue) -> {
115. if (!newValue.matches("\\d\*")) {hargaLevelField.setText(newValue.replaceAll("[^\\d]", ""));}});}
116. public void actionHandle() {
117. if (actionButton.getText().equals("Tambah")) {
118. menu = new Menu(
119. namaField.getText(),
120. tipeComboBox.getSelectionModel().getSelectedItem(),
121. Integer.valueOf(hargaField.getText()));
122. if (menu.getTipe().equals("ramen")) {
123. if (detailRamen != null) {
124. detailRamen.setNama\_menu(namaField.getText());
125. detailRamen.setDeskripsi(deskArea.getText());
126. if (menu.add() && detailRamen.add()) {
127. getDialog().information("Berhasil!", "Menu berhasil ditambahkan");
128. reset();
129. } else getDialog().information("Gagal", "Menu gagal ditambahkan");
130. } else getDialog().information("Gagal", "Gambar belum dipilih");
131. } else {
132. if (menu.add()) {
133. getDialog().information("Berhasil!", "Menu berhasil ditambahkan");
134. reset();
135. } else getDialog().information("Gagal", "Menu gagal ditambahkan");}
136. } else { // Ubah
137. menu.setNama\_menu(namaField.getText());
138. menu.setTipe(tipeComboBox.getSelectionModel().getSelectedItem());
139. menu.setHarga\_menu(Integer.valueOf(hargaField.getText()));
140. if (menu.getTipe().equals("ramen")) {
141. detailRamen.setNama\_menu(menu.getNama\_menu());
142. detailRamen.setDeskripsi(deskArea.getText());
143. if (menu.update() && detailRamen.update()) {
144. getDialog().information("Berhasil!", "Menu berhasil diubah");
145. reset();
146. } else getDialog().information("Gagal", "Menu gagal diubah");
147. } else {
148. if (menu.update()) {
149. getDialog().information("Berhasil!", "Menu berhasil diubah");
150. reset();
151. } else getDialog().information("Gagal", "Menu gagal diubah");}}}
152. public void ubahLevelHandle() {
153. level.setHarga\_level(Integer.parseInt(hargaLevelField.getText()));
154. if (level.update()) {
155. getDialog().information("Berhasil!", "Level berhasil diubah");
156. reset();}}
157. public void pilihHandle(MouseEvent mouseEvent) {
158. if (!menuTableView.getSelectionModel().isEmpty()) {
159. menu = menuTableView.getSelectionModel().getSelectedItem().getValue();
160. detailRamen = DetailRamen.detailRamen(menu);
161. namaField.setText(menu.getNama\_menu());
162. int index = tipeList.indexOf(menu.getTipe());
163. tipeComboBox.getSelectionModel().clearAndSelect(index);
164. hargaField.setText(String.valueOf(menu.getHarga\_menu()));
165. if (detailRamen != null) deskArea.setText(detailRamen.getDeskripsi());
166. titleLabel.setText("UBAH MENU");
167. namaField.setDisable(true);
168. hapusButton.setVisible(true);
169. actionButton.setText("Ubah");
170. actionButton.getStyleClass().set(2, "ubah");}
171. if (mouseEvent.getClickCount() == 2) reset();}
172. public void pilihLevelHandle(MouseEvent mouseEvent) {
173. if (!levelTableView.getSelectionModel().isEmpty()) {
174. level = levelTableView.getSelectionModel().getSelectedItem().getValue();
175. levelField.setText(String.valueOf(level.getLevel()));
176. hargaLevelField.setText(String.valueOf(level.getHarga\_level()));
177. hargaLevelField.setDisable(false);}
178. if (mouseEvent.getClickCount() == 2) reset();}
179. public void hapusHandle() {
180. Dialog alert = getDialog();
181. alert.confirmation("Anda yakin ingin menghapus menu ini?",
182. event -> {
183. Menu menu = menuTableView.getSelectionModel().getSelectedItem().getValue();
184. if (menu.getTipe().equals("ramen")) {
185. if (menu.delete() && detailRamen.delete()) {
186. alert.information("Berhasil!", "Menu berhasil dihapus");
187. reset();
188. } else alert.information("Gagal", "Menu gagal dihapus");
189. } else {
190. if (menu.delete()) {alert.information("Berhasil!", "Menu berhasil dihapus");
191. reset();
192. } else alert.information("Gagal", "Menu gagal dihapus");}});}
193. private void reset() {
194. titleLabel.setText("TAMBAH MENU");
195. hapusButton.setVisible(false);
196. actionButton.setText("Tambah");
197. actionButton.getStyleClass().set(2, "tambah");
198. menuTableView.getSelectionModel().clearSelection();
199. namaField.setDisable(false);
200. namaField.setText("");
201. tipeComboBox.getSelectionModel().clearSelection();
202. hargaField.setText("");
203. deskArea.setText("");
204. levelField.setText("");
205. hargaLevelField.setText("");
206. hargaLevelField.setDisable(true);
