Pembangunan Sistem Pelaporan *Surveyor* Lapangan untuk Kementerian Kelautan & Perikanan Republik Indonesia

Agung Prio Rismawan[1], Eduard Rusdianto[2], Joseph Eric Samodra[3]
Program Studi Informatika, Fakultas Teknologi Industri, Universitas Atma Jaya Yogyakarta Jl. Babarsari No. 43, Janti, Caturtunggal, Kec Depok, Kabupaten Sleman, 55281, Daerah Istimewa Yogyakarta, Indonesia
Email: [1]1160709001@students.uajy.ac.id, [2]eduard.rusdianto@uajy.ac.id, [3]eric.samodra@uajy.ac.id

Abstrak. *Sektor kelautan di Indonesia memiliki potensi yang besar karena sumber kelautan yang banyak dan beragam. Salah satu cara untuk menjaga dan mengawasi potensi alam tersebut adalah dengan melakukan kegiatan pelaporan dalam ruang lingkup Kementerian Kelautan dan Perikanan. Tapi dalam praktiknya, banyak sekali oknum yang melakukan kecurangan saat survei. Kecurangan tersebut di antaranya adalah memberikan laporan palsu, tidak benar-benar survei di lapangan dan data-data pelaporan dimanipulasi sehingga data pelaporan tidak akurat.*
Aplikasi pelaporan dibuat dengan platform mobile dan menggunakan arsitektur BLoC (Business Logic Component). Arsitektur BLoC digunakan karena mudah dipahami dan powerful. Arsitektur BLoC membagi setiap komponen bagian menjadi modul yang kecil-kecil dan tidak bergantung satu-sama lain (low coupling). Arsitektur BLoC juga dapat memisahkan antara komponen logic dan presentation layer. Pemisahan ini akan memudahkan proses maintenance kode. Misalnya, pada saat suatu implementasi kode diubah, maka tidak diperlukan banyak perubahan pada kode yang lainnya.

Kata Kunci: *Pelaporan, Surveyor, Mobile, Realtime, Arsitektur BLoC.*

1. Pendahuluan

Kementerian Kelautan dan Perikanan selalu melakukan survei. Survei tersebut bertujuan sebagai pertimbangan pemerintah untuk membangun atau memperbaiki fasilitas yang sudah ada di dalam ruang lingkup Kementerian Kelautan, seperti pelaporan kondisi mercusuar, pelaporan hasil perikanan, pelaporan tambak ikan nelayan, pengecekan TEWS (*Tsunami Early Warning System*), dan lain-lain. Untuk mempermudah melakukan survei tersebut, pihak Kementerian Kelautan dan Perikanan sudah memiliki tim yang bertugas untuk melakukan survei yang disebut *surveyor* [1]. Para *surveyor* ini nantinya akan melakukan pelaporan terkait kondisi laut, perikanan, sarana, dan prasarana yang ada di lapangan. Tapi dalam praktiknya, banyak sekali oknum yang melakukan kecurangan saat survei dan pelaporan di lapangan. Kecurangan tersebut di antaranya adalah memberikan laporan palsu dan data-data yang dilaporkan tidak benar. Tujuan dari penelitian ini yaitu mengurangi kecurangan pada saat survei, menyajikan data pelaporan *surveyor* yang akan diserahkan kepada Kementerian Kelautan dan Perikanan [2].

2. Tinjauan Pustaka

Paper yang digunakan penulis merupakan paper dari para peneliti yang sudah meneliti mengenai metode aplikasi *mobile* dan Sistem Informasi Geografis. Penelitian pertama dilakukan oleh Mambu, dkk. Mambu membuat aplikasi pelaporan Manado *Smart City* untuk Pemerintah Kota Manado. Penelitian ini bertujuan untuk memudahkan partisipasi masyarakat dalam pelaporan Manado *Smart City*. Pengembangan *e-report* pada Manado *Smart City* dibangun dengan menggunakan teknologi Ionic dengan bantuan Cordova. Penelitian ini menghasilkan aplikasi *mobile* yang mendukung kegiatan pelaporan dalam Manado *Smart City* [3].

