Laboratorio per apprendere le competenze del 21° secolo: percorsi didattici con Scratch per i futuri insegnanti della scuola primaria

Laboratory for learning 21st century skills: training paths for pre-service primary school teachers using Scratch

Lorella Gabriele*, Francesca Bertacchini, Pietro Pantano and Eleonora Bilotta

University of Calabria, Italy, lorella.gabriele@unical.it*, francesca.bertacchini@unical.it, pietro.pantano@unical.it, eleonora.bilotta@unical.it

* corresponding author

HOW TO CITE Gabriele, L., Bertacchini, F., Pantano, P., & Bilotta, E. (2020). Laboratorio per apprendere le competenze del 21° secolo: percorsi didattici con Scratch per i futuri insegnanti della scuola primaria. *Italian Journal of Educational Technology, 28*(1)*,* 20-42. doi: 10.17471/2499-4324/1113

SOMMARIO Il *pensiero computazionale* (CT) è una competenza chiave del 21° secolo considerata fondamentale per tutti gli individui, in quanto favorisce l'acquisizione di nuovi modi di pensare, comunicare ed esprimere idee, nonché per partecipare alla vita civica. Diversi paesi hanno avviato politiche specifiche per l'introduzione del pensiero computazionale e della programmazione nei diversi contesti educativi. Conseguentemente, sono stati avviati corsi di formazione per gli insegnanti in servizio, ma ad oggi è stata prestata solo un'attenzione marginale alla formazione iniziale degli insegnanti. In questo articolo, riportiamo i risultati di un laboratorio di *coding* che ha coinvolto 141 futuri insegnanti. Fornendo precise linee guida e una specifica metodologia didattica, i novizi programmatori hanno sviluppato 40 progetti educativi utilizzando il software Scratch. I risultati hanno mostrato che la maggior parte dei partecipanti ha raggiunto un livello medio-alto di competenze, applicando in modo appropriato sia tecniche di progettazione dei contenuti didattici che di programmazione.

PAROLE CHIAVE Coding; Pensiero Computazionale; Formazione degli Insegnanti in Servizio; Insegnanti in Formazione Iniziale; Acquisizione di Competenze; Costruttivismo; Tecnologie Educative.

ABSTRACT Computational Thinking (CT) is a key skill of the 21st century for everyone as it fosters the acquisition of new ways of thinking, communicating and expressing ideas, as well as new ways of taking part in the social life of a community. Several countries have launched specific policies for the introduction of computational thinking and programming in different educational contexts. Consequently, coding courses for in-service teachers have been run but, to the best of our knowledge, marginal attention has been devoted to the acquisition of coding skills in pre-service teachers. In this paper, we report the results of a coding laboratory that involved 141 future teachers. Following precise guidelines and a

specific training methodology, novice programmers working in groups developed 40 educational projects using Scratch software. Results have showed that most of the participants achieved a medium-high skill level, applying appropriate planning and coding techniques for the exploitation of educational contents.

KEYWORDS Coding; Computational Thinking; In-Service Teachers; Pre-Service Teachers; Acquisition of Skills; Constructivism; Educational Technology.

1. INTRODUZIONE

Attingendo ai principi fondamentali dell'informatica come modello di descrizione del comportamento dei sistemi e anche dell'uomo, nel 2006 Jeanette Wing ha affermato la centralità del pensiero computazionale (CT) nella formazione, facendo riferimento ad una serie di metodi e modelli in grado di facilitare l'apprendimento, la risoluzione dei problemi, la progettazione di sistemi e la comprensione del comportamento umano (Wing, 2006; 2017). Nel 2016 la Commissione Europea, con la "New Skills Agenda"[1], ha invitato esplicitamente gli Stati membri a puntare sulle attività di coding e a integrare l'informatica nell'istruzione. Inoltre, ha evidenziato come l'innovazione nei sistemi di istruzione e di formazione e l'acquisizione delle competenze digitali saranno fondamentali nella nuova strategia per la crescita, l'occupazione e gli investimenti in Europa.

Bocconi, Chioccariello, Dettori, Ferrari e Engelhardt (2016) hanno elaborato un report sul CT in cui forniscono una panoramica completa e un'analisi dei risultati recenti della ricerca e delle iniziative politiche utili per sviluppare il pensiero computazionale tra gli studenti come competenze chiave per il 21° secolo. Gli autori hanno, inoltre, evidenziato le implicazioni a livello politico e pratico, sottolineando, allo stesso tempo, come l'integrazione del pensiero computazionale nell'istruzione obbligatoria debba ancora affrontare diversi problemi e sfide.

In Italia, è con la legge 107 del 2015, il cosiddetto Decreto de "La Buona scuola", che si è posta significativa attenzione al coding e in generale al pensiero computazionale, includendoli tra gli obiettivi educativi (comma 7). Inoltre, con il Piano Nazionale Scuola Digitale vengono promossi corsi per la formazione e/o l'aggiornamento degli insegnanti a differenti livelli, finalizzati a introdurre il pensiero computazionale come competenza trasversale che consenta alle istituzioni educative di avvalersi di forze interne.

