

PORTFOLIO

S U B J E C T : Preorder Coffee

P A G E : <https://eunjoungpark.github.io/>

S O U R C E : <https://github.com/eunjoungpark/preorder-coffee>

P E R I O D : 2019.09 - 2019.10 (1 MONTH)

P A R T I C I P A T I O N : 100%

Contents

01_BENCHMARKING

02 _IA

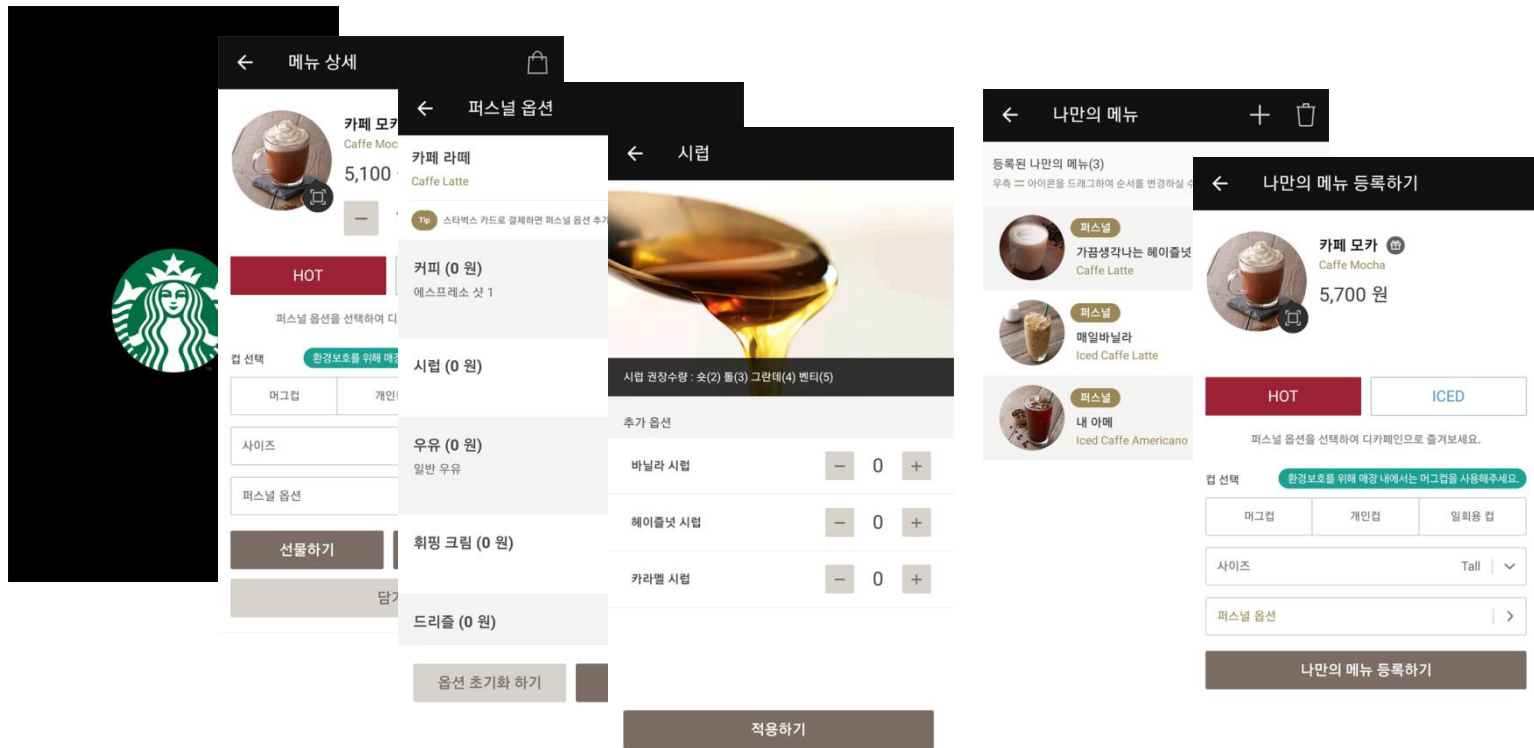
03_INTRODUCE

04_FLOW DIAGRAM

05_SKILLS

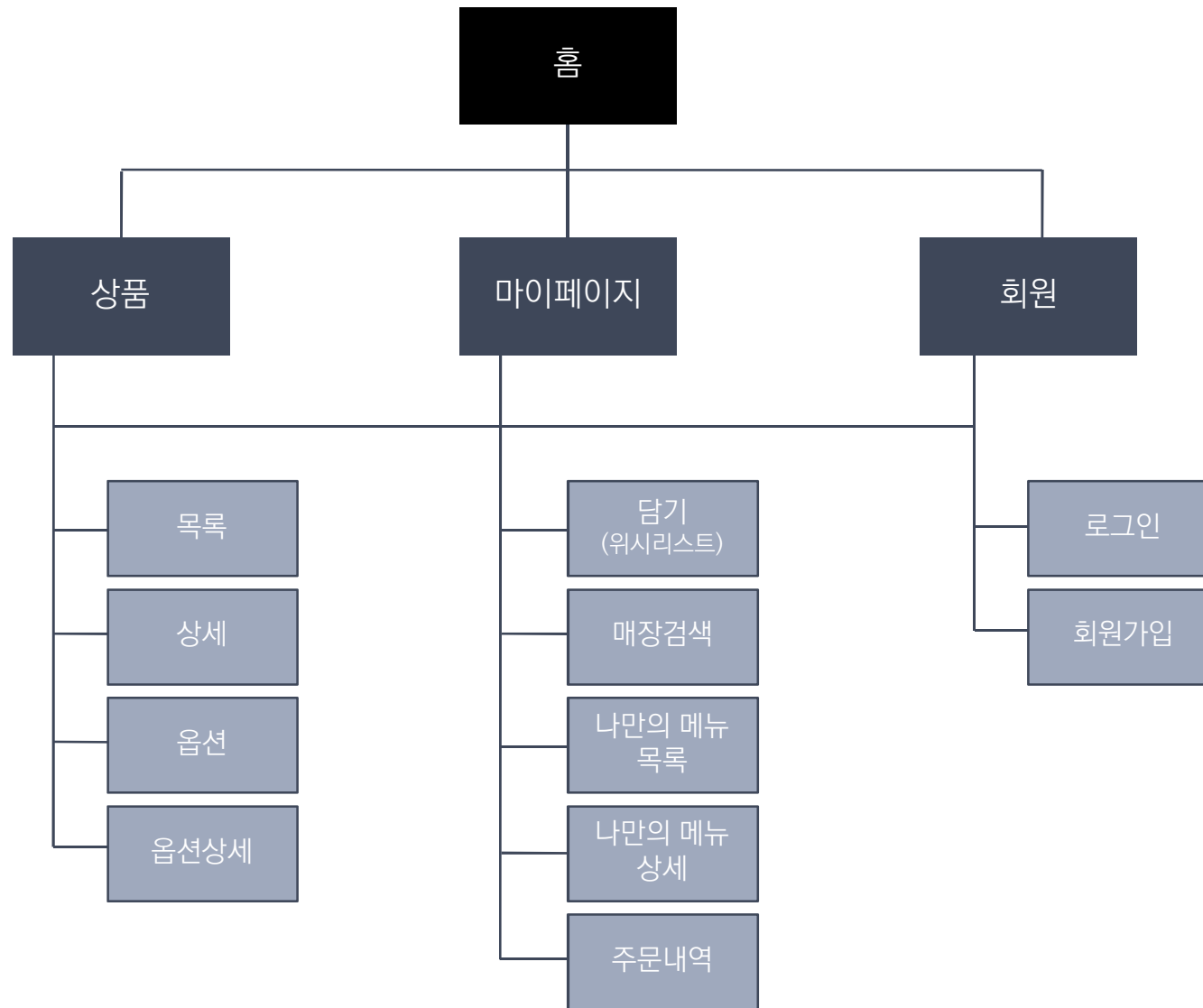
06_PACKAGES

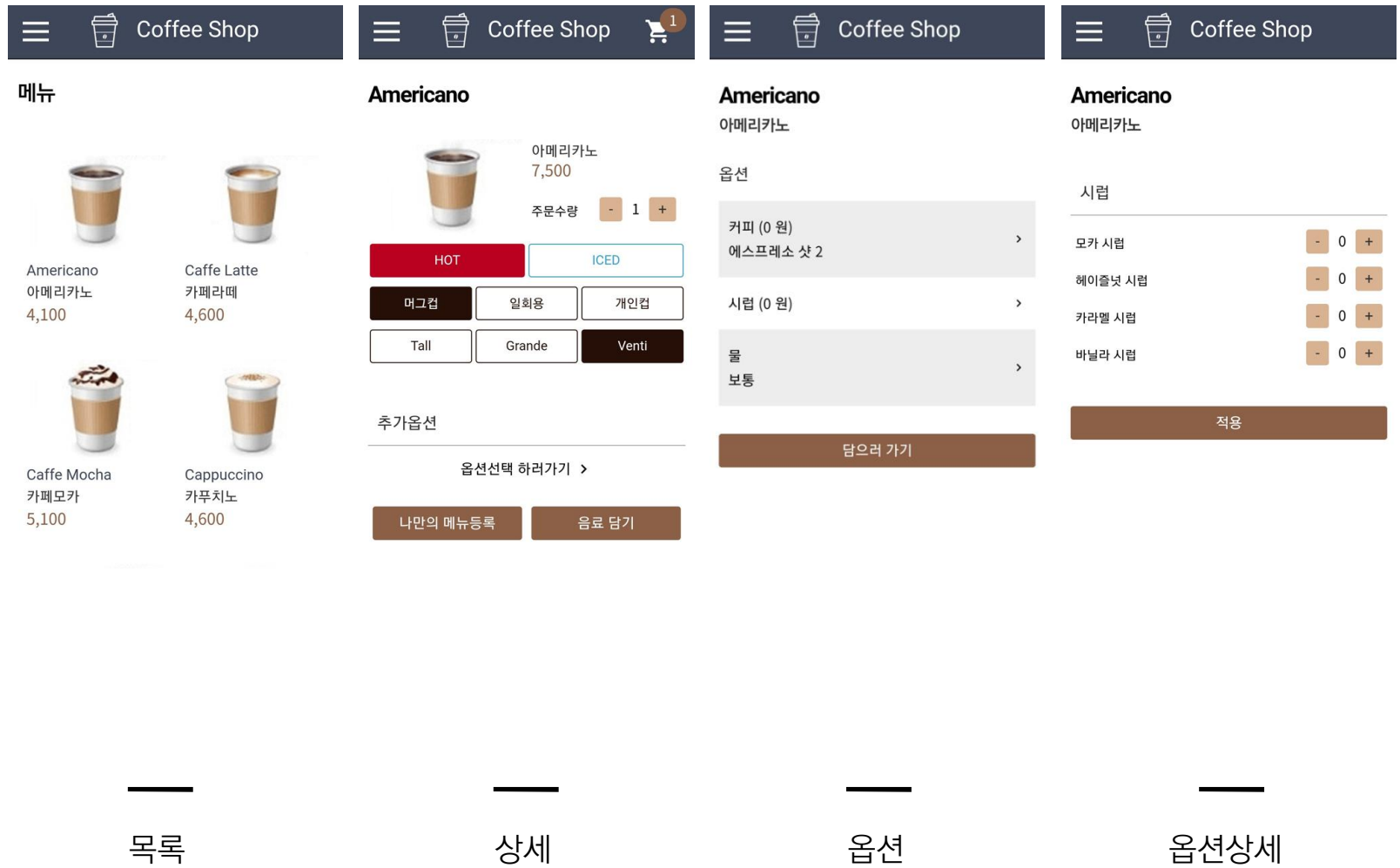
01_BENCHMARKING



스타벅스 Siren Order 벤치마킹

모바일 앱의 비대면 결제 및 개인 맞춤 메뉴 저장으로,
오프라인 주문에 드는 인력 및 시간 절약, 고객 대기 시간 단축을 위한 기능.



