PROGETTO HACKER

PUNTO 1

Capire cosa fa il programma senza eseguirlo:

```c
#include <stdio.h>
void menu ();
void moltiplica ();
void dividi ();
void ins_string();
int main () {
    char scelta = {'\0'};
    menu ();
    scanf ("%d", &scelta);
    switch (scelta)
    {
        case 'A':
        moltiplica();
        break;
        case 'B':
                dividi();
                break;
        case 'C':
                ins_string();
                break;
    }
return 0;
}
void menu (){
    printf ("Benvenuto, sono un assitente digitale, posso aiutarti a sbrigare alcuni compiti\n");
    printf ("Come posso aiutarti?\n");
    printf ("A >> Moltiplicare due numeri\nB >> Dividere due numeri\nC >> Inserire una stringa\n");
}
void moltiplica ()
{
    short int  a,b = 0;
    printf ("Inserisci i due numeri da moltiplicare:");
    scanf ("%f", &a);
    scanf ("%d", &b);
    short int prodotto = a * b;
    printf ("Il prodotto tra %d e %d e': %d", a,b,prodotto);
}
void dividi ()
{
        int  a,b = 0;
        printf ("Inserisci il numeratore:");
        scanf ("%d", &a);
    printf ("Inserisci il denumeratore:");
        scanf ("%d", &b);
        int divisione = a % b;
        printf ("La divisione tra %d e %d e': %d", a,b,divisione);
}
void ins_string ()
{
    char stringa[10];
        printf ("Inserisci la stringa:");
        scanf ("%s", &stringa);
}
```

Nello screenshot in alto, è possibile notare come l'implementazione di questo codice offra all'utente la possibilità di eseguire diverse operazioni, tra cui moltiplicazione, divisione e l'inserimento di una stringa con una lunghezza massima di 10 caratteri. Tuttavia è immediatamente evidente, come il seguente codice abbia degli errori di sintassi /logici, che saranno trattati e risolti nelle fasi successive.

PUNTO 2
Individuare dal codice sorgente le casistiche non standard che il programma non gestisce (esempio, comportamenti potenziali che non sono stati contemplati)

Dividerò in vari punti le possibili casistiche non contemplate:

1.Cosa succederebbe se l'utente digitasse una lettera diversa?un numero?Una lettera maiuscola o minuscola?

-Il programma non prevede tutto ciò ,infatti se noi digitassimo qualsiasi cosa all'infuori delle lettere (esclusivamente maiuscole) "A" ,"B" ,"C" il programma terminerebbe all'istante. Vedremo come implementare in modo ottimale questa parte di codice nel punto 4.

2.Se noi volessimo utilizare l'operazione "Moltiplicazione" con numeri elevati come nel seguente caso : "1.000.000*1.000.000". Il programma che risultato darebbe?

-Darebbe un risultato non corretto ,riscontrando un problema di Overflow(poiché il tipo di dato short-int copre un range da -32.768 a +32.767).Vedremo come implementare in modo ottimale questa parte di codice nel punto 4.

3.Nel caso in cui l'utente volesse dividere un qualsiasi numero per "0"?il programma cosa darebbe ?E se l'utente volesse dividere 2 numeri decimali ?E se il risultato della divisione fosse un numero decimale il programma cosa darebbe?

-Naturalmente il risultato di una divisione per zero è indefinito ,ovvero non esiste un numero reale che può essere il risultato di questa divisione. Il programma si interromperebbe senza dare alcun risultato.(utilizzando in questo caso il tipo di dato short int)
Nel caso in cui l'utente volesse dividere 2 numeri decimali il programma potrebbe interrompersi(se si inserisce un numero decimale solo al numeratore) o dare risultati non corretti\imprecisi.
Se avessimo come risultato un numero decimale ,il programma darebbe un risultato non corretto o meglio impreciso.
Vedremo come implementare in modo ottimale questa parte di codice nel punto 4.

4.Nel caso in cui l'utente digitasse all'interno della stringa più di 10 caratteri cosa accadrebbe?E se avessi bisogno di inserire "spazi" il programma come si comporterebbe?Ma soprattutto questo blocco di codice(riferito al void ins_string ()) cosa darà come output?