Tuttavia, questa riforma lascia fuori alcuni operatori della formazione, come ad esempio i futuri insegnanti. Corsi di laurea centrali, come Scienze della Formazione Primaria (SFP), prevedono solo un percorso di alfabetizzazione digitale, ossia l'uso base di alcuni pacchetti software applicativi di videoscrittura e di calcolo, Internet, il Web e i motori di ricerca, accessibilità e usabilità del Web, Web 2.0 e strumenti. Dalle schede di insegnamento di alcuni atenei[2] emerge, ad esempio, come il digital storytelling e alcuni elementi di coding, attraverso l'utilizzo del programma Scratch, siano da poco tempo parte, seppur marginale, del programma di "Laboratorio di tecnologie didattiche". Partendo da queste considerazioni, ossia la mancanza di pratiche didattiche specifiche ed efficaci per avviare gli studenti di SFP alla programmazione, il nostro lavoro ha lo scopo di:

1) proporre una metodologia per implementare un laboratorio di coding destinato alla formazione iniziale degli insegnanti e finalizzato allo sviluppo del pensiero computazionale;
2) fornire ai futuri insegnanti una metodologia educativa da utilizzare con i propri allievi (insegnare ad insegnare);
3) utilizzare dei metodi di valutazione oggettivi per misurare la comprensione, l'uso e il livel-

[1] https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:52016DC0381

[2] Si vedano, ad esempio, https://corsi.unisa.it/scienze-della-formazione-primaria/didattica/insegnamenti?anno=2017&id=509030 e https://www.unibo.it/it/didattica/insegnamenti/insegnamento/2017/372452

lo dei diversi concetti legati allo sviluppo del pensiero computazionale, allo scopo di capire quali competenze rafforzare e quali sono già state acquisite.

Abbiamo attivato un laboratorio di coding destinato agli insegnanti in formazione iniziale (studenti universitari) basato sull'utilizzo di Scratch per promuovere l'acquisizione di competenze legate al pensiero computazionale, attraverso un approccio costruttivista all'apprendimento (Papert, 1984, 1986; Piaget, 1967).

La ricerca è stata ideata e portata avanti presso il Laboratorio di Psicologia Cognitiva, Dipartimento di Fisica dell'Università della Calabria, i cui ricercatori da anni lavorano sull'applicazione di tecnologie didattiche avanzate in diversi contesti (Bertacchini et al., 2012; Bertacchini, Bilotta, & Pantano, 2017; Bilotta et al., 2007; Gabriele, Marocco, Bertacchini, Pantano, & Bilotta, 2017; Giglio et al., 2015; Vaca-Cárdenas et al., 2015).

1.1. *Pensiero computazionale e programmazione*

"*Il pensiero computazionale è il processo mentale alla base della formulazione dei problemi e delle loro soluzioni tali da poter essere rappresentate in una forma che possa essere efficacemente eseguita da un agente* (un computer, un essere umano o entrambi)" (Wing, 2011, p. 20). Nel 2017, la Computer Teachers Association (CSTA)[3] ha integrato la definizione di CT data da Jeanette Wing nel 2011, evidenziando gli aspetti di risoluzione dei problemi, nonché l'astrazione, l'automazione e l'analisi come elementi distintivi del CT: "*Riteniamo che il pensiero computazionale sia una metodologia di risoluzione dei problemi che dall'ambito dell'informatica può essere applicata a tutte le discipline, fornendo uno strumento distinto di analisi e sviluppo di soluzioni a problemi che possono essere risolti a livello computazionale*". Di conseguenza, le competenze relative al CT sono un prerequisito necessario nella società digitale. Si tratta di elementi fondamentali in grado di favorire l'acquisizione di un nuovo modo di pensare (Curzon, Dorling, Ng, Selby, & Woollard, 2014; Lye & Koh, 2014): riflettere su un problema, codificarlo, progettare soluzioni, analizzarle e applicarle ad altri contesti (Barr, Harrison, & Conery, 2016; Faraco & Gabriele, 2007). A livello internazionale, la riflessione sull'importanza del CT ha generato un vivace dibattito tale da rimettere in discussione l'impianto del curriculum scolastico. Diversi Paesi hanno introdotto sia il pensiero computazionale che l'informatica nei percorsi formativi a diversi livelli e in diversi comparti delle scuole. Come evidenziato da Baytak e Land (2011), la formazione iniziale universitaria degli insegnanti basata sul CT è fondamentale al fine di avviare dei percorsi formativi nella scuola primaria affinché i bambini possano implementare efficacemente giochi per computer utilizzando software adeguati al loro livello di esperienza. In diversi studi viene, ad esempio, evidenziata la facilità d'uso di Scratch per avviare i bambini, novizi e insegnanti alle attività di programmazione. Bell, Frey e Vasserman (2014) descrivono l'esperienza maturata da quattro futuri insegnanti specializzati in arte e musica, durante un campo estivo incentrato sulla programmazione. I futuri insegnanti sono stati supportati nell'esplorare le modalità con cui potevano incorporare gli attuali ambienti di programmazione user friendly (in questo caso, Scratch) nel loro curriculum e guidati a incorporare la propria esperienza musicale e artistica nelle attività di programmazione. Quan (2015) spiega le modalità con cui gli studenti e gli insegnanti in formazione iniziale hanno sviluppato dei progetti in Scratch integrando concetti relativi all'insegnamento/apprendimento della seconda lingua per studenti delle scuole elementari, medie o superiori. An e Lee (2014) hanno rilevato gli sforzi di molti Paesi di integrare il pensiero computazionale nei programmi di studio regolari e come questo implichi in primis formare sia gli insegnanti in servizio che in formazione iniziale, istituendo corsi di formazione specifici e di aggiornamento. Infine, Federici, Gola, Brau e Zuncheddu (2015) hanno evidenziato come l'approccio presentato nel loro lavoro di ricerca consenta agli insegnanti di avviare facilmente le attività di coding senza essere