207. levelTableView.getSelectionModel().clearSelection();
208. namaField.requestFocus();
209. pilihGambarButton.setText("Pilih gambar... (max : 2048 KB)");}
210. public void tipeHandle() {
211. if (tipeComboBox.getSelectionModel().getSelectedItem() != null) {
212. if (tipeComboBox.getSelectionModel().getSelectedItem().equals("ramen")) formForRamenPane.setDisable(false);
213. else {
214. formForRamenPane.setDisable(true);
215. detailRamen = null;
216. deskArea.setText("");
217. pilihGambarButton.setText("Pilih gambar... (max : 2048 KB)");}}}
218. public void pilihGambarHandle() throws IOException {
219. FileChooser fileChooser = new FileChooser();
220. fileChooser.setTitle("Open Resource File");
221. File file = fileChooser.showOpenDialog(actionButton.getScene().getWindow());
222. if (file != null) {
223. if (file.length() <= 2048000) {
224. pilihGambarButton.setText(file.getName());
225. detailRamen = new DetailRamen(Files.readAllBytes(file.toPath()));
226. } else {
227. getDialog().information("Gagal", "Ukuran foto terlalu besar");}}}
228. private Dialog getDialog() {
229. return new Dialog((Stage) actionButton.getScene().getWindow());}}
230. **LaporanController.java**
231. package com.unindra.restoserver.controllers;
232. import com.jfoenix.controls.JFXComboBox;
233. import com.jfoenix.controls.JFXDatePicker;
234. import com.jfoenix.controls.JFXTreeTableView;
235. import com.jfoenix.controls.RecursiveTreeItem;
236. import com.jfoenix.controls.datamodels.treetable.RecursiveTreeObject;
237. import com.unindra.restoserver.Laporan;
238. import com.unindra.restoserver.models.Menu;
239. import com.unindra.restoserver.models.Pesanan;
240. import com.unindra.restoserver.models.Transaksi;
241. import javafx.application.Platform;
242. import javafx.collections.FXCollections;
243. import javafx.collections.ListChangeListener;
244. import javafx.collections.ObservableList;
245. import javafx.fxml.Initializable;
246. import javafx.scene.chart.AreaChart;
247. import javafx.scene.chart.PieChart;
248. import javafx.scene.chart.XYChart;
249. import javafx.scene.control.TreeItem;
250. import javafx.scene.control.TreeTableCell;
251. import javafx.scene.control.TreeTableColumn;
252. import javafx.scene.control.TreeTableView;
253. import javafx.util.Callback;
254. import org.joda.time.LocalDate;
255. import java.io.IOException;
256. import java.net.URL;
257. import java.time.ZoneId;
258. import java.util.Date;
259. import java.util.ResourceBundle;
260. import java.util.concurrent.atomic.AtomicInteger;
261. import static com.unindra.restoserver.models.Menu.getMenus;
262. import static com.unindra.restoserver.models.Menu.menu;
263. import static com.unindra.restoserver.models.Pesanan.getPesanan;
264. import static com.unindra.restoserver.models.Transaksi.getTransaksi;
265. import static com.unindra.restoserver.models.Transaksi.getTransaksiList;
266. public class LaporanController implements Initializable {
267. public AreaChart pemasukanChart;
268. public PieChart menuFavChart;
269. public AreaChart kunjunganChart;
270. public JFXComboBox<String> pilihLaporanCombo;
271. public JFXDatePicker dariDatePicker;
272. public JFXDatePicker sampaiDatePicker;
273. public JFXTreeTableView<Pesanan> pemesananTableView;
274. @Override
275. public void initialize(URL location, ResourceBundle resources) {
276. TreeTableColumn<Pesanan, String> pukulCol = new TreeTableColumn<>("Pukul");
277. TreeTableColumn<Pesanan, String> mejaCol = new TreeTableColumn<>("No Meja");
278. TreeTableColumn<Pesanan, String> namaCol = new TreeTableColumn<>("Nama");
279. TreeTableColumn<Pesanan, Integer> jumlahCol = new TreeTableColumn<>("Jumlah");
280. TreeTableColumn<Pesanan, String> hargaCol = new TreeTableColumn<>("Harga");
281. TreeTableColumn<Pesanan, String> totalHargaCol = new TreeTableColumn<>("Total Harga");
282. pukulCol.setCellValueFactory(param -> getTransaksi(param.getValue().getValue()).pukulProperty());
283. pukulCol.setCellValueFactory(param -> getTransaksi(param.getValue().getValue()).pukulProperty());
284. mejaCol.setCellValueFactory(param -> getTransaksi(param.getValue().getValue()).no\_mejaProperty());
285. namaCol.setCellValueFactory(param -> menu(param.getValue().getValue()).nama\_menuProperty());
286. jumlahCol.setCellValueFactory(param -> param.getValue().getValue().jumlahProperty());
