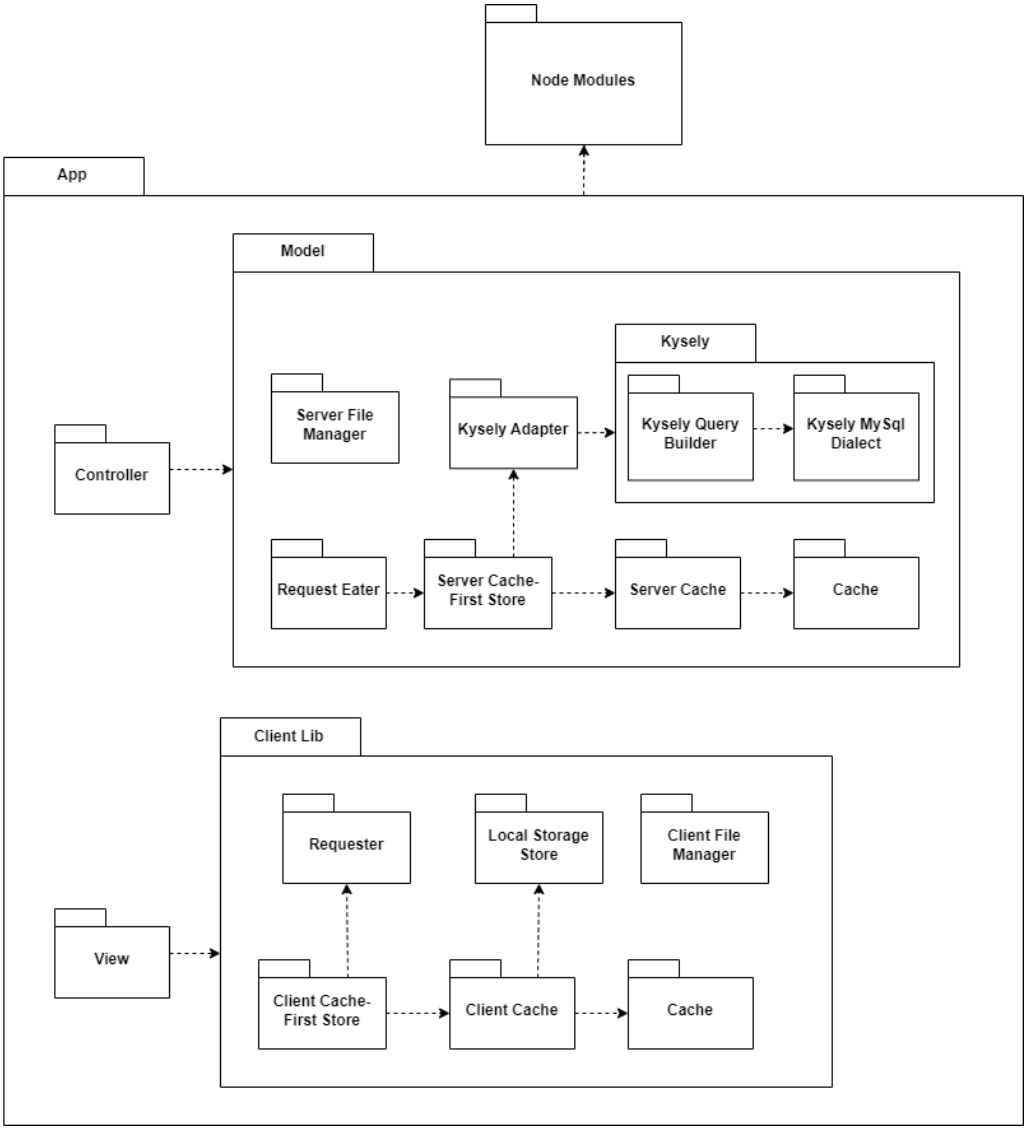
TUGAS DESAIN SISTEM

Nama : Adnan Rafiyansyah Majid

NPM : 140810200029

1. Package Diagram

Package diagram menggambarkan struktur sistem dengan menunjukkan berbagai komponen yang terdapat di dalamnya seperti class, atribut, metode, dan hubungan antar setiap objeknya.



Gambar 3.2 *Package Diagram*

Komponen-komponen yang termasuk dalam View membutuhkan package “Client Lib” komponen-komponen yang termasuk dalam Controller membutuhkan package “Model” dengan Client Lib dan Model merupakan kumpulan *library* yang memiliki fungsi dan dependensinya masing-masing. Sedangkan “Node Modules” berfungsi menyimpan *dependencies* yang dipasang menggunakan *package manager.*

2. Component Diagram

Component Diagram menjelaskan tentang *interface* yang diperlukan dan disediakan oleh komponen-komponen yang ada dan menjelaskan bagaimana komponen-komponen tersebut berinteraksi satu sama lain.