Penelitian kedua dilakukan oleh Haloho dan Pribadi. Penelitian ini berjudul Perancangan Aplikasi Berbasis Android untuk Survei Kondisi Kapal oleh Owner *Surveyor*.

Penelitian ini dibangun menggunakan bahasa pemrograman Java dengan tools Android Studio. Aplikasi ini mempunyai kelebihan dapat menyajikan laporan secara otomatis ketika user telah melakukan pengisian form. Pengisian data dilakukan dengan cara survei secara berkala untuk pengecekan kondisi kapal sekaligus pendataan kapal di lapangan. Tujuan perawatan kapal adalah menjamin terlaksananya sistem pemeliharaan terencana PMS (Planned Maintenance System) [4].

Penelitian berikutnya dilakukan oleh Hati, dkk. yang berjudul Aplikasi Penanda Lokasi Peta Digital Berbasis *Mobile* GIS Pada Smartphone Android. Penelitian ini melibatkan penggunaan GIS (Geographic Information System) yang digunakan untuk menandakan sebuah tempat atau lokasi melalui aplikasi *mobile* Android. Hasil dari penelitian ini adalah aplikasi *mobile* dengan fitur untuk menandakan lokasi dengan menggunakan GIS serta beberapa fitur utama seperti *input* data, menampilkan data yang tersimpan, dan menampilkan rute pada peta [5].

Penelitian yang terakhir dilakukan oleh Somya. Penelitian ini berjudul Sistem Monitoring Kendaraan Secara *realtime* Berbasis Android menggunakan teknologi CouchDB di PT. Pura Barutama. Penelitian ini digunakan untuk monitoring kendaraan secara *realtime* menggunakan teknologi CouchDB dan GPS tracker pada studi kasus PT. Pura Barutama. CouchDB adalah salah satu contoh dari RTDBS (*Realtime* Database System) yang dikembangkan oleh Apache. Penelitian ini bertujuan untuk mendeteksi keberadaan kendaraan secara *realtime*. Pada penelitian ini, GPS berfungsi sebagai pendeteksi keberadaan kendaraan. Kendaraan yang sudah dipasang GPS apabila berpindah tanpa diketahui oleh pemiliknya maka akan otomatis mengirimkan notifikasi melalui aplikasi Android [6].

3. Metodologi Penelitian

Metode penelitian yang dilakukan oleh penulis adalah sebagai berikut.

3.1. Wawancara

Wawancara adalah tahap pertama yang dilakukan sebelum pembuatan aplikasi dilakukan. Tahap wawancara digunakan untuk mengumpulkan informasi yang dibutuhkan untuk membangun sebuah aplikasi. Metode wawancara dipercaya sangat efektif untuk mencari informasi dari pengguna aplikasi dikarenakan kita dapat melakukan interaksi secara langsung dan narasumber juga dapat memberikan informasi secara cepat dan jelas.

3.2. Merancang Aplikasi

Setelah mendapatkan validasi dari pengguna, tahap selanjutnya adalah melakukan perancangan aplikasi. Perancangan tersebut dilakukan berdasarkan dokumen SKPL dan DPPL yang sudah divalidasi oleh pengguna pada tahap sebelumnya. Pada tahap ini pengembang akan melakukan perancangan antarmuka, pembuatan ERD, arsitektur yang digunakan, dan lain-lain.

3.3. Pengkodean Aplikasi

Tahap selanjutnya adalah pengkodean. Pada tahap ini pengembang akan melakukan implementasi aplikasi terhadap kebutuhan yang sudah dirancang sebelumnya. Hasil dari pengkodean ini adalah aplikasi jadi yang berdasarkan dokumen SKPL dan DPPL.