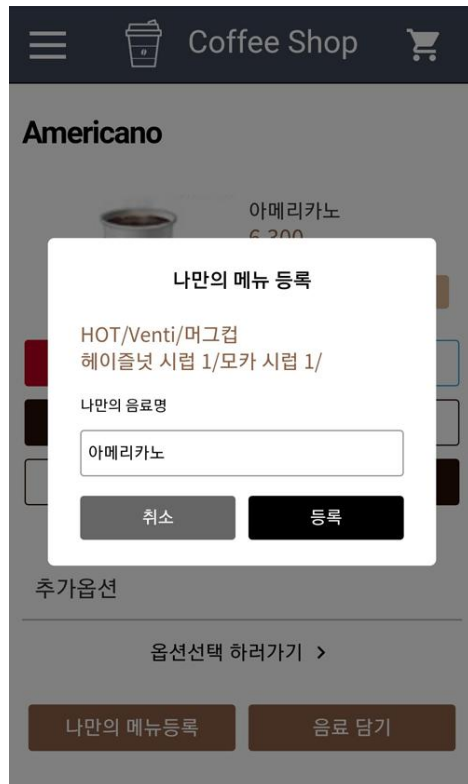


목록

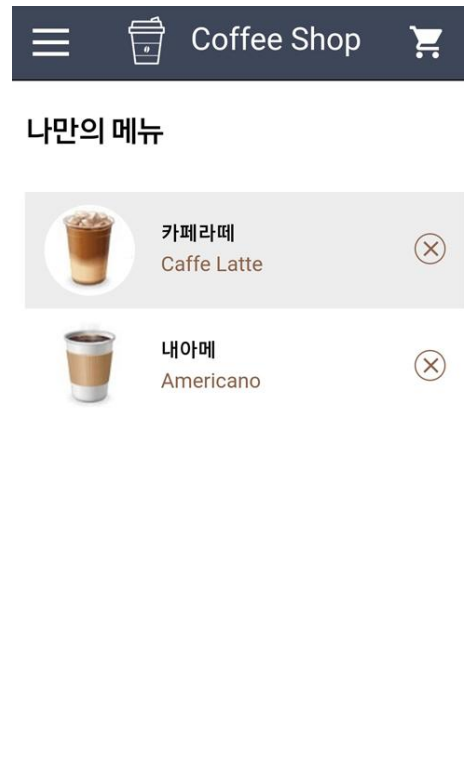
상세

옵션

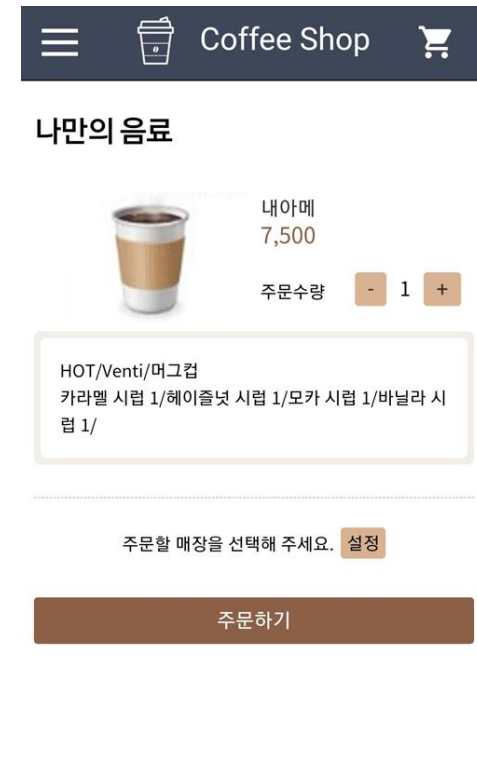
옵션상세



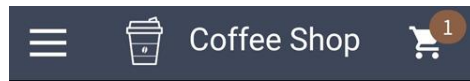
상세 / 나만의 메뉴 등록



나만의 메뉴 / 목록



나만의 메뉴 / 상세



위시 리스트

☒ 전체음료

- ☒ 아메리카노 ×
 - HOT/Venti/머그컵 5,100원
 - 모카 시럽 600원
 - 헤이즐넛 시럽 600원
 - 카라멜 시럽 600원
 - 바닐라 시럽 600원
 - 디카페인 0원

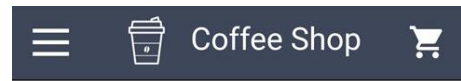
7,500원

총 1개7,500원

주문할 매장을 선택해 주세요. 설정

주문하기

담기(위시리스트)



매장검색

삼성

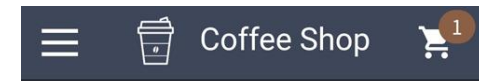


삼성역
서울특별시 강남구 테헤란로1



포스코
서울특별시 강남구 삼성로1

매장검색



주문 내역

NO.	주문 내용	지점	날짜
1	아메리카노 외 8잔 결제액 : 36,900원	논현힐탑	2019.10.19 14 : 47 : 6
2	아메리카노 1잔 결제액 : 4,100원	압구정미소	2019.10.18 19 : 27 : 16
3	아메리카노 1잔 결제액 : 4,100원	GS타워	2019.10.18 19 : 27 : 7
4	아메리카노 1잔 결제액 : 4,100원	차병원사거리	2019.10.18 19 : 23 : 54
5	아메리카노 1잔 결제액 : 4,100원	청담공원	2019.10.18 19 : 23 : 34
6	아메리카노 1잔 결제액 : 4,100원	뱅뱅사거리	2019.10.18 19 : 22 : 48
7	아메리카노 1잔 결제액 : 4,100원	강남비전타워	2019.10.18 19 : 22 : 39

주문 내역

☰ ☕ Coffee Shop

로그인

아이디

유효한 이메일양식입니다.