Dibuat sedemikian rupa untuk memudahkan perawatan dan pengujian juga meningkatkan skalabilitas dan modularitas aplikasi. Setiap komponen dapat diganti dengan komponen lain asalkan *interface* yang digunakan sesuai. Jika suatu komponen baru yang ingin digunakan memiliki *interface* berbeda maka dapat komponen *adapter* sebagai perantara untuk agar *interface*-nya sesuai.

a. *Interface* CRUD Request

Interface CRUD Request membutuhkan *body* dari HTTP *request* sesuai API yang disediakan sebagai *input* dan mengembalikan hasil operasi CRUD dan *status*.

b. *Interface* Storer CRUD

Interface Storer CRUD menyediakan 3 kelompok fungsi yaitu group, doc, dan docfile dan semuanya memiliki fungsi-fungsi berikut :

- insert, *input*-nya berupa data-data baru yang ingin ditambahkan dan *output*-nya berupa data-data yang berhasil ditambahkan.

- read, *input*-nya berupa *index* data-data yang ingin diambil dan *output*-nya berupa data-data yang berhasil diambil.
- update, *input*-nya berupa data-data yang ingin di-*update* dan *output*-nya berupa data-data yang berhasil di-*update*.
- delete, *input*-nya berupa *index* data-data yang ingin dihapus dan *output*-nya berupa data-data yang berhasil dihapus.

c. *Interface* File Storer

Interface File Storer menyediakan 3 kelompok fungsi yaitu group, doc, dan docfile dan semuanya memiliki fungsi-fungsi berikut :

- read, *input*-nya berupa nama *file* yang ingin dibaca dan *output*-nya berupa *file* yang berhasil dibaca.
- write, *input*-nya berupa nama *file* dan *file* yang ingin ditulis dan *output*-nya berupa *boolean* yang menandakan berhasil atau tidaknya proses penulisan *file*.
- delete, *input*-nya berupa nama *file* yang ingin dihapus dan *output*-nya berupa *boolean* yang menandakan berhasil atau tidaknya proses penghapusan *file*.

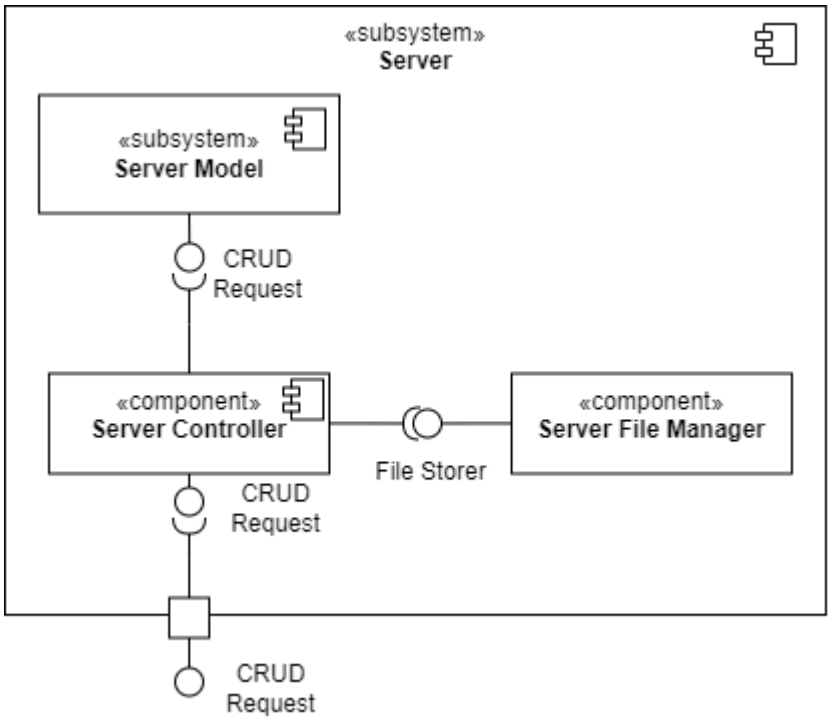
d. *Interface* Kysely CRUD

Interface Kysely CRUD merupakan *interface* yang disediakan oleh library Kysely berupa *query builder*.

e. *Interface* Kysely Database Dialect

Interface Kysely Database Dialect merupakan *interface* yang disediakan oleh library Kysely berupa *dialect* yang akan digunakan oleh *query builder* untuk berkomunikasi dengan *database* pilihan seperti MySQL.