-Questo blocco di codice non darà alcun output poiché incompleto (manca la funzione printf()) .
Se l'utente digitasse più di 10 caratteri andrebbe in overflow ,mentre nel caso degli spazi il programma darebbe come output solo i caratteri digitati prima dello spazio.
Esempio:
input: "Ciao Alex"
output: "Ciao"
Vedremo come implementare in modo ottimale questa parte di codice nel punto 4.

5.Se l'utente avesse bisogno di fare altre operazioni ?

-Una volta conclusa una delle 3 operazioni ,il programma non darà la possibilità di continuare in automatico .Quindi l'utente potrà fare solo un'operazione .
Vedremo come implementare in modo ottimale questa parte di codice nel punto 4.

PUNTO 3
Individuare eventuali errori di sintassi / logici

1-

```
char scelta = {'\0'};
menu ();
scanf ("%d", &scelta);
switch (scelta)
```

Come possiamo notare questo blocco di codice presenta un errore all'interno della funzione "Scanf" , poiché dichiarando una variabile di tipo "char" ci si aspetta di leggere un carattere e non un intero, in questo modo il programma non darà nulla come output .Quindi bisogna modificare "%d" in"%s".

```
char scelta = {'\0'};
menu ();
scanf ("%s", &scelta);
switch (scelta)
```

2-

```
short int  a,b = 0;
printf ("Inserisci i due numeri da moltiplicare:");
scanf ("%f", &a);
scanf ("%d", &b);
short int prodotto = a * b;
printf ("Il prodotto tra %d e %d e': %d", a,b,prodotto);
```

In questo blocco di codice sia la prima che la seconda funzione " scanf" presentano errori . Nella prima, avendo dichiarato una variabile di tipo "Short int" ,la funzione "scanf" si aspetta di leggere un intero e non un tipo di dato "float",quindi il programma terminerà senza alcun output.
Allo stesso modo la seconda funzione "Scanf" si aspetta di leggere un tipo di dato "Short int" e non un int che generalmente è più grande.In questo modo si otterrà un comportamento imprevisto o risultati errati a causa della differenza di dimensione.

3-

```
printf ("Inserisci il denumeratore:");
    scanf ("%d", &b);

    int divisione = a % b;

    printf ("La divisione tra %d e %d e': %d", a,b,divisione);
```

Nel blocco di codice (sopra) è presente un errore all'interno della variabile "divisione", in questo caso non parleremo di errore di sintassi ma di errori logici. Infatti l'utente si aspetta di poter effettuare una divisione e non un'operazione che dà come output il resto della divisione.Il tutto è facilmente risolvibile ,sostituendo "%" con la "/".

```
int divisione = a / b;
```

4-

```
{
    char stringa[10];
        printf ("Inserisci la stringa:");
        scanf ("%s", &stringa);
}
```

Il primo errore che possiamo riscontrare in questo blocco di codice, di primo acchito ,sta nella mancanza della funzione "printf" .L'utente una volta digitati i caratteri non avrà alcun output ,appunto per l'assenza della funzione stampa.
Bisogna ricordarsi che in "c" quando si legge una stringa tramite la funzione "scanf" , non è necessario utilizzare l'operatore "&" ,prima del nome della stringa, poiché l'array di caratteri già rappresenta un indirizzo di memoria.
Si possono riscontrare ulteriore errori , nel caso in cui l' utente inserisca uno spazio o superi la lunghezza massima consentita di 10 caratteri(""*** stack smashing detected ***: terminated).Andremo a vedere nelle fasi successive come ovviare questi errori.

```
printf ("Inserisci la stringa: ");
scanf ("%s", stringa);
printf("Ecco la Stringa : %s", stringa);
```

PUNTO 4
Proporre una soluzione per ognuno di essi

1-Per affrontare la casistica menzionata in precedenza nella sezione 2.1, ho deciso di incorporare nella struttura di controllo "switch" la clausola "default".

La clausola "default", comunemente definita come caso predefinito, è in genere facoltativa, ma in questa circostanza si dimostra particolarmente vantaggiosa, in quanto viene attivata unicamente quando non è stato individuato alcun "case" corrispondente. In questo modo, siamo in grado di affrontare la problematica derivante dall'inserimento di input non valido da parte dell'utente, garantendo una risposta di errore accurata e offrendo l'opportunità di ripetere la scelta, come sarà ulteriormente delineato nelle prossime fasi del processo.

