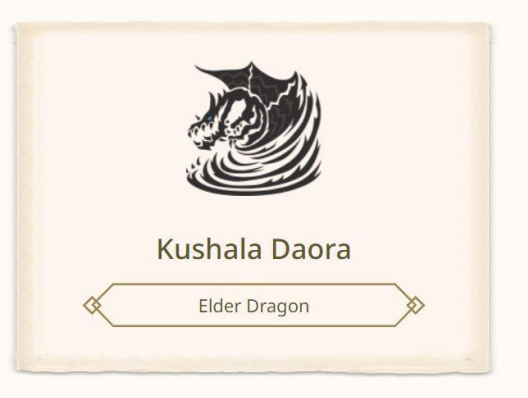
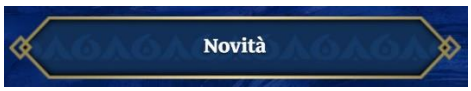
Documentazione progetto TASD

Design

Il design del progetto è stato ispirato dal sito ufficiale di Monster Hunter, ovviamente con molte modifiche, anche alcuni elementi sono stati presi direttamente dal sito





Header

Il codice del menù è stato preso dal sito ufficiale di bootstrap pe poi essere modificato successivamente per rispecchiare le esigenze del design

```
return(
  <div className={style.navBar}>
      <Navbar className="py-0" expand="md" full light>
          <div className="container">
              <div className="row" id={style.line}>
                  <div className="col-12 col-md-6 d-flex justify-content-between">
                      <NavbarBrand className={style.brandLogo}>
                          <RouterLink to="/">
                              <img className={style.logo} src={logo} alt="Monster Hunter"/>
                          </RouterLink>
                      </NavbarBrand>
                      <NavbarToggler className="ms-auto" onClick={toggle} />
                  </div>
                  <div className="col-12 col-md-6 d-flex align-items-center">
                      <Collapse className="align-self-center" isOpen={isOpen} navbar>
                          <Nav id={style.navvino} className="ms-auto" navbar>
                          {itemList}
                          </Nav>
                      </Collapse>
                  </div>
              </div>
          </div>
      </Navbar>
  </div>
)}

export default Header;
```

È stato usato un array che permetta di inserire dinamicamente le voci del menù

```
const itemList = navItems.map((item) => {
    return(
        <NavItem key={item.url} className={style.navItem}>
            <RouterLink
            exact={item.exact}
            activeClassName={style.active}
            to={item.url}
            className="nav-link">
                {item.text}
            </RouterLink>
        </NavItem>
    )
});
```

Home

Il codice della home utilizza un componente < MonsterCard > passando come props delle informazioni ricavate dall'api per poi filtrarle con una semplice funzione .filter() per ottenere 3 specifiche card

```
function Home(props){
    const {col} = props;
    const [monsterData, setMonsterData] = useState([]);

    useEffect(() =>{

        let isMounted = true;

        fetch(`https://mhw-db.com/monsters`)
        .then(res => res.json())
        .then( res => {
            if (isMounted)
            setMonsterData(res);
        })
        .catch((error) => console.log("Error" + error));

        return () => {
            isMounted = false;
        }

    });
```

Chiamata all'api

```
const monsterDataShort = monsterData.filter((monster, id) => id >=33 && id <=35)
const monsterCardsCol = monsterDataShort.map ((monster) =>{


    return (
        <div className="col-12 col-md-6 col-lg-4">
            <MonsterCard
            name={monster.name}
            number={monster.id}
            image={getMonsterIcon(monster.name)}
            species={monster.species}
            />
        </div>
    )
});
```

Filtro e card

Monster Card

Il codice della è molto semplice vengo esposte poche informazioni, degno di nota è come viene passato l'url delle immagini ovvero viene chiamata una funzione che permette di modificare l'url in base al nome del mostro

```
function MonsterCard(props) {
    const {name, image, number, species} = props;

    return (
        <NavLink to={`/monsterlist/${number}`}>
            <div className="mb-5 pb-2">
        <Card className={style.card}>
            <CardImg className={style.image} top width="100%" onError={(event) => monsterDefaultImage(event)} src={image} alt={name}/>
            <CardBody className="text-center">
                <CardTitle tag="h5" className={'h3 ${style.title}'}>{name}</CardTitle>
                <CardText>

                </CardText>
                <div className={style.type}>
                    <p className={style.species}>{species}</p>
                </div>
            </CardBody>
        </Card>
        </div>
        </NavLink>
    )
}
```

```
export const noSpace = (nome) => nome?.replace(/ /g, "_");

export const getMonsterIcon = (name) =>`https://raw.githubusercontent.com/CrimsonNynja/monster-hunter-DB/master/icons/MHW-${noSpace(name)}_Icon.png`

export const monsterDefaultImage = (onErrorEvent) => onErrorEvent.target.src = Emblem;
```

La funzione noSpace serve per eliminare gli spazzi dal nome del mostro per permettere al url di funzionare