287. hargaCol.setCellValueFactory(param -> menu(param.getValue().getValue()).harga\_menuProperty());
288. totalHargaCol.setCellValueFactory(param -> param.getValue().getValue().totalHargaProperty());
289. namaCol.setCellFactory(new Callback<TreeTableColumn<Pesanan, String>, TreeTableCell<Pesanan, String>>() {
290. @Override
291. public TreeTableCell<Pesanan, String> call(TreeTableColumn<Pesanan, String> param) {
292. return new TreeTableCell<Pesanan, String>() {
293. @Override
294. protected void updateItem(String item, boolean empty) {
295. super.updateItem(item, empty);
296. if (item == null) {
297. setText(null);
298. } else {
299. Pesanan i = Pesanan.getPesananList().get(getIndex());
300. if (menu(i).getTipe().equals("ramen"))
301. setText(item + " lv." + i.getLevel());
302. else setText(item);}}};}});
303. TreeItem<Pesanan> rootItem = new RecursiveTreeItem<>(Pesanan.getPesananList(), RecursiveTreeObject::getChildren);
304. pemesananTableView.setRoot(rootItem);
305. pemesananTableView.getColumns().add(pukulCol);
306. pemesananTableView.getColumns().add(mejaCol);
307. pemesananTableView.getColumns().add(namaCol);
308. pemesananTableView.getColumns().add(jumlahCol);
309. pemesananTableView.getColumns().add(hargaCol);
310. pemesananTableView.getColumns().add(totalHargaCol);
311. pemesananTableView.setColumnResizePolicy(TreeTableView.CONSTRAINED\_RESIZE\_POLICY);
312. Platform.runLater(() -> {
313. pemasukanChart.getYAxis().setLabel("Pemasukan (Rp)");
314. menuFavChart.setStartAngle(90);
315. kunjunganChart.getYAxis().setLabel("Kunjungan");
316. });
317. ObservableList<String> pilihLaporanObservableList = FXCollections.observableArrayList("Semua","Pemesanan","Menu Favorit","Pemasukan","Kunjungan");
318. pilihLaporanCombo.setItems(pilihLaporanObservableList);
319. pilihLaporanCombo.getSelectionModel().select(0);
320. getTransaksiList().addListener(transaksiListChangeListener());}
321. private void pemesanan(LocalDate dari, LocalDate sampai) {
322. ObservableList<Pesanan> pesananList = FXCollections.observableArrayList();
323. while (dari.isBefore(sampai.plusDays(1))) {
324. getTransaksiList(dari).forEach(transaksi -> pesananList.addAll(getPesanan(transaksi)));
325. dari = dari.plusDays(1);}
326. Pesanan.getPesananList().setAll(pesananList);}
327. @SuppressWarnings("unchecked")
328. private void pemasukan(LocalDate dari, LocalDate sampai) {
329. XYChart.Series data = new XYChart.Series();
330. while (dari.isBefore(sampai.plusDays(1))) {
331. int totalPemasukan = getTransaksiList(dari).stream().mapToInt(Transaksi::getTotalBayar).sum();
332. LocalDate tgl = dari;
333. Platform.runLater(() -> data.getData().add(
334. new XYChart.Data(tgl.getDayOfMonth() + " " + tgl.monthOfYear().getAsText(), totalPemasukan)));
335. dari = dari.plusDays(1);}
336. Platform.runLater(() -> pemasukanChart.getData().setAll(data));}
337. private void menuFavorit(LocalDate dari, LocalDate sampai) {
338. ObservableList<PieChart.Data> menuFavData = FXCollections.observableArrayList();
339. ObservableList<Menu> menus = getMenus();
340. for (Menu menu : menus) {
341. AtomicInteger jumlahMenu = new AtomicInteger();
342. LocalDate tgl = dari;
343. while (tgl.isBefore(sampai.plusDays(1))) {
344. for (Transaksi transaksi : getTransaksiList(tgl)) {
345. jumlahMenu.addAndGet(getPesanan(transaksi, menu).stream().mapToInt(Pesanan::getJumlah).sum());}
346. tgl = tgl.plusDays(1);}
347. if (jumlahMenu.get() > 0)
348. Platform.runLater(() -> menuFavData.add(new PieChart.Data(menu.getNama\_menu(), jumlahMenu.get())));}
349. Platform.runLater(() -> menuFavChart.setData(menuFavData));}
350. @SuppressWarnings("unchecked")
351. private void kunjungan(LocalDate dari, LocalDate sampai) {
352. XYChart.Series data = new XYChart.Series();
353. while (dari.isBefore(sampai.plusDays(1))) {
354. int totalKunjungan = getTransaksiList(dari).size();
355. LocalDate tgl = dari;
356. Platform.runLater(() -> data.getData().add(
357. new XYChart.Data(tgl.getDayOfMonth() + " " + tgl.monthOfYear().getAsText(), totalKunjungan)));
358. dari = dari.plusDays(1);}
359. Platform.runLater(() -> kunjunganChart.getData().setAll(data));}
360. private ListChangeListener<Transaksi> transaksiListChangeListener() {
361. return c -> {
362. pemesanan(getDariDate(), getSampaiDate());