[3] https://www.doe.k12.de.us/cms/lib/DE01922744/Centricity/Domain/176/CSTA%20Computer%20Science%20Standards%20Revised%202017.pdf

necessariamente esperti in informatica.
Un altro aspetto importante è come Scratch abbia effetti postivi sul senso di autostima, sulla motivazione ad apprendere e per sviluppare la comprensione di se stessi, delle relazioni con gli altri e del mondo tecnologico. È ciò che emerge, ad esempio, dal lavoro di Brennan e Resnick (2012), i quali nel delineare il quadro di riferimento del pensiero computazionale ne evidenziano tre dimensioni: i concetti computazionali (i concetti con cui i novizi programmatori si relazionano nella fase di scrittura del codice, es. iterazione, parallelismo, ecc.), le pratiche computazionali (sviluppare un progetto non è un processo sequenziale, ma un processo adattivo in cui il piano iniziale potrebbe cambiare in risposta all'avvicinarsi di una soluzione a piccoli passi, ad esempio il debugging dei progetti o il remix di altri) e le prospettive computazionali (lavorando con Scratch si verifica un cambiamento di prospettiva nella comprensione di se stessi, delle proprie relazioni con gli altri e del mondo tecnologico circostante. Scratch permette di fruire dei progetti già sviluppati, quindi essere *consumer*, oppure di sviluppare e ideare dei propri progetti, quindi essere *producer*, oppure *prosumer*, ovvero fruire e sviluppare nuovi progetti allo stesso tempo). Anche Moreno e Robles (2016) affermano che le attività di coding non sono fini a se stesse, ma strumenti utili per sviluppare altre competenze, per migliorare i risultati dell'apprendimento e la motivazione di chi apprende. Tuttavia, non è solo importante sperimentare percorsi di coding più o meno strutturati, ma padroneggiare anche i metodi di valutazione per testare il livello di CT acquisito. Repenning et al. (2015) affermano che la capacità di rilevare automaticamente i concetti di CT analizzando i progetti sviluppati dagli studenti può dare agli educatori e agli studenti una visione più ampia, fornendo informazioni utili su quali concetti sono stati compresi e quali devono ancora essere appresi.
A tal proposito, in letteratura sono stati proposti metodi diversi. Per esempio, Denner, Werner e Ortiz (2012) hanno analizzato le applicazioni sviluppate dagli studenti in Stagecast Creator per mappare la progressione dei concetti del pensiero computazionale. Secondo questi autori, lo schema di codifica utilizzato può essere adattato a qualsiasi ambiente di programmazione destinato alle popolazioni più giovani. Sulla base di ciò, Wilson, Haney e Connolly (2013) hanno adattato lo schema originariamente ideato per valutare i videogiochi da Denner et al. (2012) per i progetti Scratch. Con la Computational Thinking Pattern Analysis (CTPA), Repenning et al. (2015) hanno sviluppato uno strumento utile per misurare le capacità di apprendimento degli studenti e rappresentare semanticamente i loro risultati attraverso un'analisi fenomenologica in tempo reale. Simile è il modello Progression of Early Computational Thinking di Seiter e Foreman (2013). Infine, "Dr. Scratch", che è stato sviluppato da Moreno e Robles (2016), effettua una valutazione automatica dei progetti sviluppati dagli utenti.

