Slides progetto LAM 2023

Bumbyroads

Erik Koci

Università di Bologna

June 13, 2023

Contents

1. Introduzione
2. Scelte Progettuali
3. Divisione e struttura del codice
4. Analisi delle principali feature implementate
5. Screenshot interfaccia grafica

Introduzione

Con BumpyRoads, è possibile segnalare i difetti stradali in modo rapido ed efficiente. L'applicazione permette di rilevare e registrare le imperfezioni sulle strade utilizzando un dispositivo mobile. Basta avviare l'app e attivare la geolocalizzazione.
Durante il tragitto, attraverso una semplice interfaccia essa permetterà di registrare le informazioni sulle buche o altre irregolarità che incontri lungo il percorso.

Scelte Progettuali

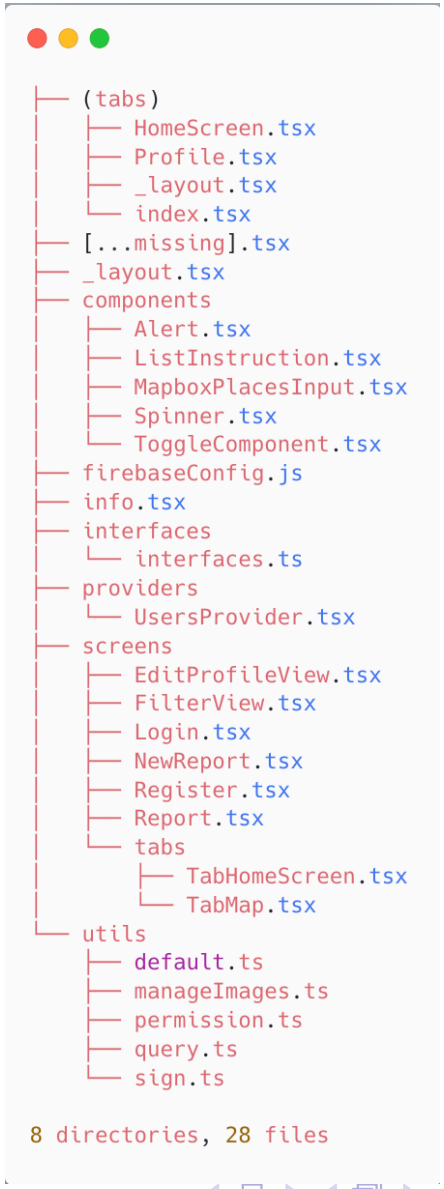
Sono state fatte diverse scelte progettuali nello sviluppo dell'applicazione le principali sono:

- Uso di react native
- Utilizzo di NativeBase
- Integrazione di Mapbox
- Backend con Firebase
- Uso di Typescript
- Strutturata in Tab

Divisione e struttura del codice

Divisione in cartelle, rispettivamente:

- Tabs
- Components
- Interfaces
- Providers
- Screens
- Utils



Analisi delle principali feature implementate

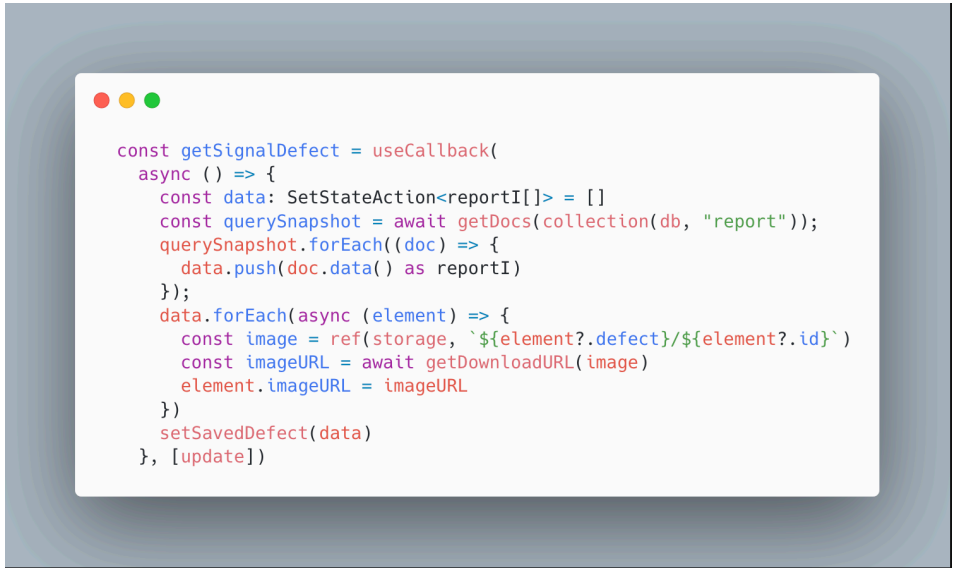
```
<NavigationContainer independent={true} linking={config}
  fallback={<ActivityIndicator color="blue" size="large" />}>
  <Tab.Navigator screenOptions={({ route }) => ({
    tabBarIcon: ({ color }) => screenOptions(route, color),
  })}>
    {user ?
      <>
        <Tab.Screen name='Maps' component={TabMap} options={{ headerShown: false }}></Tab.Screen>
        <Tab.Screen name='NewReport' component={NewReport} options={{ headerShown: true }}>
        </Tab.Screen>
        <Tab.Screen name='Profile' component={Profile}></Tab.Screen>
      </>
      :
      <>
        <Tab.Screen name='Home' component={TabHomeScreen}></Tab.Screen>
      </>
    }
  </Tab.Navigator>
</NavigationContainer>
```