PRIMA

```
switch (scelta)
{
    case 'A':
    moltiplica();
    break;
    case 'B':
            dividi();
            break;
    case 'C':
            ins_string();
            break;
}
```

DOPO

```
switch (scelta)
{
    case 'A':
    moltiplica();
    break;
    case 'B':
            dividi();
            break;
    case 'C':
            ins_string();
            break;
    default:
       printf("Inserire lettera corretta");
}
```

2-

```c
void moltiplica ()
{
    short int  a,b = 0;
    printf ("Inserisci i due numeri da moltiplicare:");
    scanf ("%hd", &a);
    scanf ("%hd", &b);

    short int prodotto = a * b;

    printf ("Il prodotto tra %hd e %hd e': %hd", a,b,prodotto);
}
```

In questo segmento di codice, è presente il tipo di dato `short int`, il quale è indicato per la rappresentazione di numeri interi di dimensioni limitate, almeno per la maggior parte degli scenari d'uso. La selezione del tipo di dato è variabile in base alle specifiche esigenze dell'utente, e considerando l'incertezza relativa all'utilizzo, abbiamo optato per una soluzione intermedia, utilizzando il tipo di dato `double`. L'uso del tipo `double` offre un ampio intervallo di valori e una maggiore precisione, adatto per una vasta gamma di applicazioni.
È rilevante notare che, in alternativa, avremmo potuto valutare anche l'impiego del tipo di dato `long double`, particolarmente indicato per calcoli finanziari o scientifici che richiedono una precisione estrema. Tuttavia, è importante sottolineare che l'uso di `long double` comporta un maggiore consumo di memoria, pertanto la scelta del tipo di dato deve essere ponderata in base alle esigenze specifiche del programma e alle limitazioni hardware disponibili.

DOPO

```c
void moltiplica ()
{
    double  a,b=0;
    printf ("Inserisci i due numeri da moltiplicare:");
    scanf ("%lf", &a);
    scanf ("%lf", &b);

    double prodotto = a * b;

    printf ("Il prodotto tra %2.1f e %2.1f e': %2.1f", a,b,prodotto);
}
```

3-

```c
void dividi ()
{
        int  a,b = 0;
        printf ("Inserisci il numeratore:");
        scanf ("%d", &a);
    printf ("Inserisci il denumeratore:");
        scanf ("%d", &b);

        int divisione = a / b;

        printf ("La divisione tra %d e %d e': %d", a,b,divisione);
}
```

Casistiche PUNTO 2.3

In questa situazione, l'uso del tipo di dati "int" potrebbe non essere adeguato a causa della perdita di precisione decimale. Per affrontare questa sfida, considererei l'utilizzo del tipo "double" (o, se la memoria è limitata, "float") per garantire una rappresentazione più accurata dei numeri decimali, risolvendo così le problematiche associate ai calcoli decimali.
Successivamente, possiamo affrontare il problema della divisione per zero attraverso l'implementazione di un ciclo "do-while", consentendo all'utente di inserire nuovi valori. In questo modo, assicuriamo un comportamento robusto e un'esperienza utente migliorata.

DOPO

```c
void dividi ()
{
    double divisione;
    double  a,b ;

    do{
        printf ("\nInserisci il numeratore:");
        scanf ("%lf", &a);
        printf ("Inserisci il denominatore:");
        scanf ("%lf", &b);
        if(a==0 || b ==0){
            printf("Diginare numeri differenti da'o'.Grazie\n " );
        }
        divisione = a / b;

 }while(a==0 || b ==0 );
        printf ("La divisione tra %g e %g e': %g", a,b,divisione);
```

4-

```
void ins_string ()
{
     char stringa[10];
          printf ("Inserisci la stringa:");
          scanf ("%s", &stringa);
}
```

Casistiche PUNTO 2.4

1-. Risolveremo la problematica, inerente all'output ,tramite la funzione "printf".

2. Risolveremo la problematica degli spazi tramite "%[^\n]".Questo permetterà di leggere gli spazi alla funzione "scanf" .