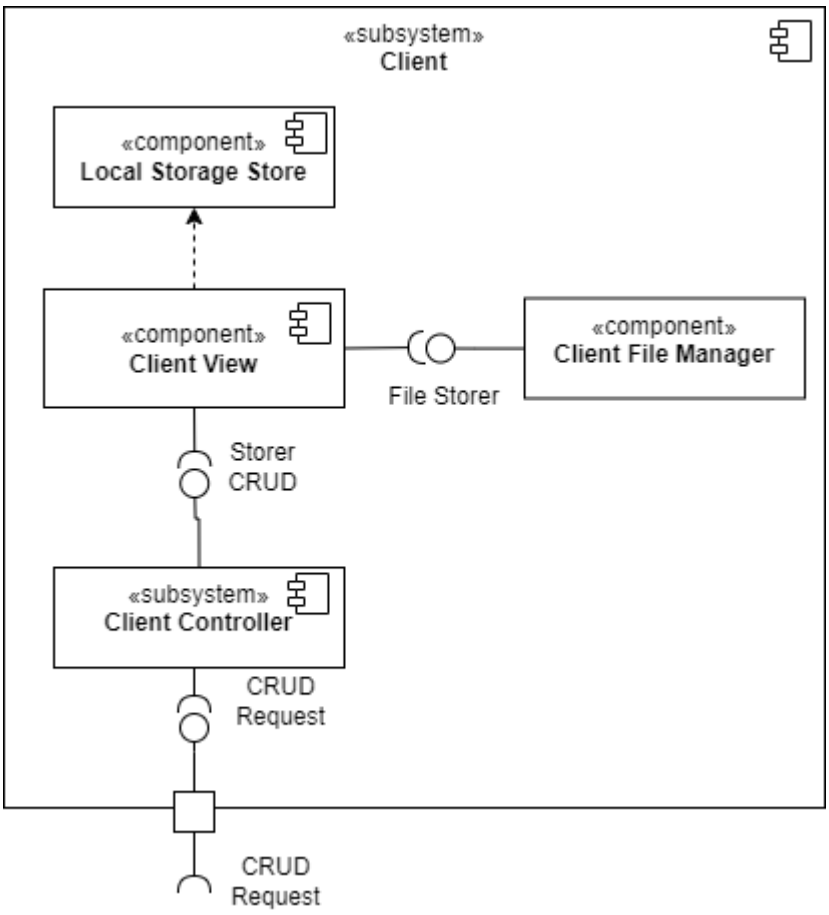
f. Komponen *Server*



Gambar 3.3 *Server Component Diagram*

Komponen ini terpakai hanya di server. *Request* yang masuk akan di-*handle* oleh komponen Server Controller lalu akan menggunakan komponen Server File Manager untuk operasi *file* atau Server Model untuk operasi CRUD lalu mengirim *response* ke klien.

