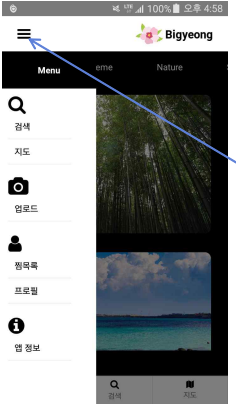
Bigyeong mobile

React native (expo)

https://expo.dev/@shineinjin/bigyeong

첫 화면



Stack Navigation
- Draw Navigation
  - Bottom Tab Navigation
    - Top Tab Navigation 구조

menu에 Draw Navigation 삽입

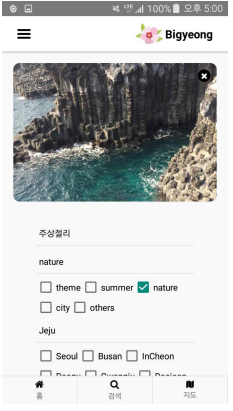
유저 인증

```
const generateSecureKey = async () => {
  // 키 값 가져오기
  const key = await SecureStore.getItemAsync("key")

  if (!key) {
    try {
      let random = Random.getRandomBytes(10).join("")
      await SecureStore.setItemAsync("key", random)
      // firebase에 유저 네로 key값으로 유저 생성
      await firestore.collection("users").doc(random).set({
        uploads: [],
        likes: [],
        created: moment().format(),
      })
    } catch (error) {
      // key값은 생성됐지만 락으로 인해 firestore유저가 생성 되지 못할
      // 수 있으므로 도중에 에러가 생기면 key값 제거
      console.log(error)
      await SecureStore.deleteItemAsync("key")
      alert(
        "key값을 생성하는데 실패하였습니다 앱을 다시 시작하여 주시기 바랍니다"
      )
    }
  }
```

firebase의 유저 인증을 사용하지 않고
expo의 SecureStore Key를 사용해
앱을 시작하면 key 생성 후 사용

업로드



체크 박스 직접 구현

ImagePicker 사용

카메라 기능은 폰 기종문제로 일단 제외

업로드 코드

```
try {
  //비어있는 칸 확인
  if (imageUri === null) return alert("사진을 첨부해주시기 바랍니다")
  if (title.trim() === "") return alert("제목을 적어주시기 바랍니다")
  else if (category === "")
    return alert("카테고리를 선택해 주시기 바랍니다")
  else if (region === "") return alert("지역을 선택해 주시기 바랍니다")
  else if (location.trim() === "")
    return alert("위치를 적어 주시기 바랍니다")

  scrollRef.current.scrollTo({ y: 0, animated: true }) // 스크롤 맨 위로

  setLoading(true) // 업로드 시작

  //image picker uri를 firebase storage uri로 변환
  // timeSet은 시간초과 시 이벤트 발생
  let timeSet
  timeSet = setTimeout(() => {
    setTimeMessage(true)
  }, 20000)
  uri = await uploadImageAsync(imageUri)
  setTimeMessage(false)
  clearTimeout(timeSet)
```

```
// firestore에 place 생성
await firestore
  .collection("places")
  .doc(`${category}_${region}_${randomId}`)
  .set({
    id: randomId,
    title,
    uri,
    category,
    location,
    region,
    likes: 0,
    views: 0,
    writer,
    created: moment().format(),
  })

// firestore에 user에 product안에 place 넣기
const userRef = firestore.collection("users").doc(writer)
await firestore.runTransaction(async (t) => {
  const doc = await t.get(userRef)
  // 트랜잭션 보면 트랜잭션 내부에서 직접 데이터 수정하지 말라고 되어있다.
  // 즉 push 쓰지말고 리액트 처럼 하라는 소리다.
  const newUpload = [...doc.data().uploads, randomId]
  await t.update(userRef, { uploads: newUpload })
```