3.4. Pengujian Aplikasi

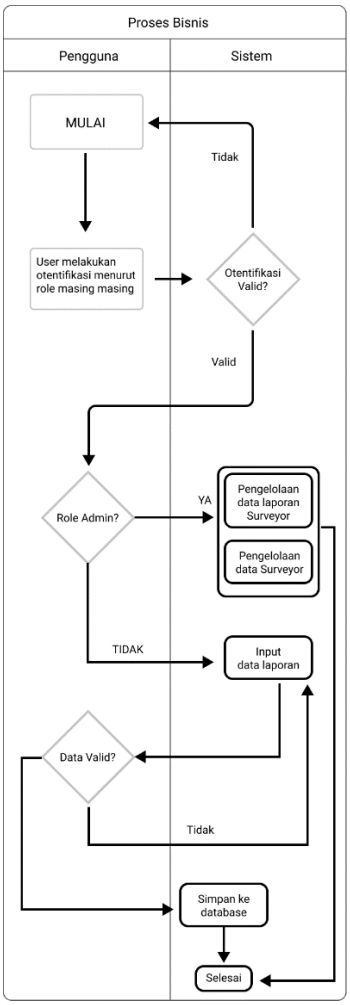
Tahap selanjutnya adalah pengujian. Pengujian dilakukan melalui beberapa tahap, yaitu pengujian oleh pengguna dan pengujian fungsionalitas aplikasi. Pengujian fungsionalitas digunakan untuk menguji aplikasi secara fungsionalitas, sedangkan pengujian pengguna digunakan untuk menguji usability dari aplikasi yang sudah dibuat. Dari pengujian ini apabila terdapat kesalahan maka akan segera diperbaiki secara fungsionalitas maupun *usability*.

3.5. Laporan pembuatan aplikasi.

Tahap terakhir adalah pembuatan laporan. Pada tahap ini mencakup beberapa hal yaitu latar belakang masalah, tinjauan pustaka, perancangan aplikasi, pengujian aplikasi, dan lain-lain. Pada laporan juga akan dilampirkan hasil aplikasi yang sudah dikembangkan oleh pengembang.

4. Hasil dan Diskusi

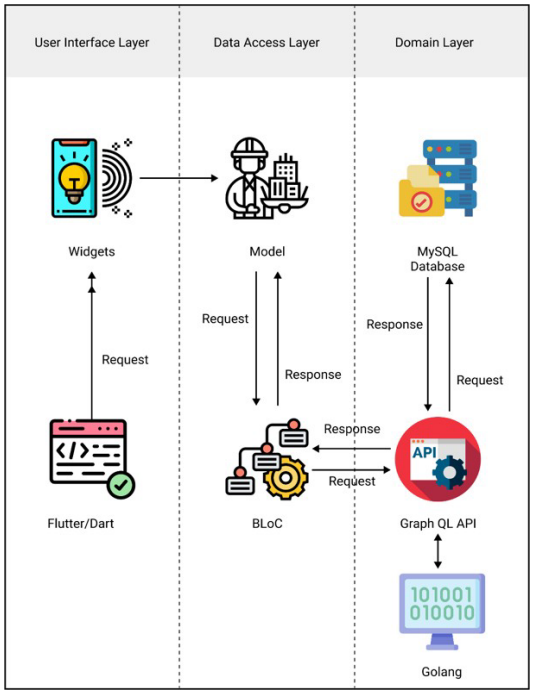
4.1. Analisis Sistem



Gambar 1. Diagram Alir Proses Bisnis

Pada Gambar 1 menunjukkan proses bisnis MOTS-PeR. Terdapat dua aktor pada diagram alir pada Gambar 1, yakni pengguna dan sistem. Proses bisnis yang digambarkan adalah proses pengadaan laporan dimulai dari pengumpulan data. Pada diagram Gambar 1, pengguna dapat mengirimkan data laporan ke dalam aplikasi, sistem kemudian akan memvalidasi data yang masuk untuk memastikan data yang masuk adalah data yang valid dan lengkap. Jika data dinyatakan valid, maka sistem akan menyimpan data tersebut. Untuk pengguna yang tidak memasukkan laporan ke dalam sistem (admin) dapat melakukan pengelolaan *surveyor* dan melihat seluruh laporan yang masuk oleh *surveyor*.