비밀번호

6자 이상의 문자(소문자, 특수문자, 숫자)를 입력해주세요

확인

로그인

☰ ☕ Coffee Shop

회원가입

아이디

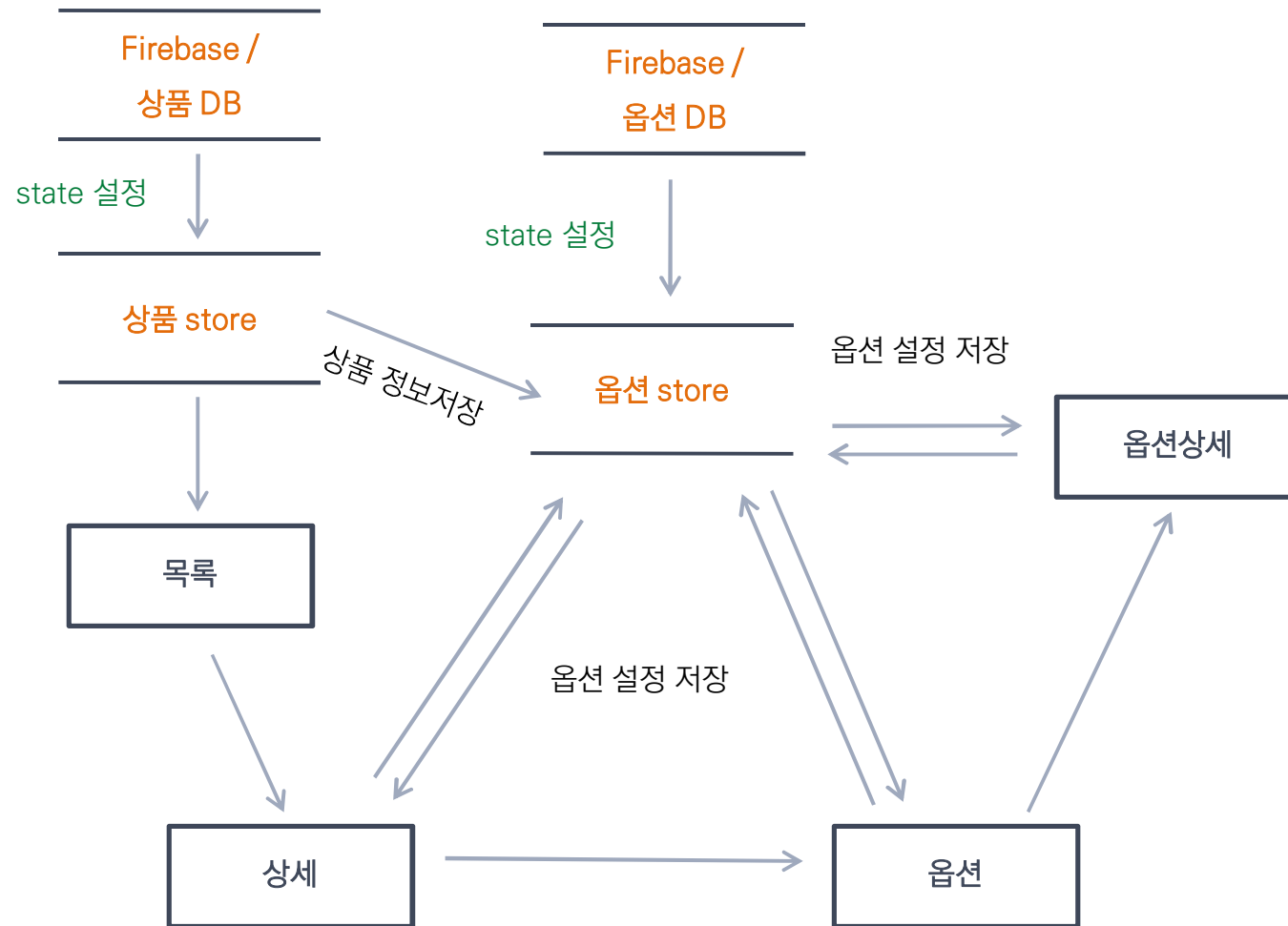
비밀번호

확인

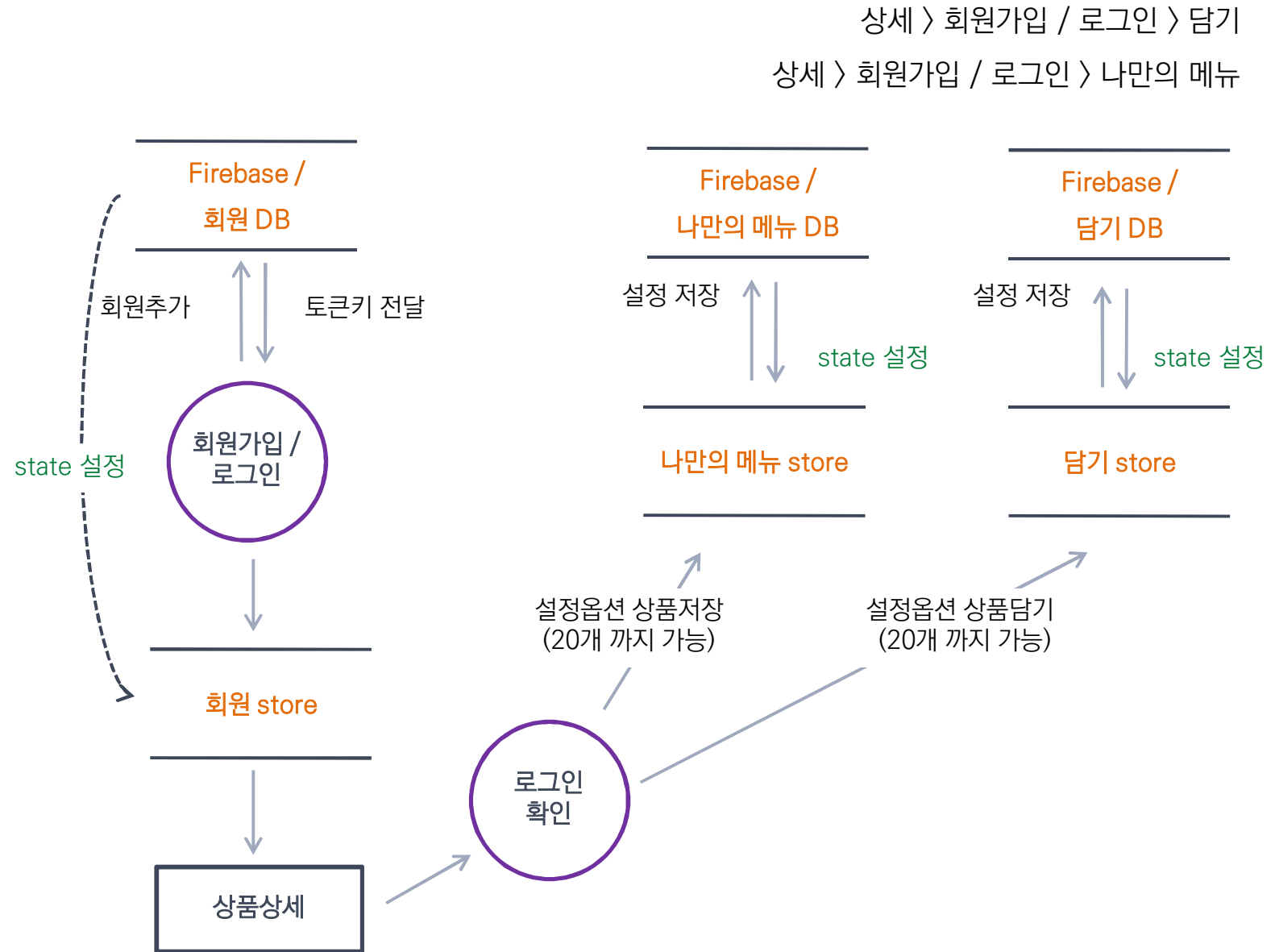
회원가입

04_FLOW DIAGRAM

목록 > 상세 > 옵션 > 옵션상세

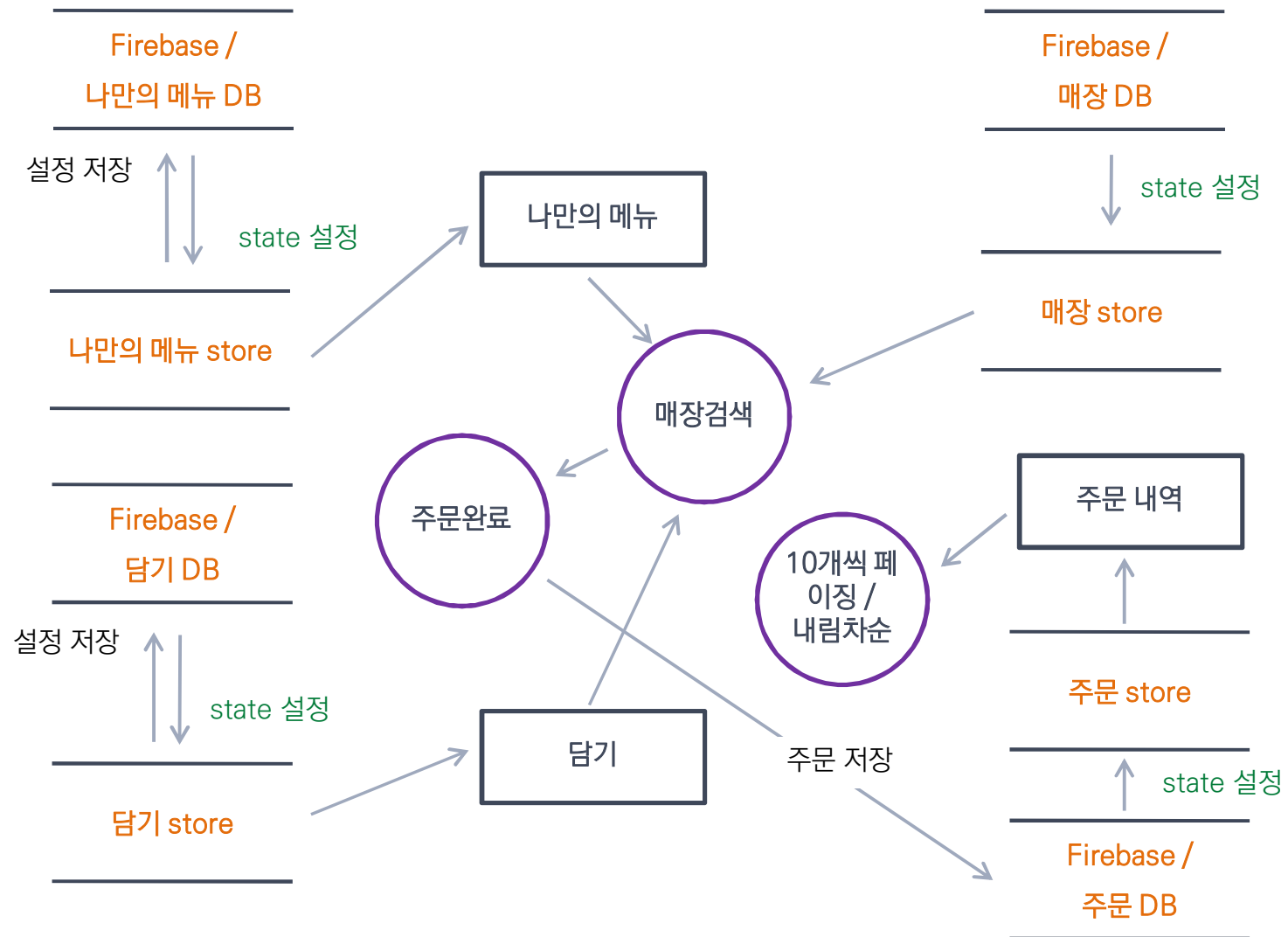


04_FLOW DIAGRAM



04_FLOW DIAGRAM

나만의 메뉴 > 주문완료 > 매장검색 > 주문내역
 담기 > 주문완료 > 매장검색 > 주문내역



```
// 음료주문 최대값
const MAX = 20;
const View = ({
  history,
  location,
  product,
  options,
  auth,
  menuList,
  wish,
  onSetProduct,
  onSetSyrup,
  onSetCount,
  onSetCup,
  onSetSize,
  onSetTotal,
  onSetShot,
  onSetPrice,
  onSetType,
  onSetMessages,
  addWish,
  addMenu,
  loadingAddWish,
  loadingAddMenu,
  emptyLoading,
}) => {
  const { type, kind, keep } = qs(location);
  const s = kind === 'espresso' ? 'Solo' : 'Tall';
  const [hasMsg, setHasMsg] = useState(false);
  const [shown, setShown] = useState(false);
  const [alertModal, setAlertModal] = useState(true);
  const [modalMsg, setModalMsg] = useState('');
  const [menuName, setMenuName] = useState('');
```

함수형 컴포넌트

```
// 음료주문 최대값
const MAX = 20;
const View = ({
  history,
  location,
  product,
  options
```