3. Per ovviare la problematica dell overflow del buffer si possono percorrere varie strade:

- 1. potremmo manualmente modificare la lunghezza dell'array (esempio da 10 a 100) ma risulterebbe poco efficiente ,o meglio ci sarebbe uno spreco di memoria;
- 2. si potrebbe utilizzare un approccio con allocazione dinamica dell'utente ('malloc') ma avrebbe come svantaggio una gestione della memoria più complessa e una implementazione potenzialmente più lunga

- 3 - si potrebbe utilizzare semplicemente la lunghezza massima da indicare all'interno della funzione (**scanf (" %9[^\n]", stringa);**) ma risulterebbe limitante.

La scelta di adottare un approccio con la dimensione specificata dall'utente, consentendo all'utente di definire direttamente la dimensione dell'array tramite input, risulta essere una soluzione pratica e versatile. Questo approccio si adatta bene a situazioni in cui le esigenze specifiche dell'utente non sono conosciute in anticipo.
Garantendo **Flessibilità**, **Semplicità** e **Riduzione significativa del rischio di Overflow**.

```
void ins_string ()
{
     int lunghezza;
     int valMax= 3000000;
    printf("Scegli la Lunghezza del testo in caratteri (Dim.Max-->%d:)\n",valMax);
    scanf("%d", &lunghezza);
       while(lunghezza>valMax){
                printf("'ERRORE'Digitare un valore più piccolo, grazie\n");
                printf("Scegli la Lunghezza del testo in caratteri (Dim.Max-->%d:)\n",valMax);
                scanf("%d", &lunghezza);
    }
    char stringa[lunghezza];
        printf ("Inserisci la stringa:");
        scanf (" %[^\n]", stringa);
        printf(" Ecco la Stringa : %s", stringa);

}
```

Adesso andremo a ottimizzare il PUNTO 2.5, dando all'utente la possibilità di fare ulteriori operazioni . Usando il ciclo "do-while" e per ogni operazione una funzione "printf"(per far rivedere il menù all'utente) e una "scanf" (per dare all'utente la possibilità di scegliere un'ulteriore operazione)

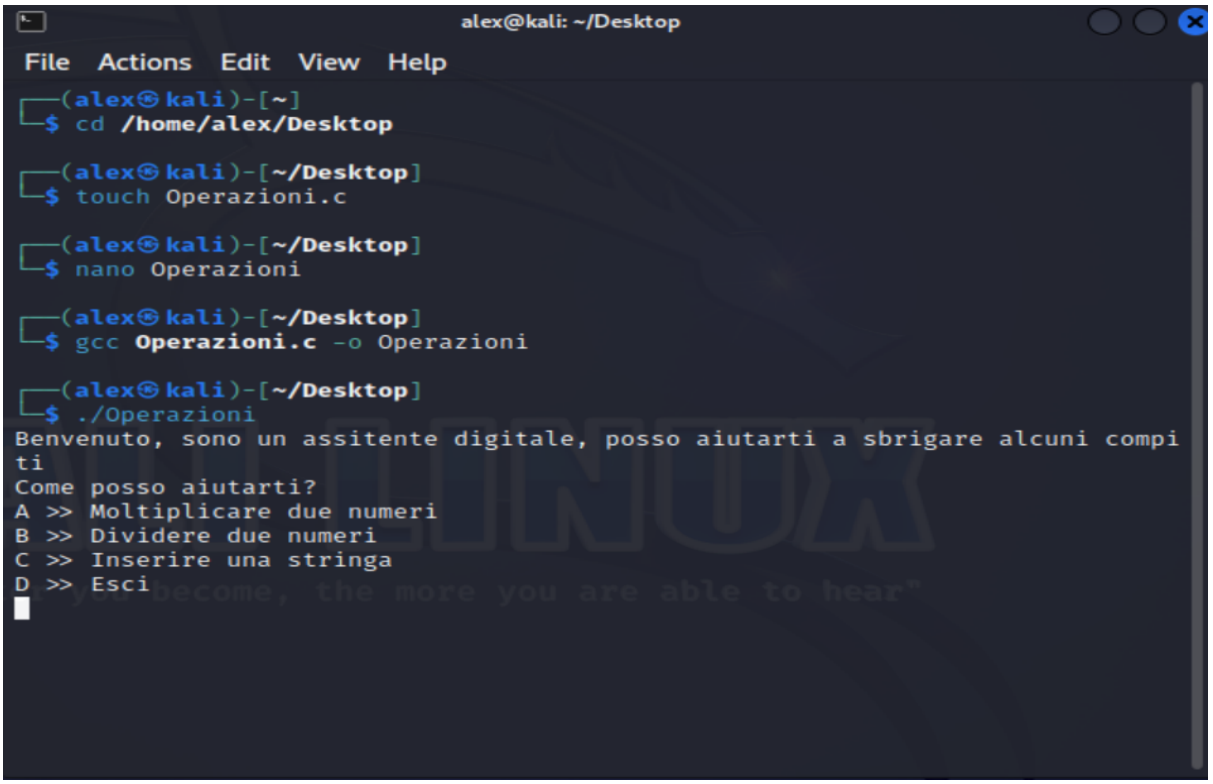
```
int main ()

{

    menu ();

    scanf ("%s", &scelta);
    do{
    switch (scelta)
    {
        case 'A':
                moltiplica();
                break;
        case 'B':
                dividi();
                break;
        case 'C':
                ins_string();
                break;
        case 'D':
                printf("\nCiao ,alla prossima!!!!\n");
        default:
                printf("\nInserire lettera corretta\n");
                scanf ("%s", &scelta);
    }
}while(scelta!='D');
```