Monster List

Questa è la parte principale del codice dove viene chiamata l'api per poi essere passata come props ai 2 componenti <MosterCardGrids> e <MonsterTable>, viene imortato clsx per permettere di inserire gli stili condizionali e viene utilizzato setDisplayGrid per dare un valore booleano che permetta di cambiare fra i 2 componenti

```
const [displayGrid, setDisplayGrid] = useState("true");
const [monsterData, setMonsterData] = useState([]);
useEffect(() =>{
    let isMounted = true;
    fetch(`https://mhw-db.com/monsters`)
    .then(res => res.json())
    .then( res => {
        if (isMounted)
        setMonsterData(res);
    })
    .catch((error) => console.log("Error" + error));
    return () => {
        isMounted = false;
    }
});
return (
    <div className="container">
        <div id={style.h1MonsterList} className="row">
            <div className="col-12 col-md-6 d-flex align-items-center justify-content-center justify-content-md-start">
                <h1 >Monster List</h1>
            </div>
            <div className="col-12 col-md-6 d-flex align-items-center justify-content-center justify-content-md-end">
                <div className={style.switch}>
                    <div className={clsx(style.option, {[style.active]: displayGrid})}
                    onClick={() => setDisplayGrid(true)}>
                        Grid
                    </div>
                    <div className={clsx(style.option, {[style.active]: !displayGrid})}
                    onClick={() => setDisplayGrid(false)}>
                        Table
                    </div>
                </div>
            </div>
        </div>
        <div className="row justify-content-center">
            <div className="col">
                {
                displayGrid ?
                <MonsterCardsGrid
                monsterSheet={monsterData}
                col={{xs:1, sm:2, md:3, lg:4, xl:5}}
                /> :
                <MonsterTable monsterSheet={monsterData}/>
                }
            </div>
```

Moster Card Grids

Molto simile al codice visto nella pagina home ad eccezione del filtro che qui non è stato applicato siccome vogliamo che vengano restituiti tutti i mostri, viene richiamato il componente <MonsterCard> che abbiamo già affrontato precedentemente

```
function MonsterCardsGrid(props) {
    const {monsterSheet, col} = props;

    const monsterCardsCol = monsterSheet.map ((monster) =>{
        return (
            <div className="col-12 col-md-6 col-lg-4">
                <MonsterCard
                name={monster.name}
                number={monster.id}
                image={getMonsterIcon(monster.name)}
                species={monster.species}
                />
            </div>
        )
    });
    return (
        <div
        className={`row row-col-${col.xs} row-col-sm-${col.sm} row-col-md-${col.md} row-col-lg-${col.lg} row-col-xl-${col.xl}`}
        >
            {monsterCardsCol}
        </div>
    )
}
```

Monster Table

Qui il codice restituisce una tabella con presenti delle informazioni sui mostri, la logica è molto simile a quella usata e costruire le card ma le informazioni vengono riportate in forma tabellare

```
function MonsterTable(props) {
    const {monsterSheet} = props;

    const monsterTr = monsterSheet.map((monster)=> {

        return (
            <NavLink to={`/monsterlist/${monster.id}`}>
                    <tr className="row py-4" key={monster.id}>
                        <td className="col-5 col-md-3 d-flex justify-content-center" id={style.monsterImage}>
                            <div>
                            <img onError={(event) => monsterDefaultImage(event)}
                            src={getMonsterIcon(monster.name)}
                            alt={monster.name}
                            loading="lazy"/>
                            </div>
                        </td>
                        <td className="col-7 col-md-4 ">
                            <h3 className="">{monster.name}</h3>
                            <div className="d-md-none">
                                <div className={style.type}>
                                    <p className={style.species}>{monster.species}</p>
                                </div>
                            </div>
                        </td>
                        <td className="col col-md-5 d-none d-md-table-cell">
                            <div className={style.type}>
                                <p className={style.species}>{monster.species}</p>
                            </div>
                        </td>
                    </tr>
            </NavLink>
        )
    });
    return (
    <table className="table">
        <thead className="container">
            <tr className="row">
                <th className="text-center col-5 col-md-3">Icon</th>
                <th className="col-7 col-md-4">Name<span className="d-md-none">/Species</span></th>
                <th id={style.marginTh} className="col-7 text-md-center col-md-5 d-none d-md-table-cell">Species</th>
            </tr>
        </thead>
        <tbody className="container">
            {monsterTr}
        </tbody>
    </table>
```

Monster Details

In questa pagina non viene usata nessuna nuova funzione ma vengono usate molte di quelle delle altre pagine, un dettagli degno di nota è l'utilizzo della funzione require() usate nell'atributo src per non dover importare una ad una le immagini dalla cartella degli assets

```
<p className="text-center mb-2 mt-4">Element/Status:</p>
<div className="row pt-3 px-4">
{monsterData.elements &&
    <>
    {elements.map((elementsItem) => {
        return <div className="col text-center">
            <div className={style.elem}>
            <img key={elementsItem.src}
        src={require(`../../assets/images/${elementsItem.src}.png`).default }
        alt={elementsItem.src}/>
        </div>
        <p className="text-capitalize">{elementsItem.src}</p>

        </div>
        }

    )}
    </>
}
```