리액트 HOOKS

- useState
- useEffect
- useCallback
- useMemo
- useRef

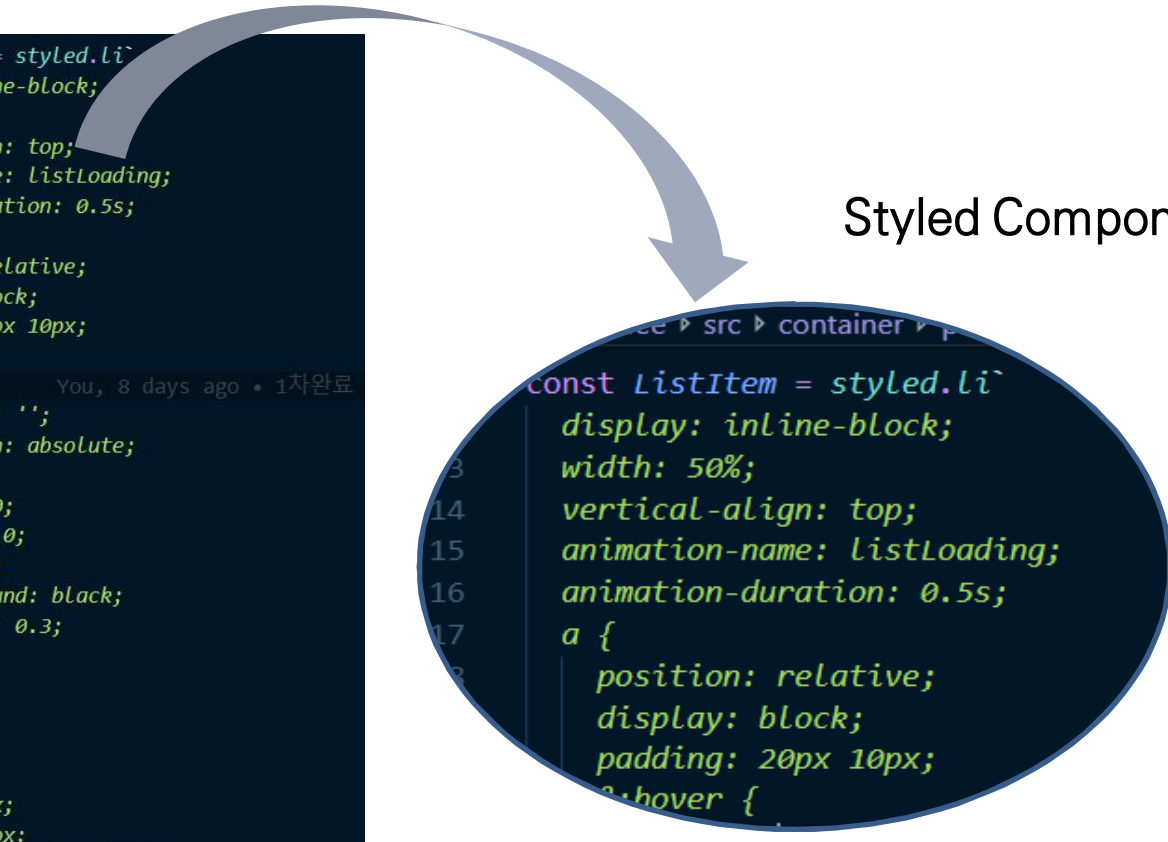
```
const { type, kind, keep } = qs(location);
const s = kind === 'espresso' ? 'Solo' : 'Tall';
const [hasMsg, setHasMsg] = useState(false);
const [shown, setShown] = useState(false);
const [alertModal, setAlertModal] = useState(true);
const [modalMsg, setModalMsg] = useState('');
const [menuName, setMenuName] = useState('');
```

```
const ListItem = styled.li`
  display: inline-block;
  width: 50%;
  vertical-align: top;
  animation-name: listLoading;
  animation-duration: 0.5s;
  a {
    position: relative;
    display: block;
    padding: 20px 10px;
    &:hover {
      :after {
        content: '';
        position: absolute;
        top: 0;
        right: 0;
        bottom: 0;
        left: 0;
        background: black;
        opacity: 0.3;
      }
    }
  }
}

img {
  width: 180px;
  height: 126px;
  margin: 0 auto 10px auto;
  display: block;
}

figcaption {
  p {
    margin-bottom: 7px;
    white-space: nowrap;
    overflow: hidden;
    text-overflow: ellipsis;
  }
}
```

Styled Components



```
const ListItem = styled.li`
  display: inline-block;
  width: 50%;
  vertical-align: top;
  animation-name: listLoading;
  animation-duration: 0.5s;
  a {
    position: relative;
    display: block;
    padding: 20px 10px;
    &:hover {
```

Americano



아메리카노
4,100

주문수량 1

☒ HOT ☐ ICED

☒ 머그컵 ☐ 일회용 ☐ 개인컵

☒ Tall ☐ Grande ☐ Venti

추가옵션

옵션선택 하러가기 >

☒ 나만의 메뉴등록 ☐ 음료 담기

커피

에스프레소 샷 2

디카페인 선택

☐ 1/2 디카페인 ☐ 디카페인

☒ 적용

아이디

비밀번호

☒ 확인

위시 리스트

☒ 전체음료

☒ 아메리카노

HOT/Tall/머그컵 4,100원

4,100원

회원가입

아이디

비밀번호

☒ 확인

로그인

아이디

비밀번호

☒ 확인

요소 컴포넌트 재사용

- 수량모듈
- Input
- Button
- Radio
- Checkbox
- 회원가입/로그인

```
/*
  * 에러메시지 처리 모달
  */
const ErrorHandler = WithErrorHandler => {
  return props => {
    const error = httpHandler();
    return (
      <>
        <Modal shown={error ? null : true}>
          {error ? error.message : null}
        </Modal>
        <WithErrorHandler {...props} />
      </>
    );
  };
};

export default ErrorHandler;
```

고차컴포넌트 (HOC)

- axios 에러 처리

```
    </OptionGroup>
  </Contents>
)
);
};

Option.propTypes = {
  product: PropTypes.object.isRequired,
  options: PropTypes.object.isRequired,
  onSetTotal: PropTypes.func.isRequired,
};

const mapStateToProps = ({ product, options }) => ({
  product: product.product,
  options: options,
});

const mapDispatchToProps = { onSetTotal };

export default connect(
  mapStateToProps,
  mapDispatchToProps,
)(Option);
```

Prop Types

```
Option.propTypes = {
  product: PropTypes.object.isRequired,
  options: PropTypes.object.isRequired,
  onSetTotal: PropTypes.func.isRequired,
```

```
const mapStateToProps = ({ product, options }) => ({
  product: product.product,
  options: options,
});

const mapDispatchToProps = { onSetTotal };

export default connect(
  mapStateToProps,
  mapDispatchToProps,
)(Option);
```

Redux
(with React-redux)

Redux Actions

- createAction
- handleActions

```
import { createAction, handleActions } from 'redux-actions';
import { delay, call, put, takeLatest } from 'redux-saga/effects';
import * as api from '../libs/api';
import { startLoading, finishLoading } from './loadings';
import { errorMessage } from './error';
import produce from 'immer';

/* TYPE 정의 */
const INIT_PRODUCTS = 'product/INIT_PRODUCTS';
export const SET_PRODUCTS = 'product/SET_PRODUCTS';
const INIT_PRODUCT = 'product/INIT_PRODUCT';
const SET_PRODUCT = 'product/SET_PRODUCT';

/* ACTION 정의 */
export const initProducts = createAction(INIT_PRODUCTS);
export const initProduct = createAction(INIT_PRODUCT);
export const onSetProduct = createAction(SET_PRODUCT, payload => payload);

/* 비동기 정의 */
const fetchListAsync = function*() {
  // You, 8 days ago • 1지완료
};

/* 비동기호출 정의 */
export const productListAsync = function*() {
  // ...
};

/* 초기값 정의 */
const initialState = {
  lists: null,
  product: null,
};

/* REDUCER 정의 */
const product = handleActions(
  {
    [SET_PRODUCTS]: (state, { payload: products }) =>
```