1.2. *Scratch: un linguaggio di programmazione visuale semplice e potente*

Negli ultimi anni sono stati sviluppati diversi linguaggi di programmazione visuali, come Agentsheets, Lego Mindstorms NXT e Scratch, dove l'uso di una sintassi scritta è stato sostituito da una manipolazione grafica degli elementi, superando la concezione di programmazione riservata a un numero ristretto di esperti in modo da favorirne l'approccio anche ai programmatori principianti.
In particolare, Scratch[4] è stato ideato nel 2008 dal Lifelong Kindergarten Group presso il MIT Media Lab (Maloney, Peppler, Kafai, Resnick, & Rusk, 2008) e consente ai giovani programmatori di creare media interattivi, come ad esempio giochi, storie e simulazioni, basandosi sulla logica dello storytelling. L'attuale versione 3.0, compatibile con le precedenti, è stata sviluppata in HTML/Javascript e può essere usata online o sui tablet; ha un'interfaccia aggiornata, nuovi mattoncini e funzionalità.
Secondo Maloney, Resnick, Rusk, Silverman e Eastmond (2010), una delle caratteristiche fondamentali di

[4] https://scratch.mit.edu/

Scratch è la "tinkerability", secondo cui l'utente può avere un feedback immediato sul funzionamento dei singoli blocchi (o sequenze di blocchi) poiché questi sono oggetti reattivi che vengono eseguiti in tempo reale. Quando si clicca su un blocco, anche quando si trova ancora nel menu di scelta, senza trascinarlo nell'area degli script (dove si collegano i blocchi in script), si visualizza sullo schermo del computer il funzionamento. Inoltre, i parametri o la sequenza stessa dei blocchi possono essere facilmente modificati anche mentre questi vengono eseguiti. Scratch è facile da usare, poiché piccoli pezzi di codice (script) possono essere assemblati insieme come mattoncini e combinati in unità più grandi, a loro volta interconnesse fino a realizzare un progetto completo. Questo approccio alla programmazione è anche detto bottom-up, dal basso verso l'alto.

I progetti sviluppati con Scratch sono composti da oggetti detti *sprite*; ogni *sprite* può essere controllato da uno o più script e può cambiare costume, ossia aspetto. Lo *stage* rappresenta lo sfondo, ossia il palco dove si svolgono le azioni. Tra le categorie di blocchi fondamentali vi sono: dichiarazioni, espressioni booleane, condizioni, cicli, variabili ed eventi che funzionano solo se sono sintatticamente impostati. In Scratch l'utente può effettuare il remix dei progetti, ossia può riutilizzare un progetto già disponibile nella comunità di Scratch modificando il codice in parte e dando i crediti allo sviluppatore originale del progetto (Monroy-Hernández, 2012).

Nella letteratura internazionale viene evidenziato come il software Scratch può essere considerato un utile supporto per acquisire competenze legate allo sviluppo del pensiero computazionale (Maloney et al., 2008; Maloney Resnick, Rusk, Silverman, & Eastmond, 2010), fornendo a studenti e insegnanti la possibilità di imparare facendo (Tan & Kim, 2015; Vaca Cárdenas et al., 2016). L'uso delle tecnologie nel contesto educativo fa riferimento alla prospettiva costruttivista (Papert, 1984; 1986; Piaget, 1967), secondo cui lo studente ha un ruolo attivo nell'acquisizione della conoscenza per cui impara meglio e di più facendo (learning by doing) e dalle esperienze reali. In questo processo, gli insegnanti assumono il ruolo di guida. Nella visione costruttivista, l'apprendimento avviene quando gli studenti sono attivamente coinvolti in un processo di costruzione del significato e della conoscenza piuttosto che quando ricevono passivamente informazioni. L'insegnamento costruttivista promuove il pensiero critico e crea pensatori motivati e indipendenti. Scratch incorpora tutti i principi del Costruttivismo poiché permette, da un lato, di esplorare facilmente e intuitivamente le funzionalità dei blocchi di codice e, dall'altro, favorisce le attività di gruppo attraverso le quali creare un clima favorevole altamente motivante improntato allo scambio di idee e soluzioni creative. Numerose ricerche confermano che le attività educative con Scratch consentono di produrre facilmente contenuti digitali (Wilson & Moffat, 2010), favorire la creatività degli studenti (Kobsiripat, 2015; Korkmaz, 2016) a diversi livelli scolastici (Armoni, Meerbaum-Salant, & Ben-Ari, 2015; Chiu, 2014; Foerster, 2016).

2. LA RICERCA

La ricerca presentata in questo articolo si configura come un caso di studio (Yin, 1994) ideato con l'obiettivo di condurre un'indagine qualitativa sull'efficacia di un approccio al coding per la formazione dei futuri insegnanti.

2.1. *Scopo dello studio*

Obiettivo della ricerca è investigare i concetti computazionali acquisiti e il relativo livello di competenza degli insegnanti in formazione iniziale (studenti universitari) dopo il laboratorio di coding. Durante il laboratorio è stato introdotto il software Scratch e sono state fornite precise linee guide per sviluppare progetti utili a favorire l'apprendimento di contenuti educativi per un determinato target di utenti. I concetti computazionali e il relativo livello di competenza acquisito sono stati valutati con la metodologia di Wilson et

al. (2013) e con il software Dr. Scratch sviluppato da Moreno-León, Robles, e Román-González (2015). Le specifiche domande a cui la presente ricerca ha voluto dare risposta sono le seguenti:

1) Quali sono i concetti computazionali acquisiti dagli insegnanti in formazione iniziale?
2) Qual è il livello di competenza acquisito?