Figure: Navigazione in tab

```
export const getUserLocation = async (setLocation: React.Dispatch<React.SetStateAction<positionI>>,
setError: React.Dispatch<React.SetStateAction<string>>) => {
  let { status } = await Location.requestForegroundPermissionsAsync();
  if (status !== 'granted') {
    setError('Permission to access location was denied')
    return
  }
  const location = await Location.getCurrentPositionAsync({ accuracy: Location.Accuracy.Highest });
  const values: positionI = location.coords
  setLocation(values);
}
```

Figure: Geolocalizzazione

Analisi delle principali feature implementate



```
const getSignalDefect = useCallback(
  async () => {
    const data: SetStateAction<reportI[]> = []
    const querySnapshot = await getDocs(collection(db, "report"));
    querySnapshot.forEach((doc) => {
      data.push(doc.data() as reportI)
    });
    data.forEach(async (element) => {
      const image = ref(storage, `${element?.defect}/${element?.id}`)
      const imageURL = await getDownloadURL(image)
      element.imageURL = imageURL
    })
    setSavedDefect(data)
  }, [update])
```

Figure: Mapbox

```
export const getUserLocation = async (setLocation: React.Dispatch<React.SetStateAction<positionI>>,
setError: React.Dispatch<React.SetStateAction<string>>) => {
  let { status } = await Location.requestForegroundPermissionsAsync();
  if (status !== 'granted') {
    setError('Permission to access location was denied')
    return
  }
  const location = await Location.getCurrentPositionAsync({ accuracy: Location.Accuracy.Highest });
  const values: positionI = location.coords
  setLocation(values);
}
```

Figure: Ottieni segnalazioni

Analisi delle principali feature implementate

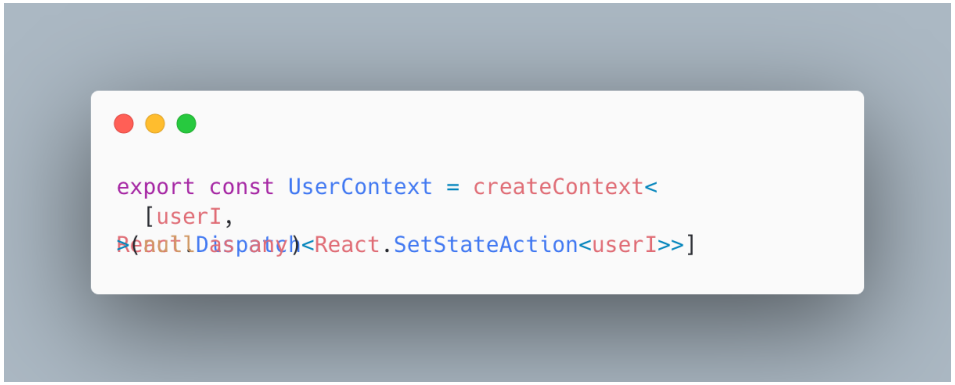
```
const sendSignalDefect = async () => {
  if ((!image || !defect || !severity || !locationUser))
    return
  try {
    const address: AddressI = (await reverseGeocodeAsync(locationUser))[0] as AddressI
    let docRef = await addDoc(collection(db, "report"), {
      defect: defect,
      severity: severity,
      userUid: user.uid,
      createdAt: new Date().toString(),
      position: locationUser,
      like: [],
      address: address,
      isResolved: false
    });
    const imageURL = await uploadImage(image, `${defect}/${docRef.id}`) // upload
image setImage('')
    await setDoc(docRef, { id: docRef.id, imageURL: imageURL }, { merge: true })
    setSeverity('')
    setDefect('')
    props.navigation.navigate("Profile")
  } catch (e) {
    console.error("Error adding document: ", e);
  }
}
```