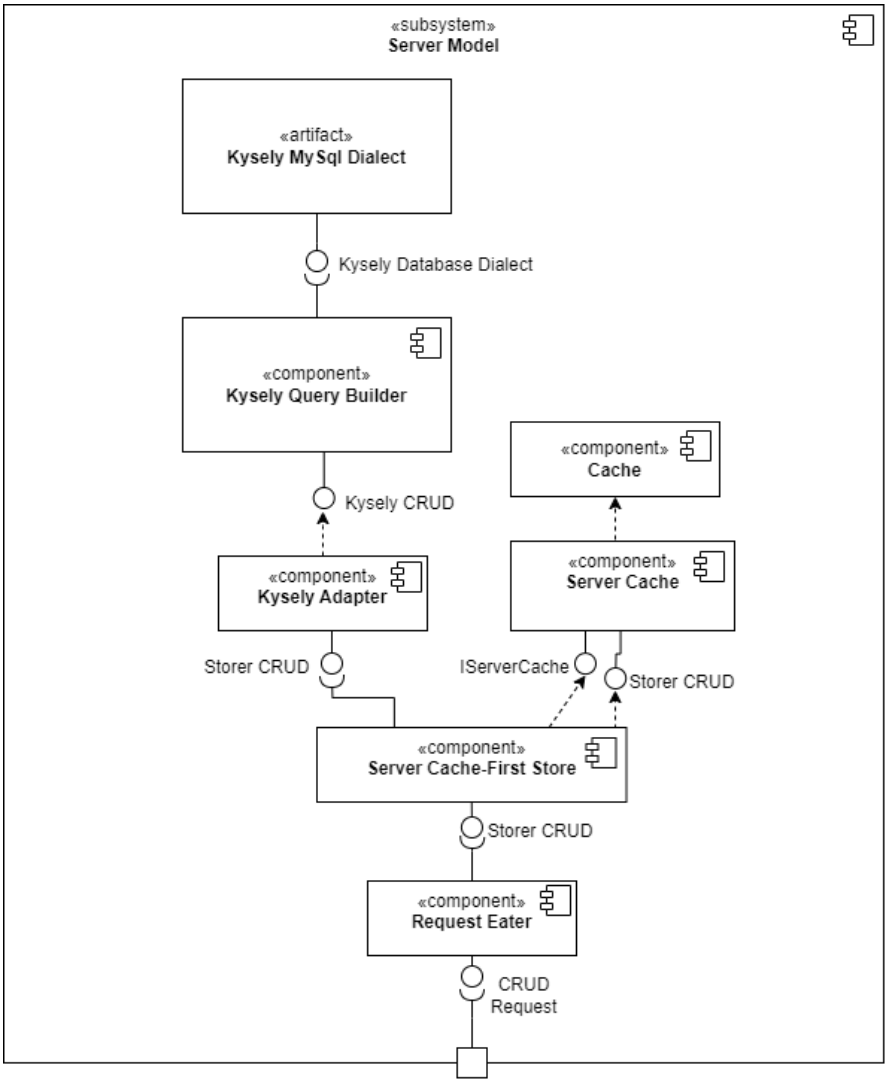
g. Komponen *Client*



Gambar 3.4 *Client Component Diagram*

Komponen ini terpakai hanya di *client* atau *browser*. Komponen Client View merupakan *view* yang digunakan oleh user dan akan menggunakan komponen Client Controller untuk mengirim *request* jika diperlukan. *Response* yang masuk akan di-*handle* oleh komponen Client Controller lalu disampaikan ke komponen Client View, kemudian akan disampaikan ke komponen Client File Manager jika diperlukan operasi *file* pada klien. Komponen Client View juga menggunakan Komponen Local Storage Store untuk menyimpan data-data *user interface* sebagai *browser cookies.*

h. Komponen *Server Model*



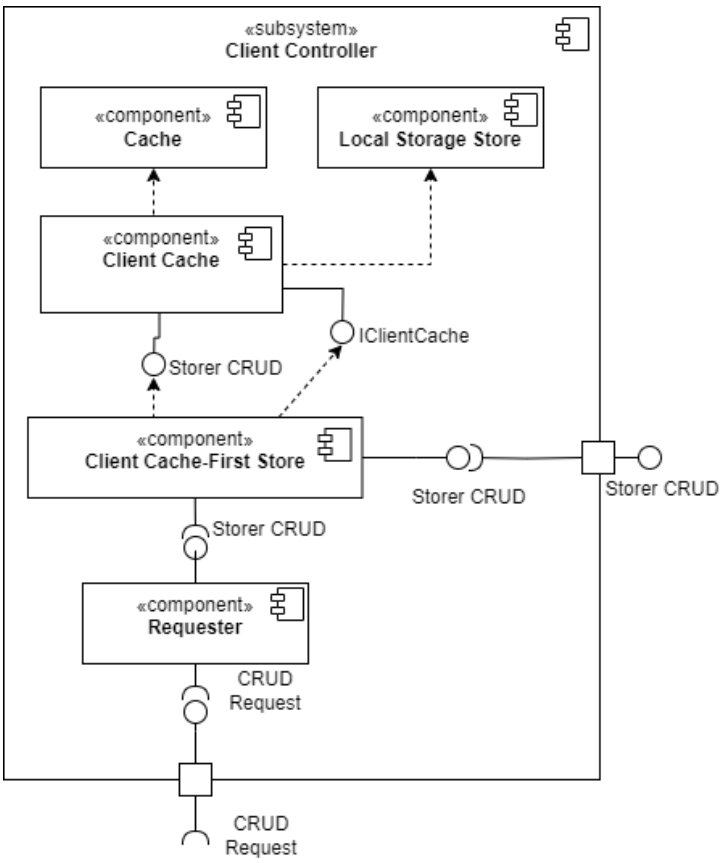
Gambar 3.5 *Server Model Component Diagram*

Komponen ini terpakai hanya di *server*. Komponen Request Eater akan menggunakan komponen Server Cache-First Store (SCFS) dengan *interface* Storer CRUD untuk menangani CRUD Request. SCFS akan menggunakan komponen Kysely Adapter untuk CRUD pada *database* dan komponen Server Cache untuk melakukan CRUD dan *refresh* pada *cache* yang tersimpan di *memory*. Komponen Kysely Adapter menggunakan Kysely Query Builder yang dipasangkan dengan *dialect* database MySQL untuk melakukan operasi pada *database.*

Aplikasi dapat menggunakan implementasi penyimpanan data apa saja asalkan menyediakan *interface* yang sesuai dengan *interface* yang digunakan oleh komponen Request Eater, atau implementasi penanganan request baru dengan *interface* CRUD Request, atau

menggunakan *database* lain dengan mengganti *dialect* yang digunakan oleh komponen Kysely Query Builder