4.2. Arsitektur Sistem



Gambar 2. Arsitektur Sistem MOTS-PeR

Rancangan arsitektur MOTS-PeR dapat dilihat pada Gambar 2. Terdapat tiga komponen utama dari MOTS-PeR, yakni MOTS API (back-end), BLoC (Business Logic Component), dan Widget. MOTS API merupakan pusat pengolahan data yang bertanggung jawab pada manajemen data-data utama, pengelolaan data *surveyor*, dan pengelolaan data laporan.

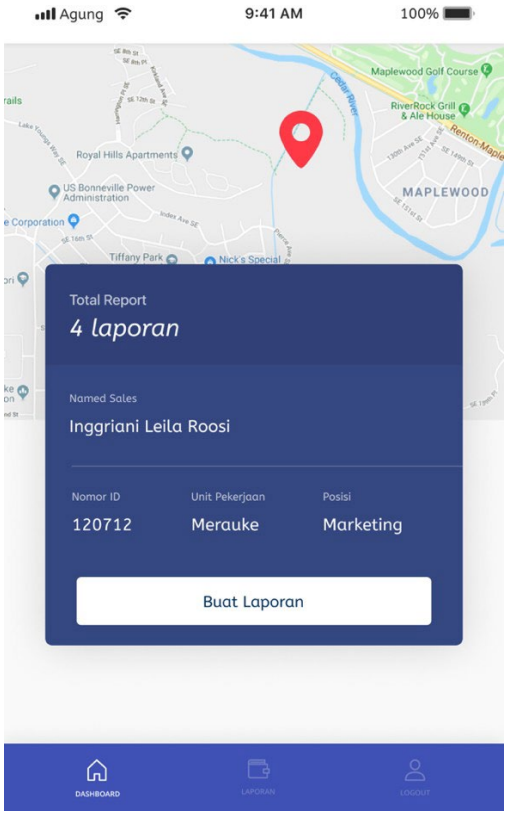
MOTS API dibangun dengan arsitektur MVC (*Model View Controller*) menjadi Graphql API (Query Language Application Programming Interface). Basis data MOTS API menggunakan MariaDB 10.3. MOTS-PeR *Mobile* dibangun dengan arsitektur BLoC (Business Logic Component). MOTS-PeR *Mobile* hanya berperan sebagai antarmuka grafis untuk mengakses Graphql API. Melalui MOTS-PeR *Mobile* pengguna dapat mengoperasikan manajemen data utama, manajemen data laporan, dan manajemen data pengguna.

Komunikasi antara tiga komponen ini menggunakan protokol yang sama, yakni HTTP/HTTPS (HyperText Transfer Protocol). Dalam komunikasi ini, MOTS API akan berperan sebagai server yang akan menunggu *request* dari MOTS-PeR *Mobile*. *Request* ini memiliki data spesifik yang berisi tujuan dari *request* tersebut, berdasarkan *request* ini MOTS API akan memberikan *response* yang berisi data sesuai *request* yang datang. Format data yang digunakan adalah JSON (JavaScript Object Notation).

Dalam operasi internal MOTS API yang memiliki arsitektur MVC, *request* yang masuk akan diterima oleh controller untuk kemudian diekstraksi datanya. Data ini kemudian diberikan ke *service layer* tertentu sesuai konteks *request*. *Service layer* akan mengolah data dari *request* dan model untuk kemudian menghasilkan keluaran data yang sesuai. Model adalah komponen yang bertugas untuk melakukan interaksi dengan basis data (*querying*). Pemisahan tugas dalam arsitektur MVC ini akan memudahkan proses pengembangan dan memastikan tiap lapisan arsitektur ini dapat diuji dengan lebih teliti.