Ducks Pattern

- 액션 타입
- 액션 생성 함수
- 리듀서

Immutability

- Immer

```
import { createAction, handleActions } from 'redux-actions';

/* TYPE 정의 */
const INIT_PRODUCTS = 'product/INIT_PRODUCTS';
export const SET_PRODUCTS = 'product/SET_PRODUCTS';
const INIT_PRODUCT = 'product/INIT_PRODUCT';
const SET_PRODUCT = 'product/SET_PRODUCT';

/* ACTION 정의 */
export const initProducts = createAction(INIT_PRODUCTS);
export const initProduct = createAction(INIT_PRODUCT);
export const onSetProduct = createAction(SET_PRODUCT, payload => payload);

/* 초기값 정의 */
const initialState = {
  lists: null,
  product: null,
};

/* REDUCER 정의 */
const product = handleActions(
  {
    [SET_PRODUCTS]: (state, { payload: products }) =>
      produce(state, draft => {
        draft.lists = products;
      }),
    [INIT_PRODUCT]: (state, action) =>
      produce(state, draft => {
        draft.product = null;
      }),
    [SET_PRODUCT]: (state, { payload: { type, kind } }) =>
      produce(state, draft => { ...
    },
    initialState,
  );
```

```

/* 비동기 정의 */
const fetchListAsync = function*() {
  yield put(startLoading(SET_PRODUCTS));
  try {
    const response = yield call(api.products);
    yield delay(500);
    yield put({
      type: SET_PRODUCTS,
      payload: response.data,
    });
  } catch (e) {
    yield put(errorMessage({ action: SET_PRODUCTS, message: e }));
  }
  yield put(finishLoading(SET_PRODUCTS));
};

/* 비동기호출 정의 */
export const productListAsync = function*() {
  yield takeLatest(INIT_PRODUCTS, fetchListAsync);
};

export function* rootSaga() {
  yield all([
    productListAsync(),
    optionsAsync(),
    authAsync(),
    storeAsync(),
    wishAsync(),
    menuAsync(),
    orderAsync(),
  ]);
}

```

Redux Saga

- call
- put
- delay
- takeEvery
- takeLatest
- all

```
import axios from 'axios';

const instance = axios.create({
  baseURL: 'https://preorder-coffee.firebaseio.com',
});

const WEB_KEY = 'AIzaSyDR1SsLdbxCfX-Eb2dnqscsLqF21mhwpHo';

//COFFEE 종류
export const products = () => instance.get('/coffee.json');

//OPTIONS
export const options = () => instance.get('/options.json');

//STORES      You, 8 days ago * 1자원료
export const store = () => instance.get('/store.json');

//WISH ADD
export const addWish = (token, userId, wish) =>
  instance.post(`/wish/${userId}.json?auth=${token}`, wish);

//WISH UPDATE
export const updateWish = (token, userId, wish) =>
  instance.put(`/wish/${userId}.json?auth=${token}`, wish);

//WISH UPDATE
export const updateAWish = (token, userId, id, wish) =>
  instance.put(`/wish/${userId}/${id}.json?auth=${token}`, wish);

//WISH REMOVE
export const removeWish = (token, userId, id) =>
  instance.delete(`/wish/${userId}/${id}.json?auth=${token}`);

//WISH GET
export const getWish = (token, userId) =>
  instance.get(`/wish/${userId}.json?auth=${token}&orderBy="date"`);
```

HTTP

- AXIOS
- HTTP ERROR처리
(axios.interceptors)

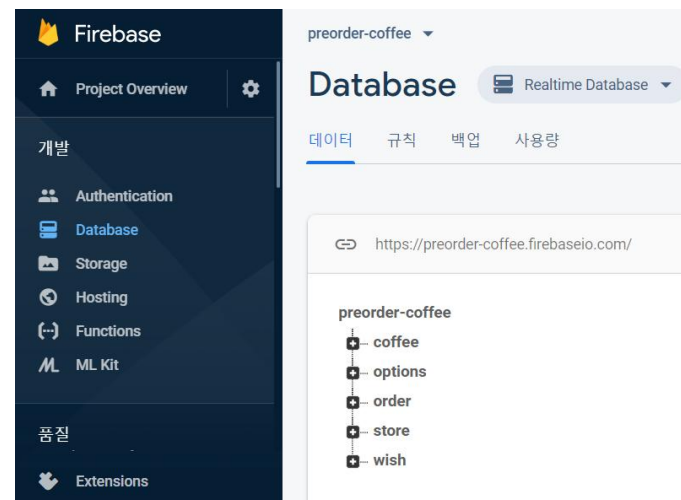
DB

- Firebase

```
import { useState, useEffect } from 'react';
import axios from '../libs/api';

export default () => {
  const [error, setError] = useState(null);
  const response = axios.interceptors.response.use(
    res => res,
    error => {
      setError(error);
    },
  );
  const request = axios.interceptors.request.use(req => {
    setError(null);
    return req;
  });

  useEffect(() => {
    return () => {
      axios.interceptors.request.eject(request);
      axios.interceptors.response.eject(response);
    };
  }, [request, response]);
  return error;
};
```





예시) 상품상세 합계계산

$[(\text{상품금액} + (\text{에스프레소 샷 갯수} * \text{단가}) + (\text{각 시럽 개수} * \text{단가})) * \text{상품갯수}]$

성능 최적화 - chrome performance (73.5ms)

- useMemo
- useCallback

```
const onScroll = useCallback(() => {
  const { innerHeight } = window;
  const { scrollHeight } = document.body || document.documentElement;
  const scrollTop =
    document.body.scrollTop || document.documentElement.scrollTop;
  if (scrollHeight - innerHeight - scrollTop < 10) {
    setIsScroll(true);
  } else {
    setIsScroll(false);
  }
}, []);

useEffect(() => {
  window.addEventListener('scroll', onScroll, false);
  return () => {
    emptyLoading(ORDER_LIST);
    emptyOrderList();
    window.removeEventListener('scroll', onScroll, false);
  };
}, []);

useEffect(() => {
  if (page && !finish) {
    initOrderList({
      token: auth.idToken,
      userId: auth.localId,
      page,
      limit: LIMIT,
    });
  }
}, [isScroll]);
```