363. pemasukan(getDariDate(), getSampaiDate());
364. menuFavorit(getDariDate(), getSampaiDate());
365. kunjungan(getDariDate(), getSampaiDate());};}
366. private LocalDate getDariDate() {
367. if (dariDatePicker.getValue() == null) return new LocalDate().minusMonths(1);
368. else {
369. Date dari = Date.from(dariDatePicker.getValue().atStartOfDay(ZoneId.systemDefault()).toInstant());
370. return new LocalDate(dari);}}
371. private LocalDate getSampaiDate() {
372. if (sampaiDatePicker.getValue() == null) return new LocalDate();
373. else {
374. Date sampai = Date.from(sampaiDatePicker.getValue().atStartOfDay(ZoneId.systemDefault()).toInstant());
375. return new LocalDate(sampai);}}
376. public void cetakHarianHandle() {
377. Thread thread = new Thread(() -> {
378. try {
379. switch (pilihLaporanCombo.getSelectionModel().getSelectedItem()) {
380. case "Semua":
381. Laporan.pemesanan(getDariDate(), getSampaiDate());
382. Laporan.pemasukan(getDariDate(), getSampaiDate());
383. Laporan.menuFavorit(getDariDate(), getSampaiDate());
384. Laporan.kunjungan(getDariDate(), getSampaiDate());
385. break;
386. case "Pemesanan":
387. Laporan.pemesanan(getDariDate(), getSampaiDate());
388. break;
389. case "Menu Favorit":
390. Laporan.menuFavorit(getDariDate(), getSampaiDate());
391. break;
392. case "Pemasukan":
393. Laporan.pemasukan(getDariDate(), getSampaiDate());
394. break;
395. case "Kunjungan":
396. Laporan.kunjungan(getDariDate(), getSampaiDate());
397. break;}
398. } catch (IOException e) {
399. e.printStackTrace();}});
400. thread.start();}}
401. **SignInController.java**
402. package com.unindra.restoserver.controllers;
403. import com.jfoenix.controls.JFXPasswordField;
404. import com.jfoenix.controls.JFXTextField;
405. import com.unindra.restoserver.Dialog;
406. import com.unindra.restoserver.models.User;
407. import javafx.fxml.FXMLLoader;
408. import javafx.scene.Parent;
409. import javafx.scene.Scene;
410. import javafx.stage.Stage;
411. import java.io.IOException;
412. public class SignInController {
413. public JFXTextField usernameField;
414. public JFXPasswordField passwordField;
415. public void signInAction() throws IOException {
416. User user = User.user(usernameField.getText());
417. if (user != null) {
418. if (user.getPassword().equals(passwordField.getText())) {
419. FXMLLoader fxmlLoader = new FXMLLoader(getClass().getResource("/fxml/app.fxml"));
420. Parent parent = fxmlLoader.load();
421. ((AppController) fxmlLoader.getController()).setUser(user);
422. getStage().setScene(new Scene(parent));
423. } else {
424. getDialog().information("Error", "Password salah");
425. passwordField.requestFocus();}
426. } else {
427. getDialog().information("Error", "Username tidak ditemukan");
428. usernameField.requestFocus();}}
429. private Stage getStage() {
430. return (Stage) usernameField.getScene().getWindow();}
431. private Dialog getDialog() {
432. return new Dialog(getStage());}}
433. **UtamaController.java**
434. package com.unindra.restoserver.controllers;
435. import com.jfoenix.controls.JFXButton;
436. import com.jfoenix.controls.JFXTextField;
437. import com.jfoenix.controls.JFXTreeTableView;
438. import com.jfoenix.controls.RecursiveTreeItem;
439. import com.jfoenix.controls.datamodels.treetable.RecursiveTreeObject;
440. import com.unindra.restoserver.Dialog;
441. import com.unindra.restoserver.Laporan;
442. import com.unindra.restoserver.models.Pesanan;
443. import com.unindra.restoserver.models.Transaksi;
444. import javafx.application.Platform;
445. import javafx.beans.property.SimpleStringProperty;
446. import javafx.collections.ListChangeListener;
447. import javafx.collections.transformation.FilteredList;
448. import javafx.fxml.Initializable;
449. import javafx.scene.control.TreeItem;
450. import javafx.scene.control.TreeTableCell;
451. import javafx.scene.control.TreeTableColumn;
452. import javafx.scene.control.TreeTableView;
453. import javafx.stage.Stage;
454. import javafx.util.Callback;
455. import java.io.IOException;
456. import java.net.URL;
457. import java.util.ResourceBundle;
458. import java.util.function.Predicate;
459. import static com.unindra.restoserver.models.PesananService.\*;
460. import static com.unindra.restoserver.models.Menu.menu;