업로드 코드2

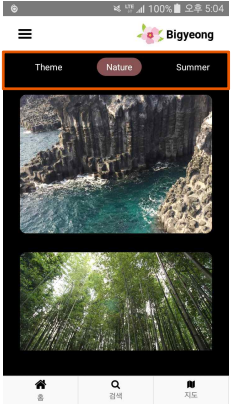
```
} catch (error) {
  console.log(error)
  // 에러시 storage에 이미지 제거
  let imageRef = storage.ref().child("image/" + randomId)
  if (imageRef) await imageRef.delete()
  // firestore의 places에 해당 문서 제거
  let placeRef = firestore
    .collection("places")
    .doc(`${category}_${region}_${randomId}`)
  if (placeRef) await placeRef.delete()
  // firestore의 users의 uploads 해당 요소 제거
  let userRef = firestore.collection("users").doc(writer)
  await firestore.runTransaction(async (t) => {
    const doc = await t.get(userRef)
    let newUpload = doc.data().uploads.filter((target) => {
      return target !== randomId
    })
    await t.update(userRef, { uploads: newUpload })
  })

  if (uri === null) {
    setImageUri(null)
    return alert("이미지를 업데이트 할 수 없습니다")
  }
  return alert("업로드에 실패하였습니다")
} finally {
  setLoading(false)
```

-에레 처리

간혹 firestore에 업데이트가 되었지만
네트워크 에러로 인해 이미지는 있고 user uploads가 비거나
또는 places collection 에는 저장되어 있지만
fire storage에 이미지가 없는 경우가 있어서
에러시 해당 업로드 데이터 모조리 삭제

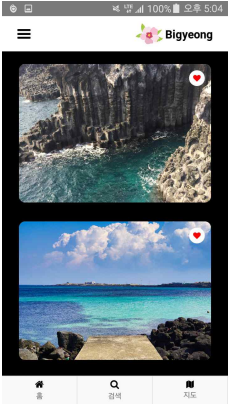
카테고리

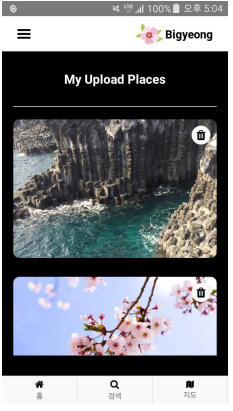


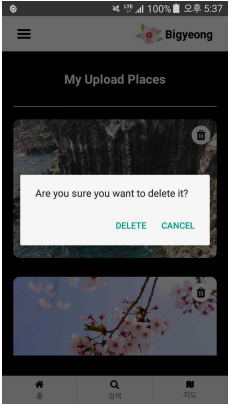

```jsx
<View style={styles.imageColumn}>
  {isFocused && contents.length > 0 && (
    <FlatList
      data={contents}
      keyExtractor={(item, index) => index.toString()}
      renderItem={renderItem}
    />
  )}
</View>
```

```jsx
const getData = async () => {
  try {
    const places = firestore.collection("places")
    let newContents = []
    if (name === "New") {
      const data = await places.get()
      data.forEach((item) => {
        newContents.push(item.data())
      })
    } else {
      const data = await places
        .where("category", "==", name.toLowerCase()) // category 나누기
        .get()
      data.forEach((item) => {
        newContents.push(item.data())
      })
    }
    setContents([...newContents])
  } catch (error) {
    alert("데이터를 불러오는데 실패하였습니다.")
  }
}
```

찜목록 및 프로필


100% 오후 5:04
Bigyeong
홈
검색
지도


100% 오후 5:04
Bigyeong
My Upload Places
홈
검색
지도


100% 오후 5:37
Bigyeong
My Upload Places
Are you sure you want to delete it?
DELETE
CANCEL
홈
검색
지도

프로필 코드

```
//유저 정보 가져오기
const userId = await SecureStore.getItemAsync("key")
const userRef = firestore.collection("users").doc(userId)
const userDoc = await userRef.get()
const uploadId = userDoc.data().uploads
// 업로드한 컨텐츠가 없을 때
if (uploadId.length === 0) return setContents([])
// firestore에서 해당 id를 가진 문서 가져오기
let uploadList = []
const placeRef = firestore.collection("places")
const placeData = await placeRef.where("id", "in", uploadId).get()
placeData.forEach((item) => uploadList.push(item.data()))
setContents(uploadList)
```

firestore의 users uploads에서 컨텐츠 가져오기

```
try {
  await firestore
    .collection("places")
    .doc(`${item.category}_${item.region}_${item.id}`)
    .delete()
  // user uploads에서 제거
  let userRef = firestore.collection("users").doc(item.writer)
  await firestore.runTransaction(async (t) => {
    const doc = await t.get(userRef)
    let newUpload = doc.data().uploads.filter((target) => {
      return target !== item.id
    })
    await t.update(userRef, { uploads: newUpload })
  })
  // storage에서 이미지 삭제
  let imageRef = storage.ref().child("image/" + item.id)
  await imageRef.delete()
```