Figure: Salva segnalazioni

Analisi delle principali feature implementate

```
  const vibrateOnNearReport = () => {
    savedDefect.forEach(report => {
      if ((Math.abs(report.position.longitude - location.longitude) < 0.00008)
        && (Math.abs(report.position.latitude - location.latitude) < 0.000008))
{       Vibration.vibrate()
      }
    });
  }
```

Figure: Vibrazione



```
export const UserContext = createContext<
  [userI,
React.Dispatch<React.SetStateAction<userI>>]
```

Figure: User context

Analisi delle principali feature implementate

```
const fetchRoute = async () => {
  const avoidPoints: positionI[] = [{ longitude: 0, latitude: 0 }]
  Object.entries(avoid).map(([key, value]) => {
    if (value) {
      savedDefect.map(defect => {
        if (defect.defect == key) {
          avoidPoints.push({ longitude: defect.position.longitude, latitude: defect.position.latitude
          })
        }
      })
    }
  })
  const points = avoidPoints?.map(location => `point(${location.longitude} ${location.latitude})`);
  const reqOptions = {
    waypoints: [
      { coordinates: origin },
      { coordinates: destination },
    ],
    profile: vehicle,
    geometries: 'geojson',
    exclude: points.toString()
  };
  const res = await directionsClient.getDirections(reqOptions).send();
  const newRoute = makeLineString(res.body.routes[0].geometry.coordinates);
  setRoute(newRoute);
}
```