461. import static com.unindra.restoserver.models.TransaksiService.getTransaksiList;
462. public class UtamaController implements Initializable {
463. public JFXTreeTableView<Pesanan> pesananTableView;
464. public JFXTreeTableView<Transaksi> pembayaranTableView;
465. @Override
466. public void initialize(URL location, ResourceBundle resources) {
467. TreeTableColumn<Pesanan, String> mejaCol = new TreeTableColumn<>("No Meja");
468. TreeTableColumn<Pesanan, String> namaCol = new TreeTableColumn<>("Nama");
469. TreeTableColumn<Pesanan, Integer> jumlahCol = new TreeTableColumn<>("Jumlah");
470. TreeTableColumn<Pesanan, String> terimaCol = new TreeTableColumn<>("Terima");
471. TreeTableColumn<Pesanan, String> tolakCol = new TreeTableColumn<>("Tolak");
472. mejaCol.setCellValueFactory(param -> param.getValue().getValue().no\_mejaProperty());
473. namaCol.setCellValueFactory(param -> menu(param.getValue().getValue()).nama\_menuProperty());
474. jumlahCol.setCellValueFactory(param -> param.getValue().getValue().jumlahProperty());
475. terimaCol.setCellValueFactory(param -> new SimpleStringProperty(""));
476. tolakCol.setCellValueFactory(param -> new SimpleStringProperty(""));
477. namaCol.setCellFactory(new Callback<TreeTableColumn<Pesanan, String>, TreeTableCell<Pesanan, String>>() {
478. @Override
479. public TreeTableCell<Pesanan, String> call(TreeTableColumn<Pesanan, String> param) {
480. return new TreeTableCell<Pesanan, String>() {
481. @Override
482. protected void updateItem(String item, boolean empty) {
483. super.updateItem(item, empty);
484. if (item == null) {
485. setText(null);
486. } else {
487. Pesanan i = getPesananList().get(getIndex());
488. if (menu(i).getTipe().equals("ramen"))
489. setText(item + " lv." + i.getLevel());
490. else setText(item);}}};}});
491. terimaCol.setCellFactory(new Callback<TreeTableColumn<Pesanan, String>, TreeTableCell<Pesanan, String>>() {
492. @Override
493. public TreeTableCell<Pesanan, String> call(TreeTableColumn<Pesanan, String> param) {
494. return new TreeTableCell<Pesanan, String>() {
495. final JFXButton button = new JFXButton("Terima");
496. @Override
497. protected void updateItem(String item, boolean empty) {
498. super.updateItem(item, empty);
499. if (item == null) {
500. setGraphic(null);
501. setText(null);
502. } else {
503. button.getStyleClass().add("terima");
504. button.setOnAction(event -> {
505. Pesanan i = pesananTableView.getRoot().getChildren().get(getIndex()).getValue();
506. i.terima();
507. update(i);});
508. setGraphic(button);
509. setText(null);}}};}});
510. tolakCol.setCellFactory(new Callback<TreeTableColumn<Pesanan, String>, TreeTableCell<Pesanan, String>>() {
511. @Override
512. public TreeTableCell<Pesanan, String> call(TreeTableColumn<Pesanan, String> param) {
513. return new TreeTableCell<Pesanan, String>() {
514. final JFXButton button = new JFXButton("Tolak");
515. @Override
516. protected void updateItem(String item, boolean empty) {
517. super.updateItem(item, empty);
518. if (item == null) {
519. setGraphic(null);
520. setText(null);
521. } else {
522. button.getStyleClass().add("tolak");
523. button.setOnAction(event -> {
524. Dialog alert = new Dialog((Stage) pesananTableView.getScene().getWindow());
525. Pesanan i = pesananTableView.getRoot().getChildren().get(getIndex()).getValue();
526. alert.confirmation("Anda yakin ingin menolak pesanan ini?",
527. e -> {
528. delete(i);
529. alert.getDialog().hide();});});
530. setGraphic(button);
531. setText(null);}}};}});
532. Predicate<Pesanan> predicate = item -> item.getStatus\_item().equals("dipesan");
533. FilteredList<Pesanan> filteredList = new FilteredList<>(getPesananList(), predicate);
534. getPesananList().addListener((ListChangeListener<Pesanan>) c -> filteredList.setPredicate(predicate));
535. TreeItem<Pesanan> rootItem = new RecursiveTreeItem<>(filteredList, RecursiveTreeObject::getChildren);
536. pesananTableView.setRoot(rootItem);
537. pesananTableView.getColumns().add(mejaCol);
538. pesananTableView.getColumns().add(namaCol);
539. pesananTableView.getColumns().add(jumlahCol);
540. pesananTableView.getColumns().add(terimaCol);
541. pesananTableView.getColumns().add(tolakCol);
542. pesananTableView.setColumnResizePolicy(TreeTableView.CONSTRAINED\_RESIZE\_POLICY);
543. TreeTableColumn<Transaksi, String> mejaTransaksiCol = new TreeTableColumn<>("No Meja");