2.2. *Campione*

Lo studio ha coinvolto 141 insegnanti studenti (F=128; M=13) iscritti al corso di Laurea Magistrale in Scienze della Formazione Primaria, presso l'Università della Calabria nell'a.a. 2016/2017. I partecipanti (età compresa tra i 18 e i 40 anni; M=26; SD=5,93) hanno liberamente formato 40 gruppi (5 gruppi=due partecipanti; 13 gruppi=tre partecipanti; 18 gruppi=quattro partecipanti; 4 gruppi=cinque partecipanti).
Al campione di ricerca è stato somministrato un questionario demografico iniziale per rilevare il background scolastico di provenienza e il livello di alfabetizzazione digitale e di familiarità con le tecnologie, attraverso un'autovalutazione (vedere 8. Appendice). I risultati hanno mostrato un'alta percentuale di partecipanti con una formazione umanistica (Tabella 1).

SCUOLA SECONDARIA ITALIANA	N ° di partecipanti
Diploma di scuola superiore in studi classici	79 (56%)
Diploma di scuola superiore in studi scientifici	10 (7%)
Diploma di scuola superiore in lingue straniere	11 (8%)
Diploma di scuola superiore in studi pedagogici	41 (29%)
TOTALE	141

Tabella 1. Background scolastico dei partecipanti.

Il 93% del campione ha auto-valutato "insufficiente" o "appena sufficiente" (8. Appendice, domanda n. 9) il proprio livello di alfabetizzazione digitale.

2.3. *Strutturazione del piano formativo*

La ricerca ha avuto una durata complessiva di dieci mesi. Le attività includevano la definizione del piano di ricerca, l'organizzazione del materiale didattico, l'avvio di una ricerca pilota che ha coinvolto un piccolo campione di partecipanti al fine di testare il materiale organizzato ed eventualmente migliorarlo. Infine, sono state avviate le attività laboratoriali di coding con Scratch 2.0 che hanno avuto una durata complessiva di otto settimane. I partecipanti hanno seguito le lezioni teoriche sulla programmazione durante quattro lezioni in presenza, di due ore ciascuna. Per superare il corso è stato assegnato loro il compito di ideare e sviluppare un progetto in Scratch (il dettaglio delle attività è illustrato nella Tabella 2).

LEZIONI	MATERIALE DIDATTICO
Settimana 1 Panoramica generale sulle attività di laboratorio: regole e impostazione, spiegazione dell'obiettivo di ricerca e introduzione di Scratch Introduzione su Scratch: funzionalità e progetti in Scratch 2.0 Linee guida su come creare un progetto	Presentazioni PowerPoint: Introduzione alla programmazione Concetti generali: cos'è un algoritmo? Programmazione strutturata Scratch e le sue funzionalità Esempio di progetto sviluppato con Scratch Video: Analisi passo passo di un progetto realizzato con Scratch
Settimana 2 Focus group	Agli studenti sono state fornite istruzioni su come lavorare in gruppo e su come sviluppare un progetto in Scratch
Settimana 3 Coding	Agli studenti sono state date istruzioni su come creare un progetto didattico (gioco, storia, ecc.). Gli esercitatori hanno spiegato loro come creare il report e il manuale dell'utente del progetto sviluppato in Scratch.
Settimana 4-8 Ideazione e realizzazione del progetto didattico	Gli studenti hanno lavorato in gruppo con il supporto degli esercitatori.

Tabella 2. Strutturazione del laboratorio.

Ad ogni gruppo di lavoro sono state fornite le seguenti linee guida metodologiche da seguire per sviluppare il progetto:

1) scegliere un argomento didattico (es. matematica, scienze, lingua straniera);
4) sviluppare un progetto user-centred, ossia centrato sull'utente, prendendo in considerazione i prerequisiti e i requisiti dell'utente;
5) utilizzare uno storyboard per ideare il progetto;
6) implementare il progetto con un alto livello di interattività utilizzando, ad esempio, la lavagna interattiva (LIM) o la webcam;
7) testare il progetto Scratch con l'utente finale.

Da un punto di vista concettuale, le attività di sviluppo del progetto si possono quindi riassumere in tre fasi: 1) fase decisionale; 2) fase di implementazione; 3) fase di follow-up.