Figure: Ottieni percorso

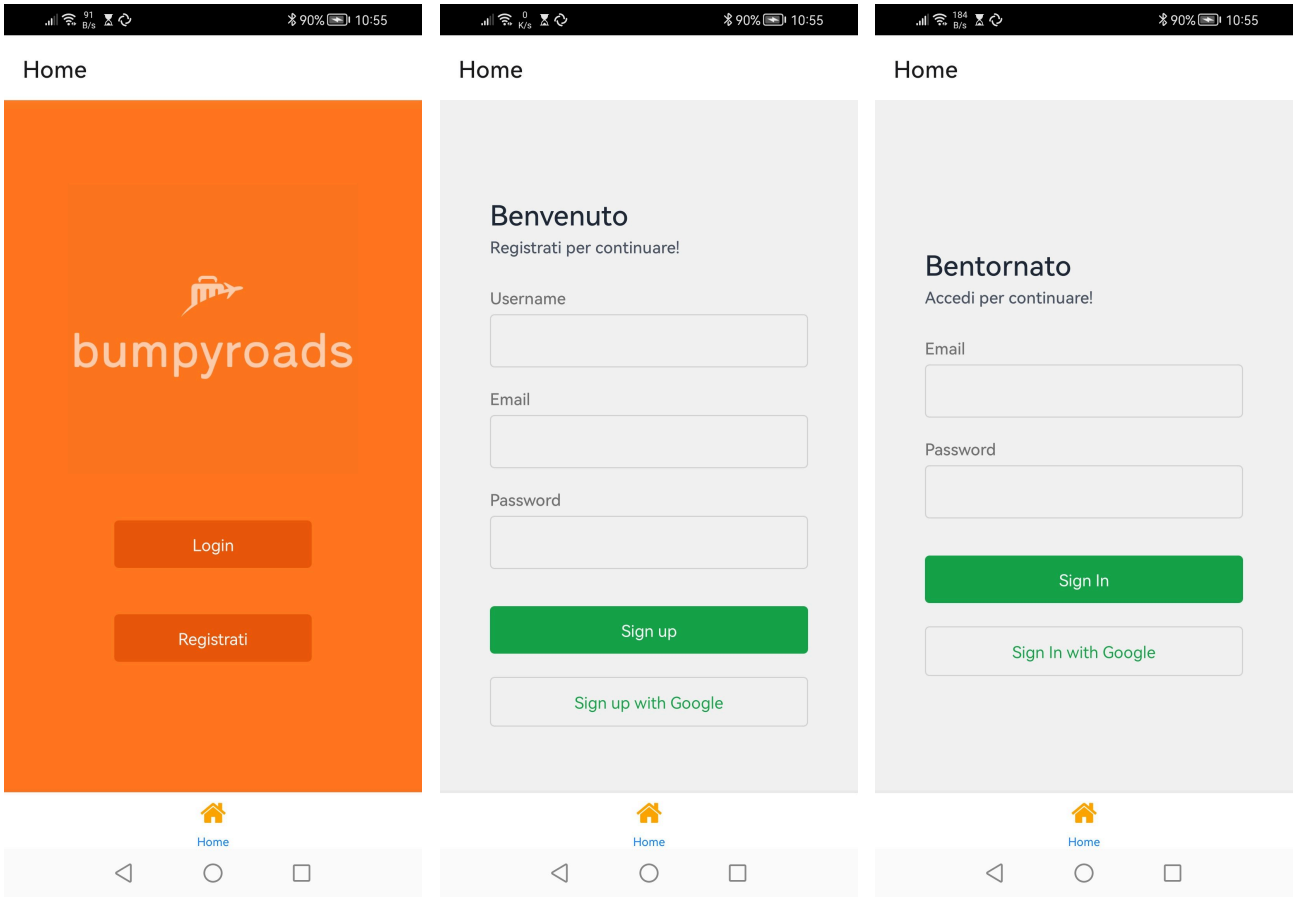
Analisi delle principali feature implementate

```
  const userLike = async (isLikedPost: boolean) => {
    try {
      let newLikes: string[] = reportData?.like!
      if (isLikedPost && !isLike) {
        setIsLike(true)
        setTotalLike(totalLike + 1)
        newLikes?.push(user.uid!)
      } else {
        if (newLikes.includes(user.uid!) && isLike) {
          setIsLike(false)
          newLikes = newLikes.filter(report => report !==
user.uid!)setTotalLike(totalLike - 1)
        }
      }
      const uniqueLikes = [...new Set(newLikes)];
      await updateDoc(doc(db, "report", reportData?.id!), {
        like: uniqueLikes
      });
    } catch (error) {
      console.log(error)
    }
  }
```