i. Komponen *Client* Controller

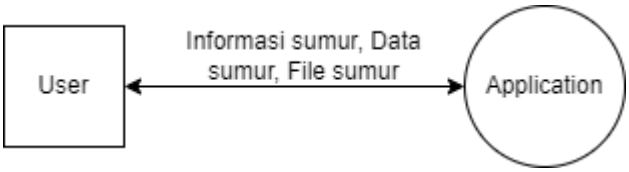


Gambar 3.6 *Client Controller Component Diagram*

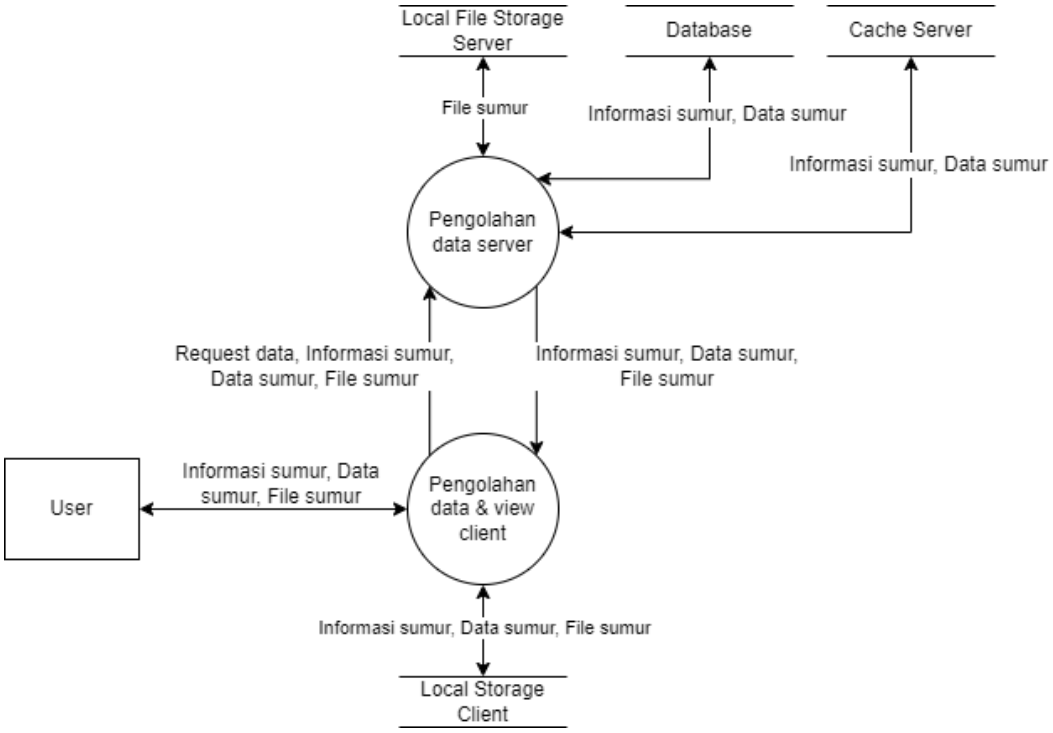
Komponen Client Cache-First Store (CCFS) digunakan oleh komponen Client View lalu menggunakan komponen Requester untuk mengirim *request* ke server dan mendapatkan *response*. CCFS menggunakan komponen Client Cache untuk melakukan refresh dan operasi CRUD pada *cache* klien yang tersimpan secara *persistent* di *browser cookies*. Kegunaan dari terdapatnya *cache* pada klien adalah agar tidak memberatkan *server* setiap kali ingin melihat data, hanya me-*request* ke *server* jika dibutuhkan.

3. *Data Flow Diagram*

Data Flow Diagram menjelaskan proses-proses dan aliran data pada sistem. DFD level 0 merupakan pengembangan dari diagram konteks, DFD level 1 merupakan pengembangan dari DFD level 0. Tiap proses dari DFD dapat dikembangkan lagi menjadi lebih detail sampai proses-proses tersebut tidak dapat dikembangkan lagi.



Gambar 3.7 DFD Level 0



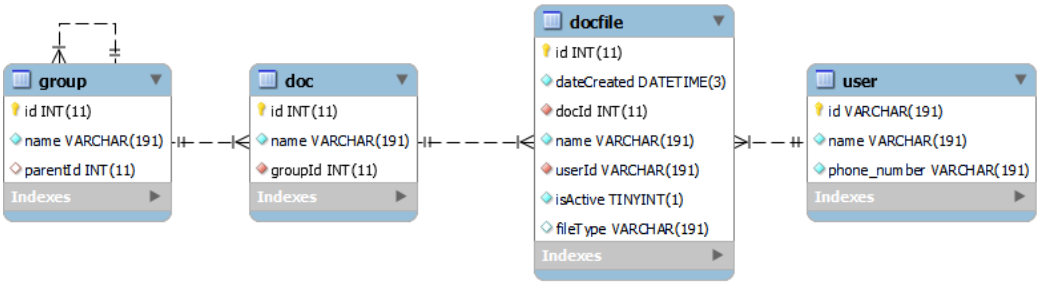
Gambar 3.8 DFD Level 1

Informasi dan file sumur yang yang di-*input* oleh pengguna akan diolah di klien dan akan terikirim request ke server untuk diolah dan disimpan disana lalu data akan dikembalikan ke klien dan akan diolah dan disimpan di klien. Pengguna mendapatkan

informasi dan file sumur dari klien yang mendapat data tersebut dari penyimpanan klien atau, jika tidak tersedia atau dibutuhkan, dari penyimpanan *server* dengan cara mengirim request ke *server*.

