ATLAS annotation 색상 변경 작업 내역

SideMenu.tsx (modal 내부에 color picker 작업 내역 - 수정 후)

> 주요 변경 사항
>
> 1. `antd Form` 컴포넌트 사용
> 2. edit content 로직과 컬러 변경 로직을 한 번의 API 호출로 가능하도록 개선
> 3. state 관리 항목, effect 라이프싸이클 처리 로직 일부 추가
> 4. function & component 일부 수정

```
/* useState 추가 */
const [currentAnnotation, setCurrentAnnotation] = useState<object>({});

/* useForm 추가 */
const [annotationForm] = Form.useForm();

/* useEffect 추가 */
// annotation 수정 모달이 열릴 때
useEffect(() => {
  if (showAnnotationModal) {
     resetAnnotationModalForm();
  }
}, [showAnnotationModal])

// 현재 선택된 annotation을 관리하기 위해 선택된 종속 배열 값들을 추적해서 실시간 반영
useEffect(() => {
  setCurrentAnnotation(currentSlide?.annotations.filter((item) => item.annoId === currentAnnotationId))
}, [currentAnnotationId, currentSlide?.annotations])
```

changeColor function

```
const changeColor = (color: string) => {
    annotationForm.setFieldsValue({color}); // form 필드 데이터에 선택한 색상코드 값 추가 { color: selected }
}
```

getCurrentColor function

```
  const getCurrentColor = (obj: Object) => {
    let currentColor = "red"; // 색상 초기화
    let ca = null; // current annotation
    if(currentAnnotation) ca = Object.values(currentAnnotation); // JSON 객체 형식이라 object로 변환

    if(ca !== null && ca[0]) {
      const color = ca[0].annoInfo[1].stroke; // annoInfo 1번 인덱스에 있는 색상만 추출
      currentColor = color;
    }
    return currentColor;
  }
```

+ handleAnnotationContent & resetAnnotationModalForm function

```
const handleAnnotationContent = (e: any) => {
    setAnnotationContent(e.target.value);
    annotationForm.setFieldsValue({content: e.target.value});
}

const resetAnnotationModalForm = () => {
    annotationForm.resetFields();
}
```

render component

```
     <Modal centered title='Edit Annotation' open={showAnnotationModal} footer={null} onCancel={() => setShowAnnotationModal
        <Form
          form={annotationForm}
          name="update-annotation"
          onFinish={editAnnotation}
          style={{maxWidth: 600, marginTop:"1.5rem"}}
```

```
        >
          <Form.Item name="content" label="description" style={{margin: "0.6rem 0"}}>
            <Input onChange={handleAnnotationContent} defaultValue={annotationContent} />
          </Form.Item>
          <Form.Item name="color" label="color" style={{ margin: "0.6rem 0 1rem 0" }}>
            <AnnotationColorBtn changeColor={changeColor} currentColor={getCurrentColor()} />
          </Form.Item>
          <Form.Item style={{display:"flex", justifyContent:"flex-end"}}>
            <Space>
              <Button type="primary" htmlType="submit">
                Ok
              </Button>
              <Button htmlType="button" onClick={() => setShowAnnotationModal(false)}>
                Cancel
              </Button>
            </Space>
          </Form.Item>
        </Form>
      </Modal>
```

```
// slideApi.ts
// annoInfo 추가
export const patchAnnotation = async( annotationId: number, annoInfo: object, content: string ) => {
  try {
    const body = { id: annotationId, annoInfo, content };
    console.log(body);
    const { data } = await api().patch('/annotation', body);
    return data;
  }
  catch(e) {
    console.log(e);
    return {};
  }
};
```

Component - AnnotationColorBtn

```
// handleColor 일부 로직 수정 및 props 삭제 - annoId
import { ColorPicker } from 'antd';
import { ColorFactory } from 'antd/es/color-picker/color';
import { useState } from 'react';

interface AnnotationColorBtnProps {
  changeColor: (color: string ) => void;
  currentColor: string;
}

const AnnotationColorBtn = ({ changeColor,  currentColor }: AnnotationColorBtnProps) => {
  const [haxColor, setHaxColor] = useState<string>(currentColor || "red");

  const handleColor = (metaColor: ColorFactory) => {
    const value = metaColor.toHex();
    setHaxColor('#' + value);
    changeColor("#" + value);
  }

  return (
    <>
      <ColorPicker defaultValue={haxColor} onChangeComplete={handleColor} size="small" />
    </>
  )
}

export default AnnotationColorBtn;
```