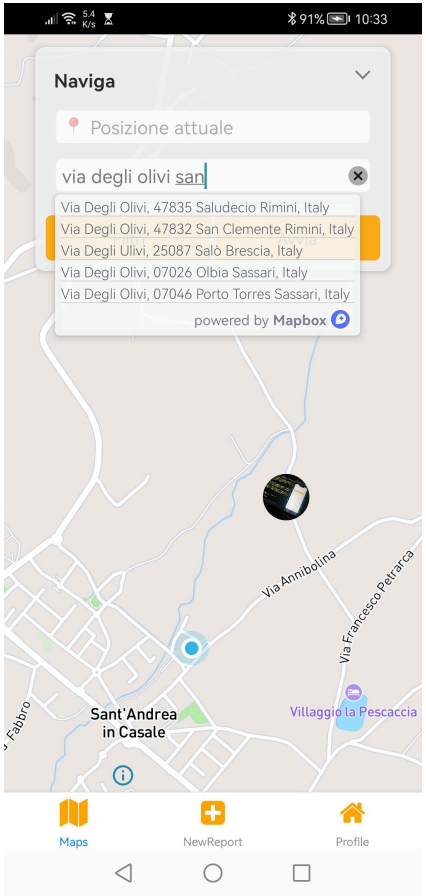
Figure: Gestione like

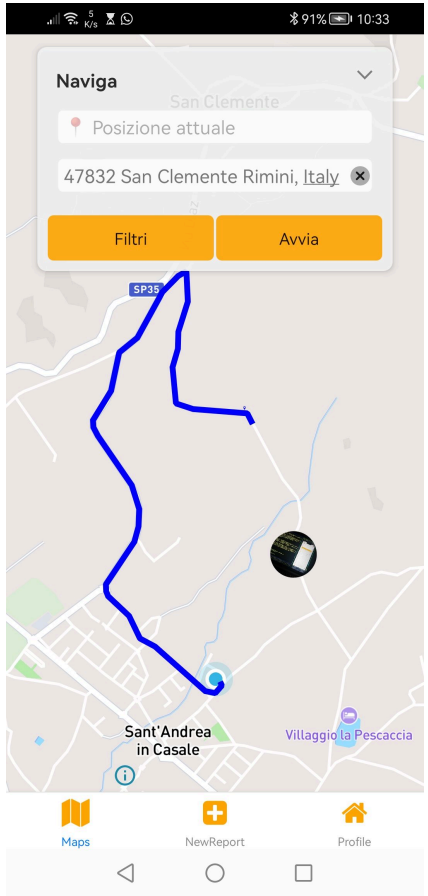
Screenshot interfaccia grafica

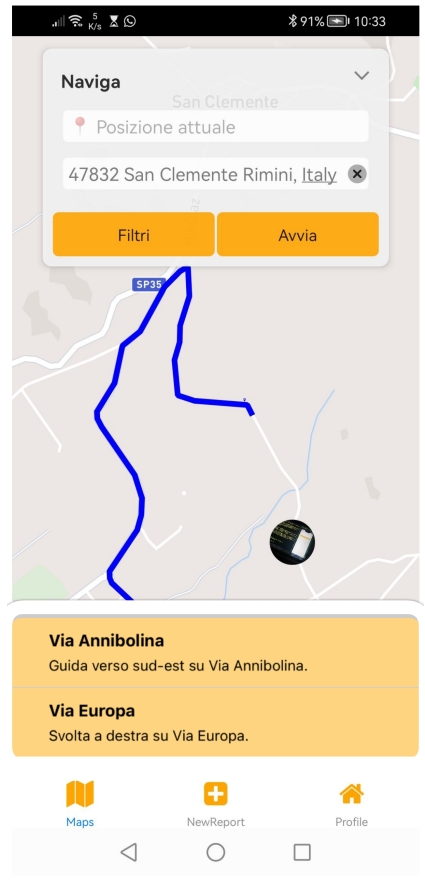
L'interfaccia grafica appare gradevole e semplice da utilizzare, grazie all'utilizzo della libreria nativeBase