4.3. Implementasi Sistem

4.3.1. Tampil data *Surveyor*



Gambar 3. Halaman Dashboard *Surveyor*

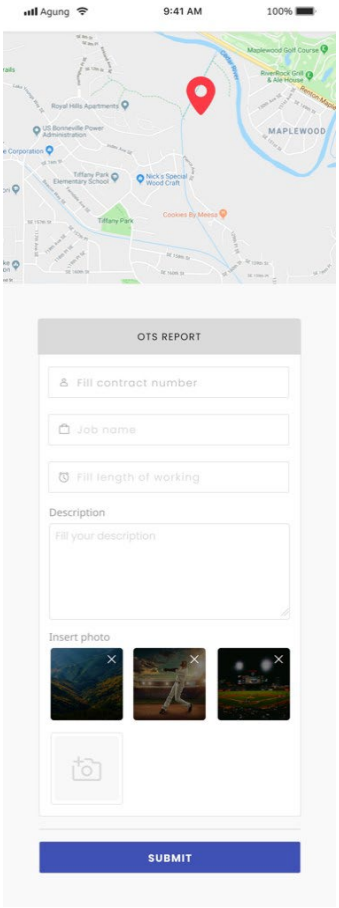
Pada Gambar 3 halaman *dashboard client*, akan dilakukan API Call pada *endpoint whoami*. *Endpoint* ini akan mengembalikan data *user* dan data personal seperti nama, id, unit pekerjaan, posisi, dan data report-nya. Pada peta, tampilan marker pada peta di-update secara *realtime*. *Update* ini dilakukan pada userOverviewBloc dengan memanggil fungsi `_handleGetCurrentLocation()` seperti pada Gambar 4.

```
_handleGetCurrentLocation() async {
  Geolocator _geolocator = Geolocator()..forceAndroidLocationManager;
  await _geolocator.getCurrentPosition(desiredAccuracy: LocationAccuracy.best)
    .then((Position position) async {
      final coordinates = new Coordinates(position.latitude, position.longitude);
      final addresses = await Geocoder.local.findAddressesFromCoordinates(coordinates);
      final cityName = addresses.first.locality;

      final location = new model.Location(new LatLng(position.latitude, position.longitude), cityName);

      dispatch(UserOverviewLocationFetchDone(location: location));
    });
}
```

Gambar 4. Potongan Kode Untuk Menampilkan Data Secara *Realtime*

Pada Gambar 4 dijelaskan proses update lokasi disesuaikan dengan *latitude* dan *longitude surveyor* yang *login* tersebut. Pada potongan kode Gambar 4, geolocator aman mengambil posisi dengan fungsi `getCurrentPosition(desiredAccuracy:LocationAccuracy.best)` dengan mengambil posisi terbaik dari *surveyor* yang sedang *login*.

4.3.2. Halaman Buat Laporan



Gambar 5. Halaman Membuat Laporan Baru

Pada halaman pembuatan laporan baru, menggunakan *plugin* yang disediakan oleh Flutter untuk menampilkan Google Maps. *Plugin* tersebut adalah *geolocator*. Pada halaman pembuatan laporan pertama kali dilakukan pemanggilan pada fungsi *geolocator* untuk mendapatkan data lokasi terkini berupa koordinat *latitude* dan *longitude* menggunakan kode pada Gambar 5.

```
_geolocator.getCurrentPosition(desiredAccuracy: LocationAccuracy.best)
  .then((Position position) async {
    final coordinates = new Coordinates(position.latitude, position.longitude);
    final addresses = await Geocoder.local.findAddressesFromCoordinates(coordinates);
    final cityName = addresses.first.locality;

    _currentLocationCtrl.sink.add(new Location(LatLng(position.latitude, position.longitude), cityName));
  });
```