4. Rancangan *Database*

Perancangan *database* dilakukan dengan membuat tabel-tabel yang diperlukan. Tabel dibuat menggunakan *Data Base Management System* MySQL.



Gambar **Error! No text of specified style in document.**.9 Rancangan *Database*

Akan dibuat 4 tabel untuk fitur promosi yang dibuat yaitu: group, doc, docfile, dan user.

a. Tabel Group

Tabel Group berfungsi untuk menyimpan data grup yang merupakan *parent* dari doc atau sumur. Grup sendiri merupakan lokasi atau lapangan yang dapat digunakan untuk mengelompokkan sumur-sumur, suatu grup dapat memiliki banyak grup dan sumur karena ada bisa ada beberapa lapangan dalam suatu lokasi. Tabel Group harus memiliki satu row yang memiliki id 0 untuk digunakan sebagai root.

Tabel **Error! No text of specified style in document.**.1 Tabel Group

Atribut	**Tipe**	**Ukuran**	**Keterangan**
id	int	11	*Primary key* untuk identifikasi unik sebuah grup

name	varchar	191	Nama lokasi atau lapangan
parentId	int	11	*Foreign key* sebagai kode unik penghubung ke grup *parent.* Jika grup tersebut dihapus maka *row* akan ikut terhapus

b. Tabel Doc

Tabel Doc berfungsi untuk menyimpan data doc yang merupakan *parent* dari docfile. Doc sendiri merupakan sebuah sumur yang dapat memiliki banyak docfile atau data sumur.

Tabel **Error! No text of specified style in document.**.2 Tabel Doc

Atribut	**Tipe**	**Ukuran**	**Keterangan**
id	int	11	*Primary key* untuk identifikasi unik sebuah doc atau sumur
name	varchar	191	Nama sumur
groupId	int	11	*Foreign key* sebagai kode unik penghubung ke grup *parent.* Memiliki default value 0. Jika grup tersebut dihapus maka *row* akan ikut terhapus

c. Tabel Docfile

Tabel Docfile berfungsi untuk menyimpan data docfile yang merupakan data dari sebuah sumur dan masing-masing dapat menyimpan satu *file* sumur.

Tabel **Error! No text of specified style in document.**.3 Tabel Docfile

Atribut	**Tipe**	**Ukuran**	**Keterangan**
id	int	11	*Primary key* untuk identifikasi unik sebuah docfile dan *refrence*

			untuk *file* yang bersangkutan
name	varchar	191	Nama informasi
docId	int	11	*Foreign key* sebagai kode unik penghubung ke sumur pemilik informasi. Jika sumur atau doc tersebut dihapus maka *row* akan terhapus juga.
dateCreated	datetime		Berisi waktu kapan terakhir kali *file* di diperbarui.
userId	varchar	191	*Foreign key* berbentuk uuid sebagai kode unik penghubung ke *user* pengunggah informasi
isActive	boolean		Penanda apakah informasi aktif atau tidak.
fileType	varchar	191	Mime type dari *file* sumur. Null jika *file* tidak tersedia.

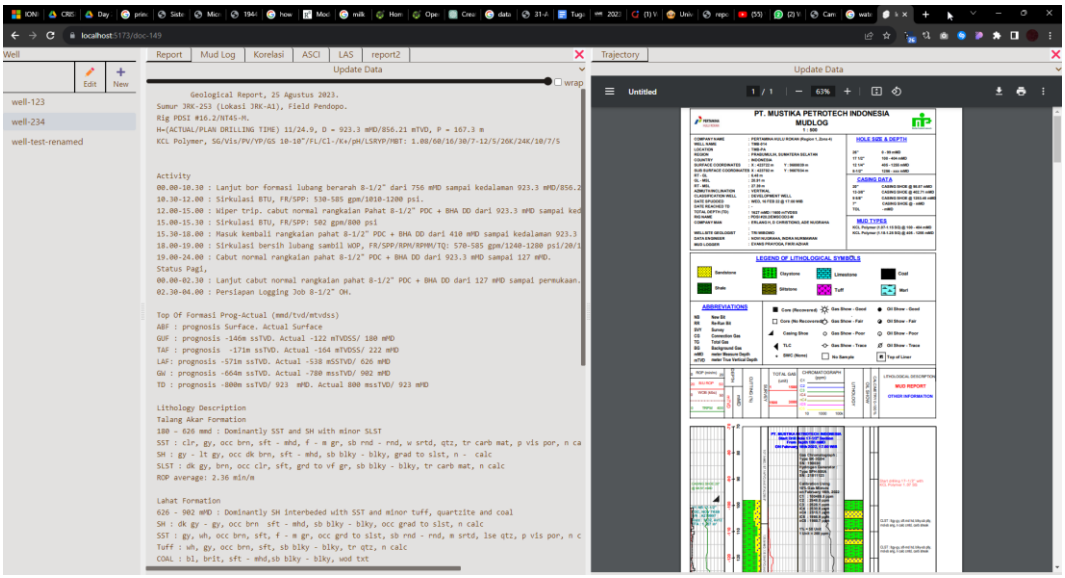
d. Tabel User

Tabel User digunakan untuk menyimpan data pembuat atau pengunggah data atau *file* sumur. Untuk implementasi oleh penulis, hanya akan ada satu user dengan id 1 dikarenakan batasan proyek ini.