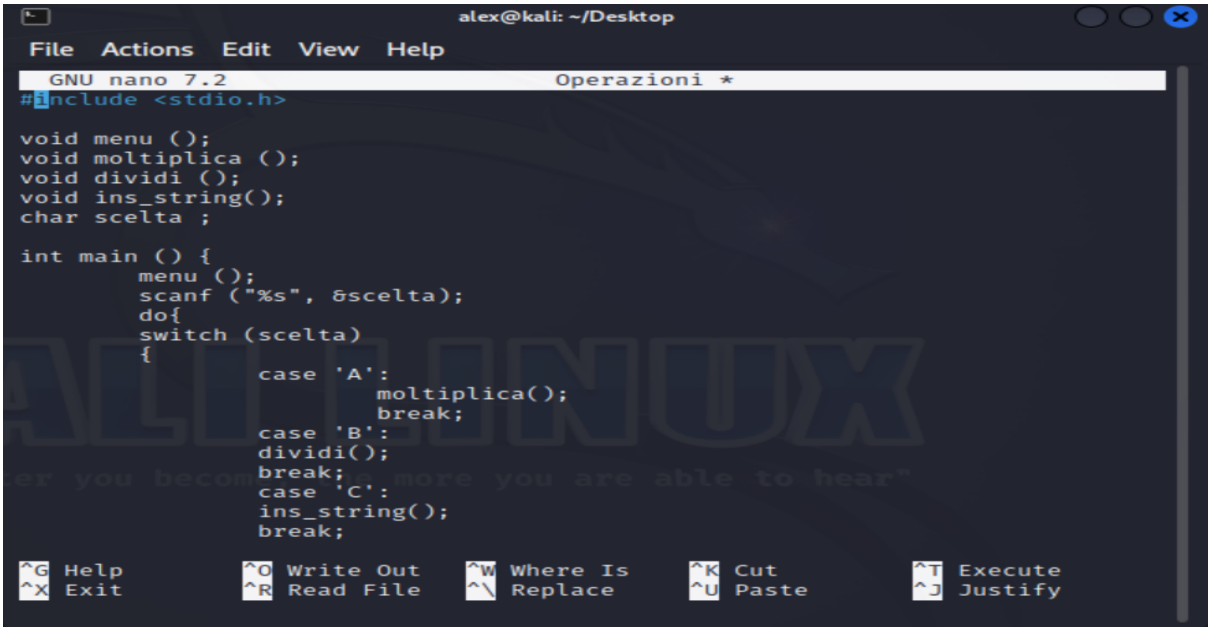
CODICE COMPLETO DOPO LE MODIFICHE APPORTATE

```c
#include <stdio.h>

void menu ();
void moltiplica ();
void dividi ();
void ins_string();
char scelta ;

int main () {
    menu ();
    scanf ("%s", &scelta);
    do{
    switch (scelta)
    {
        case 'A':
                moltiplica();
                break;
        case 'B':
                dividi();
                break;
        case 'C':
                ins_string();
                break;
        case 'D':
                printf("\nCiao ,alla prossima!!!!\n");
        default:
                printf("\nInserire lettera corretta\n");
                scanf ("%s", &scelta);
    }
}while(scelta!='D');

return 0;
}
void menu ()
{
    printf ("Benvenuto, sono un assitente digitale, posso aiutarti a sbrigare alcuni compiti\n");
    printf ("Come posso aiutarti?\n");
    printf ("A >> Moltiplicare due numeri\nB >> Dividere due numeri\nC >> Inserire una stringa\nD >> Esci\n");
}
void moltiplica ()
{
    double  a,b=0;
    printf ("Inserisci i due numeri da moltiplicare:\n");
    scanf ("%lf", &a);
    scanf ("%lf", &b);

    double prodotto = a * b;

    printf ("Il prodotto tra %2.1f e %2.1f e': %2.1f", a,b,prodotto);
    printf("\nVuoi continuare con altre operazioni ?  \nA >> Moltiplicare due numeri\nB >> Dividere due numeri\nC >> Inserire una stringa\nD >> Esci\n " );
    scanf ("%s", &scelta);
}
```

```c
void dividi ()
{
    double divisione;
    double  a,b ;

    do{
        printf ("\nInserisci il numeratore:");
        scanf ("%lf", &a);
        printf ("Inserisci il denominatore:");
        scanf ("%lf", &b);
        if(a==0 || b ==0){
            printf("Diginare numeri differenti da'o'.Grazie\n " );
        }
        divisione = a / b;

 }while(a==0 || b ==0 );
        printf ("La divisione tra %g e %g e': %g", a,b,divisione);
        printf("\nVuoi continuare con altre operazioni ? \nA >> Moltiplicare due numeri\nB >> Dividere due numeri\nC >> Inserire una stringa\nD >> Esci\n " );
    scanf ("%s", &scelta);

}

void ins_string ()
{
     int lunghezza;
     int valMax= 3000000;
    printf("Scegli la Lunghezza del testo in caratteri (Dim.Max-->%d:)\n",valMax);
    scanf("%d", &lunghezza);
       while(lunghezza>valMax){
                printf("'ERRORE'Digitare un valore più piccolo, grazie\n");
                printf("Scegli la Lunghezza del testo in caratteri (Dim.Max-->%d:)\n",valMax);
                scanf("%d", &lunghezza);
}
    char stringa[lunghezza];
        printf ("Inserisci la stringa:");
        scanf (" %[^\n]", stringa);
        printf(" Ecco la Stringa : %s", stringa);
        printf("\nVuoi continuare con altre operazioni ? \nA >> Moltiplicare due numeri\nB >> Dividere due numeri\nC >> Inserire una stringa\nD >> Esci\n " );
        scanf ("%s", &scelta);

}
```