Nella *fase decisionale*, un brainstorming ha coinvolto tutti i gruppi permettendo di esplorare diverse idee e definire il focus del progetto, la sua fattibilità e i principali obiettivi didattico-educativi. Le linee guida e i suggerimenti forniti agli studenti insegnanti miravano a trasmettere l'idea che il progetto dovesse rispettare il criterio user-centred, ossia realmente centrato sulle capacità/abilità e conoscenze dell'utente finale. Facendo riferimento alle Indicazioni nazionali per il curriculum[5], considerate quadro di riferimento per la

[5] http://www.indicazioninazionali.it/wp-content/uploads/2018/08/Indicazioni_Annali_Definitivo.pdf

progettazione curricolare, sono stati individuati i prerequisiti dell'utente finale a cui il progetto è destinato in correlazione all'età, alle conoscenze, alla classe frequentata e all'argomento che gli sviluppatori hanno voluto affrontare. Nella *fase di implementazione*, i partecipanti hanno sviluppato i progetti utilizzando Scratch 2.0. I progetti sono stati poi testati coinvolgendo gli utenti finali, ossia i bambini. Infine, nella *fase di follow-up* gli studenti hanno preparato una relazione e un handbook (guida) per l'utente in cui sono spiegate le funzionalità del progetto.

Durante ogni attività, in caso di difficoltà o dubbi, gli studenti potevano chiedere chiarimenti ai ricercatori coinvolti nelle attività di laboratorio. In questo modo avrebbero avuto la certezza di ricevere il supporto, il feedback o l'incoraggiamento necessario per proseguire con il lavoro assegnato. Di contro, il rischio sarebbe stato che gli studenti fossero più fiduciosi nei suggerimenti dei ricercatori che nelle loro capacità. Ecco perché si è scelto di guidarli a trovare nuove soluzioni e non fornire mai solo la risposta cercata.

Dopo l'esperienza del laboratorio agli studenti è stato chiesto di rispondere alla seguente domanda: Come percepisci le tue conoscenze di programmazione dopo l'esperienza di laboratorio? (opzioni di risposta: "Nessuna conoscenza", "Principiante", "Esperto", accompagnata da un'opinione personale sulle attività svolte).

Inoltre, è stato chiesto di autovalutare il senso di autoefficacia percepito in relazione alle competenze acquisite dopo le attività di coding (Tabella 3) e utilizzando una scala Likert a quattro punti (da "Molto buona" a "Insufficiente").

Valutazione dell'autoefficacia
Q1 - Soddisfazione personale
Q2 - Sviluppo personale
Q3 - Conoscenze pedagogiche
Q4 - Alta capacità di progettare materiale educativo dopo l'acquisizione delle competenze CT
Q5 - Miglioramento delle competenze nell'insegnamento dei concetti CT

Tabella 3. Test di autovalutazione somministrato alla fine del laboratorio.

2.4. *Metodo di codifica e analisi dei dati*

I report elaborati da ciascun gruppo sono stati utilizzati per verificare le modalità di lavoro adottate dagli studenti per sviluppare il progetto e quindi per verificare se avessero:

1) chiaramente definito l'obiettivo didattico del progetto;
8) preso in considerazione le caratteristiche (età, classe frequentata, prerequisiti) del target a cui il progetto si rivolge;
9) elaborato un chiaro manuale che ne descrivesse il funzionamento.

Inoltre, i file sorgente dei progetti sono stati analizzati per rilevare: 1) i concetti computazionali acquisiti e 2) il relativo livello di competenza.

Per quanto riguarda il primo punto, si è fatto riferimento alla metodologia sviluppata da Denner et al. (2012) e adattata da Wilson et al. (2013) per valutare i progetti. Gli autori organizzano lo schema di codifica in tre macro categorie – concetti di programmazione, organizzazione del codice e usabilità – definite a loro volta da specifiche sottocategorie. La metodologia prevede l'attribuzione di 1 punto per ogni categoria presente nel progetto e l'attribuzione di 0 punti se la categoria è assente.

In Tabella 4 viene presentata la rubrica di valutazione utilizzata per valutare i progetti. Allo stesso modo di Denner et al. (2012), è stata calcolata la frequenza percentuale relativamente ai "Concetti di programma-

zione", "Organizzazione del codice" e "Usabilità" per individuare in quanti progetti fossero presenti tali concetti.