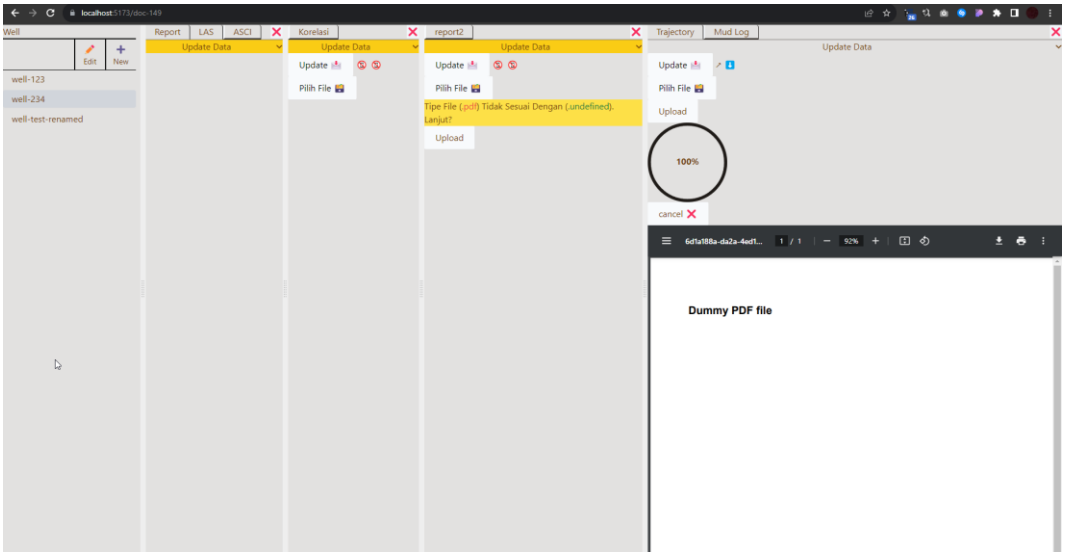
Tabel **Error! No text of specified style in document.**.4 Tabel User

Atribut	**Tipe**	**Ukuran**	**Keterangan**
id	varchar	191	*Primary key* berbentuk UUID sebagai identifikasi unik seorang *user*
name	varchar	191	Nama pengguna
phone_number	varchar	191	Nomor telpon pengguna

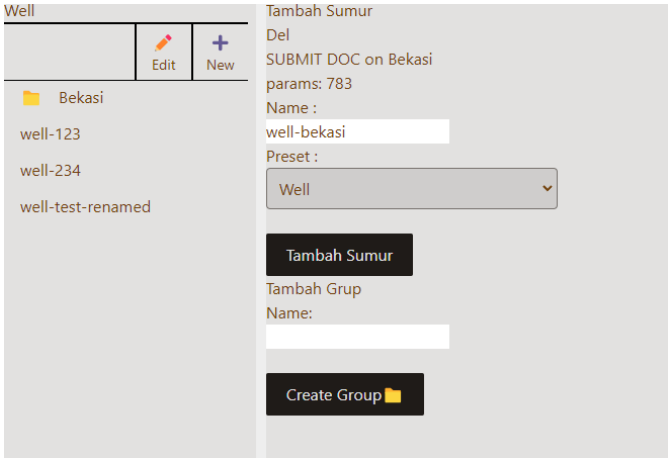
5. Desain UI



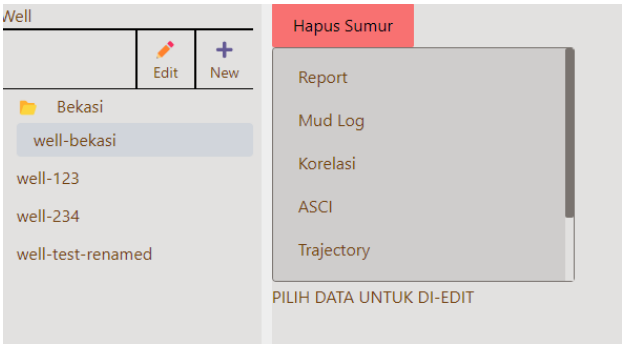
Tampilan data sumur (teks dan pdf)