성능 최적화

- scroll event

(주문내역 10개씩 페이징)


```

const View = React.lazy(() => import('./container/products/view'));
const Options = React.lazy(() => import('./container/products/Options'));
const Details = React.lazy(() => import('./container/products/Details'));
const MyMenu = React.lazy(() => import('./container/myPage/MyMenu'));
const MyMenuPay = React.lazy(() => import('./container/myPage/MyMenuPay'));
const Orders = React.lazy(() => import('./container/myPage/Orders'));
const SignUp = React.lazy(() => import('./container/member/SignUp'));
const SignIn = React.lazy(() => import('./container/member/SignIn'));
const Wish = React.lazy(() => import('./container/myPage/Wish'));
const Store = React.lazy(() => import('./container/myPage/Store'));

const App = ({
  lists,
  auth,
  wish,
  loadingProducts,
  initProducts,
  initOptions,
  removeAuth,
  accessAuth,
  initWishList,
  initMenu,
  emptyMenu,
  emptyWishList,
}) => {
  const [cntWish, setCntWish] = useState(0);
  const [appear, setAppear] = useState(false);
  const [alertModal, setAlertModal] = useState(true);
  const [modalMsg, setModalMsg] = useState('');

  let route = (
    <Switch>
      <Route path="/view" render={prop => <View {...prop} lists={lists} /> />
      <Route path="/options" component={Options} />
      <Route path="/details" component={Details} />
      <Route path="/signup" component={SignUp} />
      <Route path="/signin" component={SignIn} />
    </Switch>
  );

```

React.lazy

```

const View = React.lazy(() => import('./container/products/view'));
const Options = React.lazy(() => import('./container/products/Options'));
const Details = React.lazy(() => import('./container/products/Details'));
const MyMenu = React.lazy(() => import('./container/myPage/MyMenu'));
const MyMenuPay = React.lazy(() => import('./container/myPage/MyMenuPay'));
const Orders = React.lazy(() => import('./container/myPage/Orders'));
const SignUp = React.lazy(() => import('./container/member/SignUp'));
const SignIn = React.lazy(() => import('./container/member/SignIn'));
const Wish = React.lazy(() => import('./container/myPage/Wish'));
const Store = React.lazy(() => import('./container/myPage/Store'));

```

Router &
Suspense

```

if (auth.LocalId) {
  route = (
    <Switch>
      <Route path="/view" render={prop => <View {...prop} lists={lists} /> />
      <Route path="/options" component={Options} />
      <Route path="/details" component={Details} />
      <Route path="/mymenu" exact component={MyMenu} />
      <Route path="/mymenu/pay" component={MyMenuPay} />
      <Route path="/orders" component={Orders} />
      <Route path="/wish" component={Wish} />
      <Route path="/store" component={Store} />
      <Redirect to="/" />
    </Switch>
  );
}

<Suspense fallback={<Loading />}>{route}</Suspense>

```

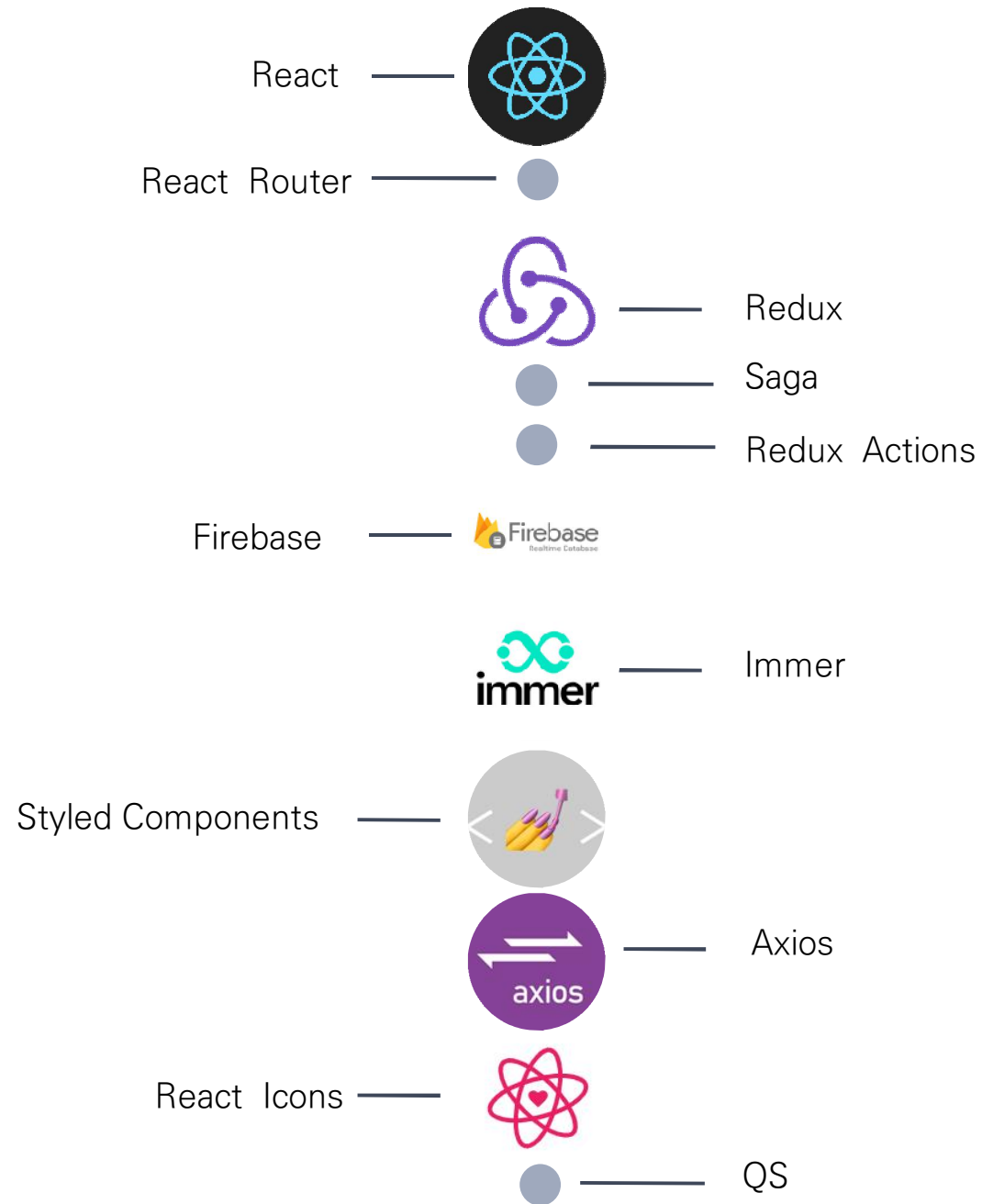
ECMA SCRIPT 6

- let, const
- 구조분해
- 전개 연산자
- 화살표 함수
- 템플릿 리터럴
- 제너레이터
- 배열 함수
- Promise
- export / import

WEB PUBLISHING

- HTML5
- CSS3
- 웹표준
- 웹접근성 (WAI-ARIA)

06_PACKAGES



감사합니다.