544. TreeTableColumn<Transaksi, String> totalCol = new TreeTableColumn<>("Total Harga");
545. TreeTableColumn<Transaksi, String> billCol = new TreeTableColumn<>("Bill");
546. TreeTableColumn<Transaksi, String> strukCol = new TreeTableColumn<>("Struk");
547. TreeTableColumn<Transaksi, String> simpanCol = new TreeTableColumn<>("Simpan");
548. mejaTransaksiCol.setCellValueFactory(param -> param.getValue().getValue().no\_mejaProperty());
549. totalCol.setCellValueFactory(param -> param.getValue().getValue().totalProperty());
550. billCol.setCellValueFactory(param -> new SimpleStringProperty(""));
551. strukCol.setCellValueFactory(param -> new SimpleStringProperty(""));
552. simpanCol.setCellValueFactory(param -> new SimpleStringProperty(""));
553. billCol.setCellFactory(new Callback<TreeTableColumn<Transaksi, String>, TreeTableCell<Transaksi, String>>() {
554. @Override
555. public TreeTableCell<Transaksi, String> call(TreeTableColumn<Transaksi, String> param) {
556. return new TreeTableCell<Transaksi, String>() {
557. final JFXButton button = new JFXButton("Cetak");
558. @Override
559. protected void updateItem(String item, boolean empty) {
560. super.updateItem(item, empty);
561. if (item == null) {
562. setGraphic(null);
563. setText(null);
564. } else {
565. button.getStyleClass().add("print-20");
566. button.setOnAction(event -> {
567. Thread thread = new Thread(() -> {
568. Transaksi transaksi = getTransaksiList().get(getIndex());
569. try {
570. Laporan.bill(transaksi);
571. } catch (IOException e) {
572. e.printStackTrace();}});
573. thread.start();});
574. setGraphic(button);
575. setText(null);}}};}});
576. strukCol.setCellFactory(new Callback<TreeTableColumn<Transaksi, String>, TreeTableCell<Transaksi, String>>() {
577. @Override
578. public TreeTableCell<Transaksi, String> call(TreeTableColumn<Transaksi, String> param) {
579. return new TreeTableCell<Transaksi, String>() {
580. final JFXButton button = new JFXButton("Cetak");
581. @Override
582. protected void updateItem(String item, boolean empty) {
583. super.updateItem(item, empty);
584. if (item == null) {
585. setGraphic(null);
586. setText(null);
587. } else {
588. button.getStyleClass().add("print-20");
589. button.setOnAction(event -> {
590. Dialog jumlahTunaiDialog = new Dialog((Stage) pesananTableView.getScene().getWindow());
591. JFXTextField tunaiField = new JFXTextField();
592. tunaiField.textProperty().addListener((observable, oldValue, newValue) -> {
593. if (!newValue.matches("\\d\*")) {
594. tunaiField.setText(newValue.replaceAll("[^\\d]", ""));}});
595. jumlahTunaiDialog.input(
596. tunaiField,
597. e -> {
598. Thread thread = new Thread(() -> {
599. Transaksi transaksi = getTransaksiList().get(getIndex());
600. try {
601. int tunai = Integer.parseInt(tunaiField.getText());
602. if (tunai >= transaksi.getTotalBayarFromService())
603. Laporan.struk(transaksi, tunai);
604. else {
605. Platform.runLater(() -> {
606. Dialog dialog = new Dialog(
607. (Stage) pesananTableView
608. .getScene().getWindow());
609. dialog.information("Error","Jumlah tunai tidak mencukupi total pembayaran");});}
610. } catch (IOException ex) {
611. ex.printStackTrace();}});
612. thread.start();
613. jumlahTunaiDialog.getDialog().hide();});});
614. setGraphic(button);
615. setText(null);}}};}});
616. simpanCol.setCellFactory(new Callback<TreeTableColumn<Transaksi, String>, TreeTableCell<Transaksi, String>>() {
617. @Override
618. public TreeTableCell<Transaksi, String> call(TreeTableColumn<Transaksi, String> param) {
619. return new TreeTableCell<Transaksi, String>() {
620. final JFXButton button = new JFXButton("Simpan");
621. @Override
622. protected void updateItem(String item, boolean empty) {
623. super.updateItem(item, empty);
624. if (item == null) {
625. setGraphic(null);
626. setText(null);
627. } else {
628. button.getStyleClass().add("simpan");
629. button.setOnAction(event -> {
630. Dialog confirmDialog = new Dialog((Stage) pesananTableView.getScene().getWindow());
631. confirmDialog.confirmation("Transaksi sudah selesai?",
632. e -> {
633. Transaksi transaksi = getTransaksiList().get(getIndex());
634. transaksi.simpan();
635. confirmDialog.getDialog().hide();});});
636. setGraphic(button);
637. setText(null);}}};}});