Gambar 6. Potongan Kode Geolocator

Pada Gambar 6, *geolocator* akan mengambil posisi terkini menggunakan fungsi `getCurrentPosition(desiredAccuracy: LocationAccuracy.best)`. Fungsi ini akan mengambil lokasi terbaik dari *surveyor* yang sedang *login*. Lokasi terbaik yang dimaksud adalah ketepatan akurasi pengambilan data *latitude* dan *longitude*, yaitu hingga tujuh angka di belakang koma.

```
bool _isFormValid(ReportForm form, List<File> files) {
  if (form.contractNumber.isEmpty || form.description.isEmpty || form.jobName.isEmpty) {
    this.dispatch(ReportStoringError(error: Exception('Data tidak boleh kosong')));
    return false;
  }

  if (form.duration <= 0) {
    this.dispatch(ReportStoringError(error: Exception('Durasi tidak valid')));
    return false;
  }

  if (files.length == 0) {
    this.dispatch(ImagesUploadingError(error: Exception('Gambar tidak boleh kosong')));
    return false;
  }

  return true;
}
```

Gambar 7. Kode Pengecekan `_isFormValid()` Pada Halaman *Input* Laporan

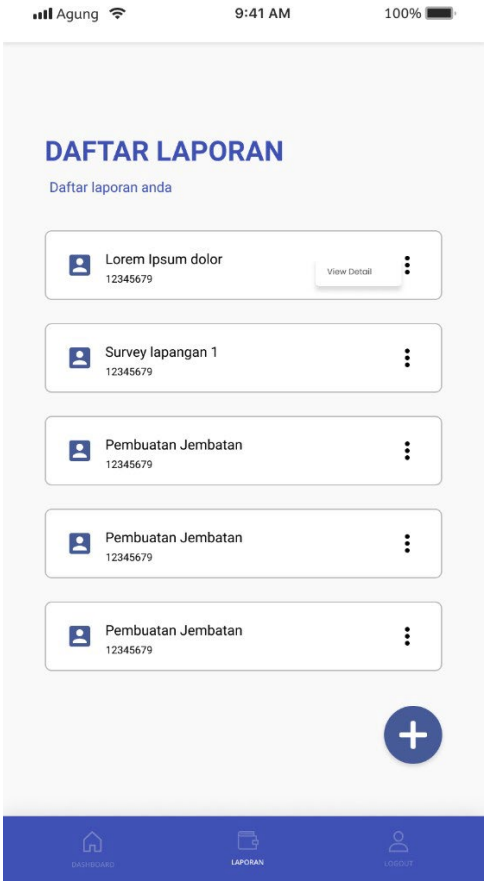
Pada potongan kode pada Gambar 7, akan dilakukan pengecekan apabila data dari form yang dimasukan valid, maka proses submit report akan dilanjutkan dengan melakukan mutasi pada *endpoint createReport*. Pada potongan kode pada Gambar 7, terdapat contoh pengecekan form durasi yang tidak boleh berupa teks, kurang dari nol, dan gambar tidak boleh kosong.

```
Future<ReportForm> _handleImagesUpload(ReportForm form, List<File> files) async {
  try {
    List<Future<Image>> imagesUploadFutures = files
      .map<Future<Image>>((item) {
        return _fileService.upload(item);
      })
      .toList();
    List<Image> images = await Future.wait(imagesUploadFutures)..toList();

    dispatch(ImagesUploadingSuccess());
    return form.copyWith(images: images);
  } catch (error) {
    dispatch(ImagesUploadingError(error: error));
    throw error;
  }
}
```

Gambar 8. Kode Untuk Upload Gambar

Pada Gambar 8 dijelaskan bahwa *upload* gambar dilakukan satu persatu dengan memanggil fungsi `_handleImagesUpload()`. Fungsi `_handleImagesUpload()` memiliki dua buah parameter, yaitu data laporan dan gambar. Data gambar diberikan tipe array list, dengan harapan gambar bisa dimasukkan lebih dari satu yang ditujukan pada endpoint FileService.