CATEGORIA	**DEFINIZIONE**
CONCETTI DI PROGRAMMAZIONE	
Sequenza	I blocchi sono in ordine sistematico e il programma viene eseguito correttamente
Interazione dell'utente (es. input della tastiera)	Uso di blocchi come "chiedere" e "attendere" che richiedono agli utenti di digitare una risposta o cliccare un tasto specifico
Iterazione (cicli)	Uso di cicli "per sempre" e "ripeti" per creare iterazioni
Variabili	Dichiarazione e uso di variabili
Istruzioni condizionali	Uso di "se", "se altrimenti" per verificare le condizioni
Liste (array)	Uso di liste per memorizzare e accedere a elenchi di stringhe e numeri
Coordinamento e sincronizzazione (Parallelismo)	Uso di blocchi come "attenti", "invia tutti" per coordinare le azioni di più Sprite
Numeri casuali	Usato per selezionare numeri casuali all'interno di un determinato intervallo
Logica booleana	Uso di AND, OR, NOT, Vero o falso
ORGANIZZAZIONE DEL CODICE	
Blocchi estranei	Sono presenti blocchi che non sono stati inizializzati quando viene eseguito il programma
Nomi Sprite	Gli Sprite sono stati rinominati o vengono utilizzati i nomi predefiniti
Nomi di variabili	Uso di nomi di default per le variabili oppure attribuzione di nomi significativi
USABILITÀ	
Funzionalità	Il programma funziona?
Personalizzazione degli Sprite	È stato utilizzato uno Sprite predefinito disponibile nella libreria del programma o è stato personalizzato dallo sviluppatore?
Personalizzazione dello sfondo	È stato utilizzato uno sfondo predefinito disponibile nella libreria del programma o è stato personalizzato dallo sviluppatore?
Istruzioni chiare	Lo studente ha chiaramente definito come dovrebbe funzionare il gioco?
Originalità del progetto	Gli studenti hanno sviluppato il proprio progetto in base all'obiettivo iniziale?

Tabella 4. Rubrica per la valutazione dei progetti di Scratch.

Per quanto riguarda il secondo punto, al fine di individuare il livello di competenza acquisito, abbiamo utilizzato Dr. Scratch (Moreno-León, Robles, & Román-González, 2015), un software che valuta automaticamente la presenza di sette parametri (Tabella 5):

Concetto computazionale	LIVELLO DI COMPETENZA			
	Nulla (0)	Base (1 punto)	Intermedio (2 punti)	Avanzato (3 punti)
Astrazione e decomposizione dei problemi	-	Più di uno script e Sprite	Definizione di blocchi	Uso di cloni
Parallelismo	-	Due script collegati alla bandierina verde	Due script sul tasto premuto, su Sprite cliccato sullo stesso Sprite	Due script con "quando ricevo il messaggio", creare clone, due script con strutture condizionali. Due script "quando cambia lo sfondo"
Strutture condizionali	-	Se	Se altrimenti	Operatori logici
Sincronizzazione	-	Aspettare	Trasmetti, quando ricevo il messaggio, interrompi tutto, ferma programma, interrompi	Attendi fino a quando, quando lo sfondo cambia, trasmetti e attendi
Strutture di controllo di flusso	-	Sequenza di blocchi	Ripeti, per sempre	Ripeti fino a
Interattività definita dall'utente	-	Bandierina verde	Tasto premuto, Sprite cliccato, chiedi e attendi, click del mouse	Quando %s è >% s, video, audio
Rappresentazione dei dati	-	Modifica delle proprietà degli Sprite	Operazioni su variabili	Operazioni su liste

Tabella 5. Livello di competenza per ciascun concetto computazionale.

Dal momento che Dr. Scratch assegna un punteggio da 1 a 3 a ciascuno dei parametri, e in base al punteggio totale, il progetto potrà così avere un livello "base", "intermedio" oppure "avanzato". È stata, inoltre, calcolata la frequenza percentuale per i diversi concetti computazionali per verificare in quanti progetti Scratch fossero presenti.

I progetti finali (Report e Progetti in Scratch) presentano nell'insieme piccolissime differenze l'uno dall'altro. I criteri di valutazione sono riassunti nella Tabella 6.