638. TreeItem<Transaksi> rootTrans = new RecursiveTreeItem<>(getTransaksiList(), RecursiveTreeObject::getChildren);
639. pembayaranTableView.setRoot(rootTrans);
640. pembayaranTableView.getColumns().add(mejaTransaksiCol);
641. pembayaranTableView.getColumns().add(totalCol);
642. pembayaranTableView.getColumns().add(billCol);
643. pembayaranTableView.getColumns().add(strukCol);
644. pembayaranTableView.getColumns().add(simpanCol);
645. pembayaranTableView.setColumnResizePolicy(TreeTableView.CONSTRAINED\_RESIZE\_POLICY);}}
646. **DetailRamen.java**
647. package com.unindra.restoserver.models;
648. import com.unindra.restoserver.DB;
649. import org.sql2o.Connection;
650. import java.util.Arrays;
651. import java.util.List;
652. public class DetailRamen {
653. private String nama\_menu;
654. private byte[] foto;
655. private String deskripsi;
656. private DetailRamen(String nama\_menu, byte[] foto, String deskripsi) {
657. this.nama\_menu = nama\_menu;
658. this.foto = foto;
659. this.deskripsi = deskripsi;}
660. public DetailRamen(byte[] foto) {
661. this("", foto, "");}
662. private static List<DetailRamen> detailRamen() {
663. try (Connection connection = DB.sql2o.open()) {
664. final String query = "SELECT \* FROM `detail\_ramen`";
665. return connection.createQuery(query).executeAndFetch(DetailRamen.class);}}
666. public static DetailRamen detailRamen(String nama\_menu) {
667. return detailRamen()
668. .stream()
669. .filter(detailRamen -> detailRamen.getNama\_menu().equals(nama\_menu))
670. .findFirst()
671. .orElse(null);}
672. public static DetailRamen detailRamen(Menu menu) {
673. return detailRamen()
674. .stream()
675. .filter(detailRamen -> detailRamen.getNama\_menu().equals(menu.getNama\_menu()))
676. .findFirst()
677. .orElse(null);}
678. public boolean add() {
679. try (Connection connection = DB.sql2o.open()) {
680. final String query =
681. "INSERT INTO `detail\_ramen` (`nama\_menu`, `foto`, `deskripsi`) " +
682. "VALUES (:nama\_menu, :foto, :deskripsi)";
683. connection.createQuery(query).bind(this).executeUpdate();
684. return connection.getResult() > 0;}}
685. public boolean update() {
686. try (Connection connection = DB.sql2o.open()) {
687. final String query =
688. "UPDATE `detail\_ramen` SET `foto` = :foto, `deskripsi` = :deskripsi " +
689. "WHERE `nama\_menu` = :nama\_menu";
690. connection.createQuery(query).bind(this).executeUpdate();
691. return connection.getResult() > 0;}}
692. public boolean delete() {
693. try (Connection connection = DB.sql2o.open()) {
694. final String query = "DELETE FROM `detail\_ramen` WHERE `nama\_menu` = :nama\_menu";
695. connection.createQuery(query).bind(this).executeUpdate();
696. return connection.getResult() > 0;}}
697. @SuppressWarnings("WeakerAccess")
698. public String getNama\_menu() {
699. return nama\_menu;}
700. @SuppressWarnings("unused")
701. public byte[] getFoto() {
702. return foto;}
703. @SuppressWarnings("unused")
704. public String getDeskripsi() {
705. return deskripsi;}
706. public void setNama\_menu(String nama\_menu) {
707. this.nama\_menu = nama\_menu;}
708. public void setDeskripsi(String deskripsi) {
709. this.deskripsi = deskripsi;}
710. @Override
711. public String toString() {
712. return "DetailRamen{" +"nama\_menu='" + nama\_menu + '\'' +", foto=" +Arrays.toString(foto) +", deskripsi='" + deskripsi + '\'' +'}';}}
713. **Transaksi.java**
714. package com.unindra.restoserver.models;
715. import com.google.gson.annotations.Expose;
716. import com.jfoenix.controls.datamodels.treetable.RecursiveTreeObject;
717. import com.unindra.restoserver.DB;
718. import javafx.beans.property.SimpleStringProperty;
719. import javafx.beans.property.StringProperty;
720. import javafx.collections.FXCollections;
721. import javafx.collections.ObservableList;
722. import org.joda.time.\*;
723. import org.sql2o.Connection;
724. import java.util.Date;
725. import java.util.List;
726. import java.util.stream.Collectors;
727. import static com.unindra.restoserver.Rupiah.rupiah;
728. public class Transaksi extends RecursiveTreeObject<Transaksi> {
729. private String id\_transaksi;
730. private String no\_meja;
731. private Date tanggal;
732. @Expose
733. private static ObservableList<Transaksi> transaksiList = FXCollections.observableArrayList();