사진 삭제시 쓰이는 코드

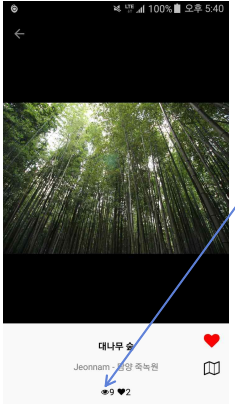
프로필 코드2

새로 고침 누를시 다시 데이터 불러오기

```
  console.log(error)
  setError(true)
  // 만약 네트워크 연결 잘못이라면 알려주기
  NetInfo.fetch().then((state) => {
    if (state.isConnected === false)
      return alert(
        "데이터를 불러오는데 실패하였습니다 네트워크 연결을 확인해 주시기 바랍니다"
      )
  })
  alert("데이터를 불러오는데 실패하였습니다")
} finally {
  setLoading(false)
}
```

```
{error ? (
  /* 에러 시 */
  <>
    <Text style={styles.textStyle}>
      데이터를 불러올 수 없습니다 {"\n"}네트워크 연결을 확인해
      주십시오
    </Text>
    <Button
      title="새로고침"
      color="blue"
      onPress={() => {
        setError(false)
        getData()
      }}
    />
  </>
```

디테일



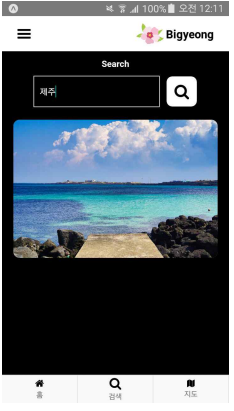

```
<TouchableHighlight
  onPress={() => navigation.navigate("Detail", { item })}
  style={styles.imageContainer}
>
```

navigate로 데이터 전달

```
// 조회수 증가
export const updateView = async (item) => {
  const placeRef = firestore
    .collection("places")
    .doc(`${item.category}_${item.region}_${item.id}`)
  try {
    await placeRef.update({
      views: firebase.firestore.FieldValue.increment(1),
    })
    const doc = await placeRef.get()
    const data = await doc.data()
    return data
  } catch (error) {
    NetInfo.fetch().then((state) => {
      if (state.isConnected === false)
        return alert("네트워크 연결을 확인해 주시기 바랍니다")
    })
    if (error) return "error" // 에러시 Home으로 되돌리도록 설정함
    const doc = await placeRef.get()
    const data = await doc.data()
    return data
  }
}
```

조회수 컨트롤러

검색



firestore에는 직접적인 검색 기능이 없어 따로 구현

```
const searchLogic = async (text) => {
  // 특수문자 및 띄어쓰기 제거
  let special = /[\{\}\[\]\/?.,;:|\)*~`!^\-_+<>@\#$%&\\\=\(\'\" ]/gi
  let cleanText = text.replace(special, "")
  // 한글 체크
  let checkKor
  let korean = /[ㄱ-ㅎ|ㅏ-ㅣ|가-힣]/g
  checkKor = korean.test(cleanText)
  let list = []
  if (checkKor) {
    const placeRef = firestore.collection("places")
    const result = await placeRef.where("title", ">=", cleanText).get()
    result.forEach((item) => {
      list.push(item.data())
    })
  } else {
    const placeRef = firestore.collection("places")
    const result = await placeRef.where("title", "<=", cleanText).get()
    result.forEach((item) => {
      list.push(item.data())
    })
  }
  if (list.length > 0) return list
  else return null
  // 이렇게 해도 일단 데이터는 뒤쪽먹쪽으로 나온다. 나중에 검색 엔진 서비스를 도입해야한다
}
```

감사합니다