Kali-Linux


alex@kali: ~/Desktop
File Actions Edit View Help
(alex㉿kali)-[~]
$ cd /home/alex/Desktop
(alex㉿kali)-[~/Desktop]
$ touch Operazioni.c
(alex㉿kali)-[~/Desktop]
$ nano Operazioni
(alex㉿kali)-[~/Desktop]
$ gcc Operazioni.c -o Operazioni
(alex㉿kali)-[~/Desktop]
$ ./Operazioni
Benvenuto, sono un assitente digitale, posso aiutarti a sbrigare alcuni compi
ti
Come posso aiutarti?
A >> Moltiplicare due numeri
B >> Dividere due numeri
C >> Inserire una stringa
D >> Esci


alex@kali: ~/Desktop
File Actions Edit View Help
GNU nano 7.2 Operazioni *
#include <stdio.h>
void menu ();
void moltiplica ();
void dividi ();
void ins_string();
char scelta ;
int main () {
menu ();
scanf ("%s", &scelta);
do{
switch (scelta)
{
case 'A':
moltiplica();
break;
case 'B':
dividi();
break;
case 'C':
ins_string();
break;
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify

Ho proceduto creando un nuovo file denominato "Operazioni.c" sul mio Desktop In Kali Linux. Successivamente, ho utilizzato il compilatore GCC per compilare il codice contenuto nel file "Operazioni.c", generando un file eseguibile denominato "Operazioni". Infine, ho eseguito il programma "Operazioni" dal mio terminale ,come si può vedere dagli screen caricati.
Confido che le spiegazioni fornite siano state chiare e dettagliate, coprendo in modo esauriente le diverse fasi del progetto.
Alex doddis