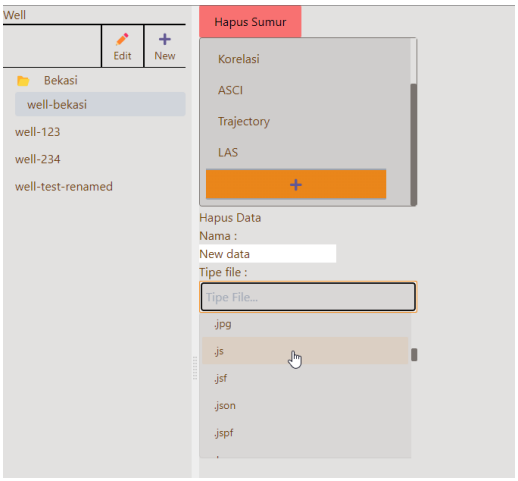
Tampilan upload file sumur



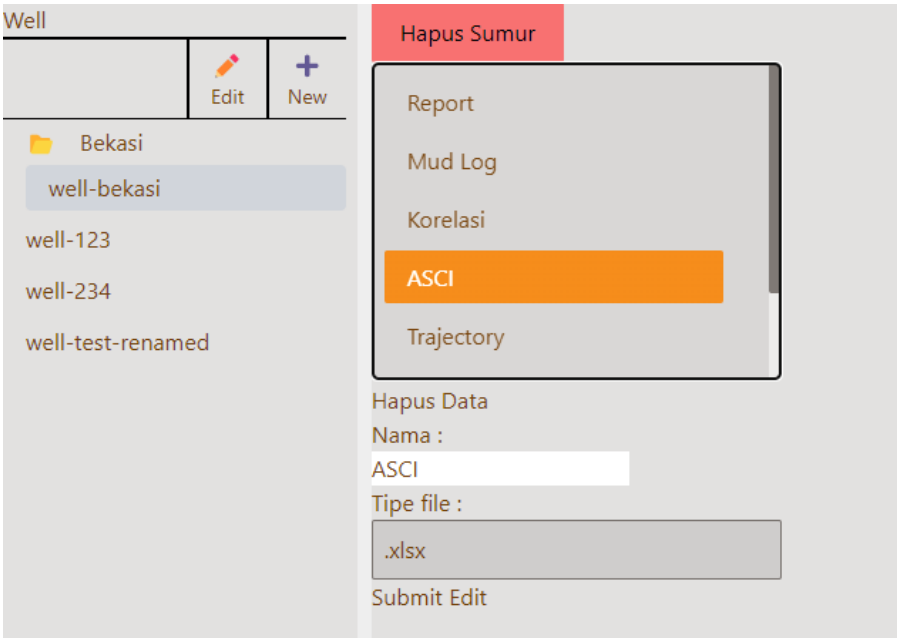
Tampilan tambah sumur atau grup



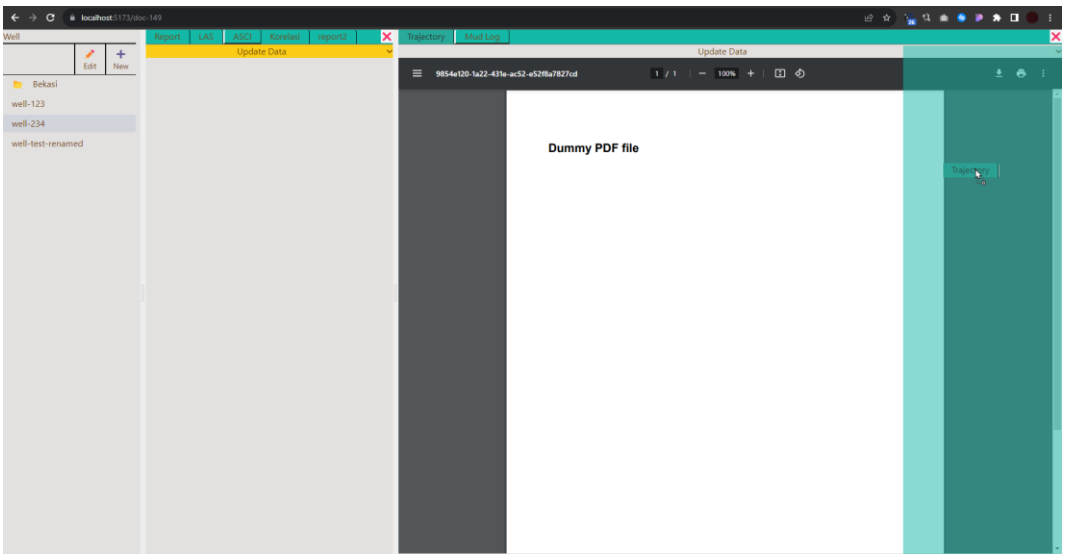
Tampilan edit sumur



Tampilan tambah data sumur



Tampilan edit data sumur



Tampilan mengubah layout tab data sumur