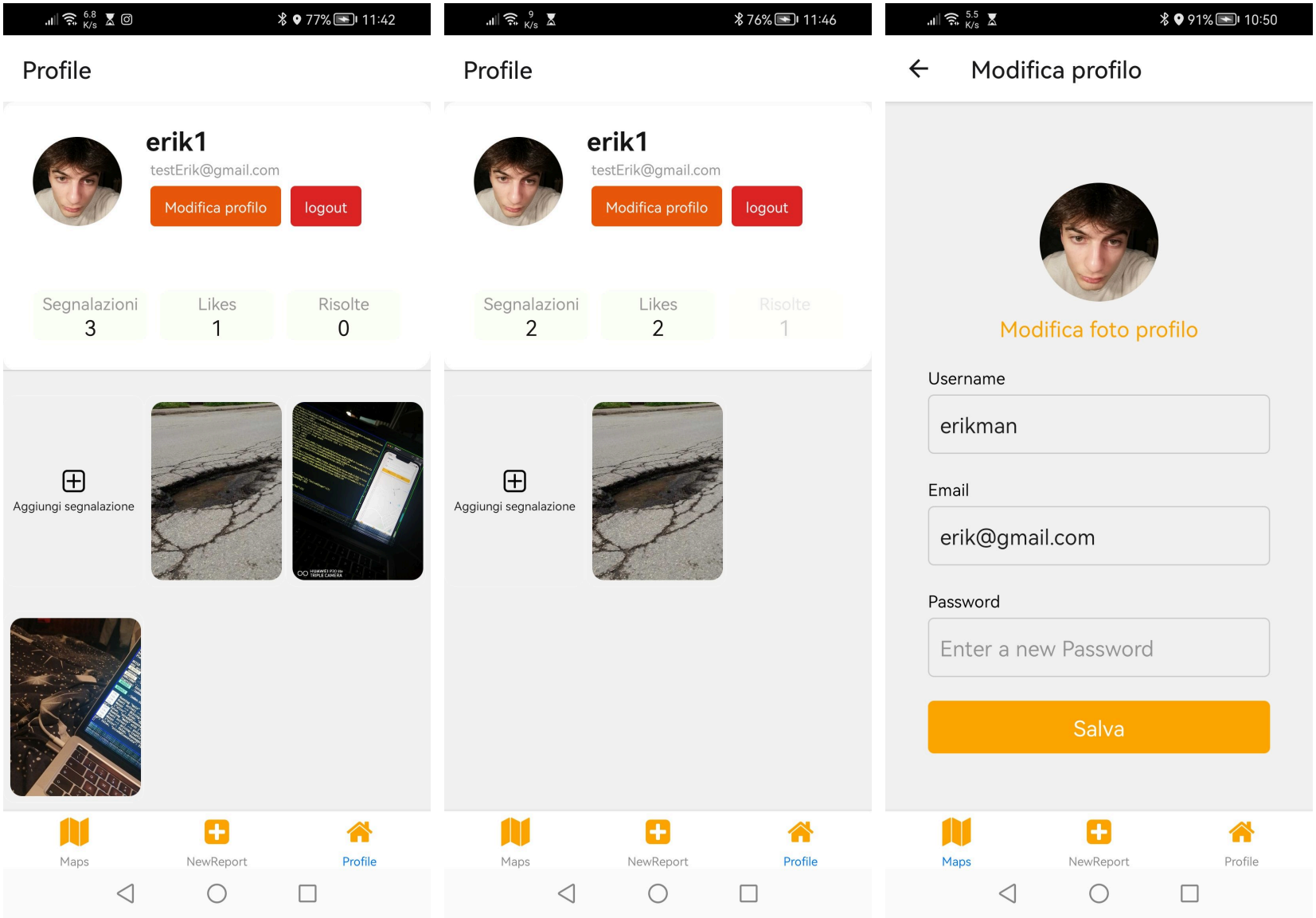
Screenshot interfaccia grafica



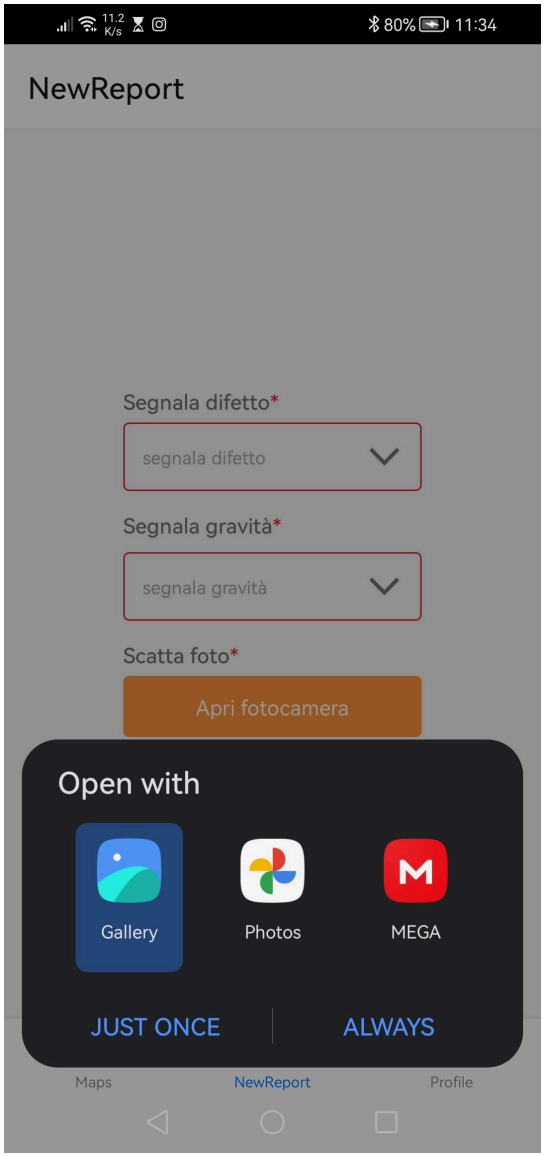




Screenshot interfaccia grafica



Screenshot interfaccia grafica



Screenshot interfaccia grafica