4.3.3. Tampil Daftar Laporan *Client*



Gambar 9. Daftar Laporan *Surveyor*

Pada Gambar 9 akan dilakukan API Call pada endpoint getMyReports dan akan memberikan kembalian berupa data laporan *surveyor* yang *login*. Data kumpulan laporan ini yang kemudian dimasukkan ke dalam ListView.

```
child: ListView.builder(
  itemCount: state.reports.length,
  itemBuilder: (context, index) => reportCard(context, state.reports[index]),
), // ListView.builder
```

Gambar 10. Kode Untuk Menampilkan ListView

Pada potongan kode seperti Gambar 10, setiap laporan *surveyor* akan ditampilkan menggunakan ListView yang berisi nama laporan dan nomor kontraknya. Pada halaman ini apabila *surveyor* melakukan klik pada salah satu ListView, maka data laporan pada ListView tersebut akan diarahkan pada halaman detail laporan.

5. Kesimpulan dan Saran

Berdasarkan penelitian yang dilakukan penulis dengan menganalisis aplikasi MOTS-PeR berdasarkan teori-teori yang digunakan dalam penelitian, maka dapat ditarik kesimpulan dari tugas akhir ini antara lain: MOTS-PeR dapat menyediakan fitur untuk melakukan pelaporan *surveyor* secara digital dan dapat melacak data lokasi laporan secara *realtime* berdasarkan lokasi *surveyor*. Lokasi tersebut akan ditampilkan dalam bentuk peta digital pada aplikasi. Dengan adanya fitur ini lokasi laporan *surveyor* dapat dilacak secara *realtime* sehingga mengurangi kecurangan *surveyor* apabila melakukan pelaporan tidak sesuai dengan

lokasi yang sudah ditentukan. Dari hasil proses analisis, perancangan, implementasi, hingga pengujian pada penelitian ini, didapatkan saran pengembangan lebih lanjut untuk MOTS-PeR yaitu kemampuan untuk *cache* data pelaporan dan informasi user pada aplikasi MOTS-PeR *surveyor*. Dengan adanya fitur ini ketika kondisi *device surveyor offline* maka data-data *surveyor* masih bisa diakses dan data pelaporan *surveyor* akan disimpan pada penyimpanan lokal.

Referensi

[1] Y. A. Nugroho, "Evaluasi Reaksi Peserta Pada Penyelenggaraan Diklat Diklat Aparatur Kementerian Kelautan Dan Perikanan Evaluation of Participant' S Reactions on the Implementation of Basic Training of Functional Position of Fishery' S Extension Workers in the Training ," vol. 13, no. 1, pp. 49–60, Jun, 2018.

[2] I. M. H. Antara, I. G. M. Darmawiguna, and I. M. A. Pradnyana, "Pengembangan Aplikasi Mobile Crowdsourcing Informasi Layanan Umum (Studi Kasus di Kabupaten Buleleng)," Kumpul. Artik. Mhs. Pendidik. Tek. Inform., vol. 8, no. 2, p. 154, 2019, doi: 10.23887/karmapati.v8i2.18362.

[3] O. E. Mambu, Y. D. Y. Rindengan, and S. D. S. Karouw, "Pengembangan Aplikasi E-Report Layanan Masyarakat untuk Manado Smart City," J. Tek. Inform., vol. 8, no. 1, 2016, doi: 10.35793/jti.8.1.2016.12233.

[4] P. S. Haloho and T. Wuruk Pribadi, "Perancangan Aplikasi Komputer Berbasis Android untuk Survei Kondisi Kapal oleh Owner Surveyor," J. Tek. ITS, vol. 5, no. 2, 2017, doi: 10.12962/j23373539.v5i2.20921.

[5] G. Hati, A. Suprayogi, and B. Sasmito, "Aplikasi Penanda Lokasi Peta Digital Berbasis Mobile Gis Pada Smartphone Android," J. Geod. Undip, vol. 2, no. 4, p. 82406, 2013.

[6] R. Somya, "Sistem Monitoring Kendaraan Secara Real Time Berbasis Android menggunakan Teknologi CouchDB di PT. Pura Barutama," J. Nas. Teknol. dan Sist. Inf., vol. 4, no. 2, pp. 53– 60, 2018, doi: 10.25077/teknosi.v4i2.2018.53-60.