734. public Transaksi(String no\_meja) {
735. this.id\_transaksi = Id.getIdByDateTime(new LocalDateTime());
736. this.no\_meja = no\_meja;
737. this.tanggal = new Date();}
738. static {
739. Thread thread = new Thread(() -> {
740. while (!Thread.interrupted()) {
741. try {
742. updateTransaksi();
743. Thread.sleep(5000);
744. } catch (InterruptedException e) {
745. e.printStackTrace();}}});
746. thread.start();}
747. public void simpan() {
748. try (Connection connection = DB.sql2o.open()) {
749. final String query =
750. "INSERT INTO `transaksi` (`id\_transaksi`,`no\_meja`,`tanggal`) " +
751. "VALUES (:id\_transaksi,:no\_meja,:tanggal)";
752. connection.createQuery(query).bind(this).executeUpdate();
753. if (connection.getResult() > 0) {
754. List<Pesanan> pesanans = PesananService.getItems(this);
755. pesanans.forEach(pesanan -> pesanan.simpan(this));
756. pesanans.forEach(PesananService::delete);
757. TransaksiService.delete(this);}}}
758. private static void updateTransaksi() {
759. try (Connection connection = DB.sql2o.open()) {
760. final String query = "SELECT \* FROM `transaksi`";
761. transaksiList.setAll(connection.createQuery(query).executeAndFetch(Transaksi.class));}}
762. public static ObservableList<Transaksi> getTransaksiList() {
763. return transaksiList;}
764. public static List<Transaksi> getTransaksiList(LocalDate tanggal) {
765. return getTransaksiList()
766. .stream()
767. .filter(transaksi -> new LocalDate(transaksi.getTanggal()).equals(tanggal))
768. .collect(Collectors.toList());}
769. public static List<Transaksi> getTransaksiList(int tahun, int bulan) {
770. return getTransaksiList()
771. .stream()
772. .filter(transaksi -> {
773. LocalDate localDate = new LocalDate(transaksi.getTanggal());
774. return localDate.getYear() == tahun && localDate.getMonthOfYear() == bulan;})
775. .collect(Collectors.toList());}
776. public static Transaksi getTransaksi(Pesanan pesanan) {
777. return getTransaksiList()
778. .stream()
779. .filter(transaksi -> transaksi.getId\_transaksi().equals(pesanan.getId\_transaksi()))
780. .findFirst()
781. .orElse(null);}
782. public static int getTotalBayar(int tahun, int bulan) {
783. return getTransaksiList(tahun, bulan)
784. .stream()
785. .mapToInt(Transaksi::getTotalBayar)
786. .sum();}
787. public int getTotalBayar() {
788. return Pesanan.getPesanan(this).stream().mapToInt(Pesanan::getTotal).sum();}
789. public int getTotalBayarFromService() {
790. return PesananService.getPesananList().stream()
791. .filter(item -> item.getNo\_meja().equals(no\_meja))
792. .collect(Collectors.toList()).stream()
793. .mapToInt(Pesanan::getTotal)
794. .sum();}
795. String getId\_transaksi() {
796. return id\_transaksi;}
797. public String getNo\_meja() {
798. return no\_meja;}
799. public Date getTanggal() {
800. return tanggal;}
801. public StringProperty no\_mejaProperty() {
802. return new SimpleStringProperty(no\_meja);}
803. public StringProperty totalProperty() {
804. return new SimpleStringProperty(rupiah(getTotalBayarFromService()));}
805. public StringProperty pukulProperty() {
806. LocalTime t = new LocalTime(tanggal);
807. return new SimpleStringProperty(String.format("%d:%d WIB", t.getHourOfDay(), t.getMinuteOfHour()));}
808. @Override
809. public String toString() {
810. return "Transaksi{" +"id\_transaksi=" + id\_transaksi +", no\_meja='" + no\_meja + '\'' +", tanggal=" + tanggal +'}';}}
811. **User.java**
812. package com.unindra.restoserver.models;
813. import com.unindra.restoserver.DB;
814. import org.sql2o.Connection;
815. public class User {
816. private String username;
817. private String password;
818. public User(String username, String password) {
819. this.username = username;
820. this.password = password;}
821. public static User user(String username) {
822. try (Connection connection = DB.sql2o.open()) {
823. final String query = "SELECT \* FROM `user` WHERE `username` = :username";
824. return connection
825. .createQuery(query)
826. .addParameter("username", username)
827. .executeAndFetchFirst(User.class);}}
828. public String getUsername() {
829. return username;}
830. public String getPassword() {
831. return password;}
832. @Override
833. public String toString() {
834. return "User{" +
835. "username='" + username + '\'' +
836. ", password='" + password + '\'' +
837. '}';}}