Valutazione	Criteri
Eccellente	Progetto in Scratch→ Il progetto è funzionante, molto creativo e dimostra chiaramente idee uniche. Creazione di sfondi e personaggi ad hoc, registrazione di audio e integrazione di musiche. Gli script funzionano tutti, sono molto ben progettati. Lo studente ha un'ottima conoscenza degli script. Sono tutti nominati correttamente e molto ben progettati. REPORT→ Dal report emergono chiaramente che sono state seguite le linee guida fornite. Sono partiti dalla scelta dell'argomento, tenendo in considerazione età del target, definiti i prerequisiti. Emerge uno schema di progettazione dell'applicazione sviluppata in Scratch. Hanno indicato e motivato i vari cambiamenti e le integrazioni effettuate al progetto. Emerge un eccellente lavoro di gruppo. Test del progetto con utente finale.
Molto Buono	Progetto in Scratch → Funzionante. Lavoro creativo e design unico, facile da usare e facile da capire. Tutti gli script funzionano, non sono presenti comandi non collegati. REPORT→ Dal report emergono chiaramente che sono state seguite le linee guida fornite. Sono partiti dalla scelta dell'argomento, tenendo in considerazione età del target, definiti i prerequisiti. Emerge uno schema di progettazione dell'applicazione sviluppata in Scratch. Hanno indicato e motivato i vari cambiamenti e le integrazioni effettuate al progetto. Hanno personalizzato sfondi, personaggi, registrato audio. Emerge un lavoro di gruppo molto buono improntato alla collaborazione e alla cooperazione. Test del progetto con utente finale.
Buono	Progetto in Scratch → Funzionante, emergono idee creative e uniche. Non sono presenti comandi non collegati. Gli script sono stati tutti nominati correttamente. REPORT→ Sono state seguite le linee guida fornite. Sono partiti dalla scelta dell'argomento, tenendo in considerazione età dell'utente finale e definito i prerequisiti. Emerge uno schema di progettazione dell'applicazione sviluppata in Scratch. Non sempre hanno personalizzato sfondi, personaggi, registrato audio. Emerge un buon lavoro di gruppo. Test del progetto con utente finale.
Discreto	Progetto in Scratch → Funzionante, sono presenti comandi non collegati REPORT→ Dal report emergono chiaramente che sono state seguite le linee guida fornite. Dal report emergono chiaramente che sono state seguite le linee guida fornite. Sono partiti dalla scelta dell'argomento, tenendo in considerazione età del target, definiti i prerequisiti. Emerge uno schema di progettazione dell'applicazione sviluppata in Scratch. Hanno indicato e motivato i vari cambiamenti e le integrazioni eseguite nel progetto. Non hanno personalizzato tutti gli sfondi, personaggi. Hanno preferito uso di testo alle registrazioni audio. Il lavoro di gruppo è discretamente apprezzabile. Test del progetto con utente finale.
Sufficiente	Progetto in Scratch → Funzionante, sono presenti comandi non collegati. È stato effettuato il remix di alcuni progetti. Alcuni sfondi e sprite non sono stati rinominati. REPORT→ L'applicazione in Scratch è stata progettata per un target di età ma non sono stati definiti i prerequisiti. Emerge uno schema di progettazione dell'applicazione sviluppata in Scratch. Hanno indicato e motivato i vari cambiamenti e le integrazioni eseguite nel progetto. Non sempre emerge un lavoro coeso di gruppo. Test del progetto con utente finale.

Tabella 6. Rubrica di valutazione dei progetti finali.

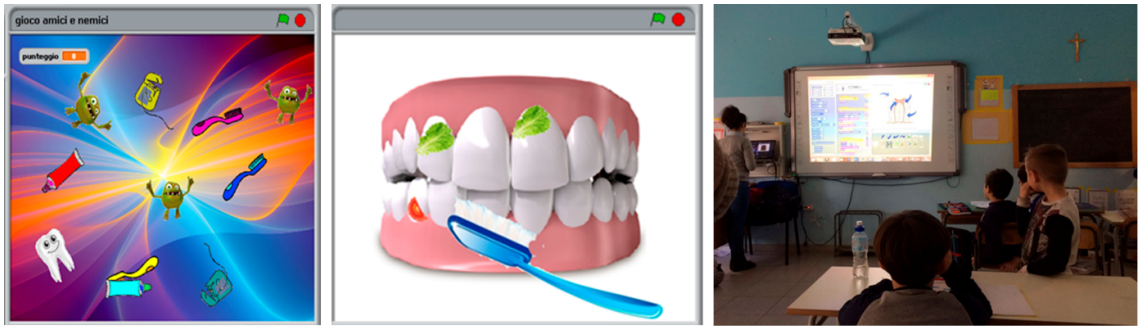
3. RISULTATI

I 141 studenti insegnanti hanno sviluppato 40 progetti su differenti tematiche studiate nella scuola elementare (Tabella 7), tenendo conto dell'età, della classe frequentata e del bagaglio di conoscenze dell'utente finale.

Numero di progetti	Argomento
4	Riconoscere e nominare i colori
6	Matematica
5	L'alfabeto (italiano e inglese)
12	Concetti base di Inglese
1	Giorni della settimana
3	Stagioni e frutti
1	Giochi di attenzione visiva e velocità di reazione
3	Geografia
1	Musica
1	Sport
1	Formazione scolastica
1	Emozioni
1	Scienza

Tabella 7. Argomenti dei progetti.

A titolo di esempio, nella Figura 1 è riportata l'immagine del progetto "Lava i tuoi denti". Nell'immagine a) vengono mostrati ai bambini gli amici e i nemici dei denti. L'immagine b) mostra un gioco interattivo da utilizzare con la LIM (i bambini possono pulire i denti maneggiando lo spazzolino sulla LIM). L'immagine c) mostra i bambini durante la fase di test del progetto in classe.



[a] [b] [c]

Figura 1. Il progetto "Lava i tuoi denti".

La Figura 2 mostra uno degli script del progetto "La ruota della fortuna". Per sviluppare il progetto sono

stati utilizzati comandi di Controllo (ripeti fino a quando), il concetto "Rappresentazione dei dati", Astrazione (Uso di più script), Interattività (per sempre), Coordinamento e Sincronizzazione (invia a tutti, quando ricevo). Gli studenti hanno duplicato due script, utilizzato variabili e definito due attributi per gli sprite.

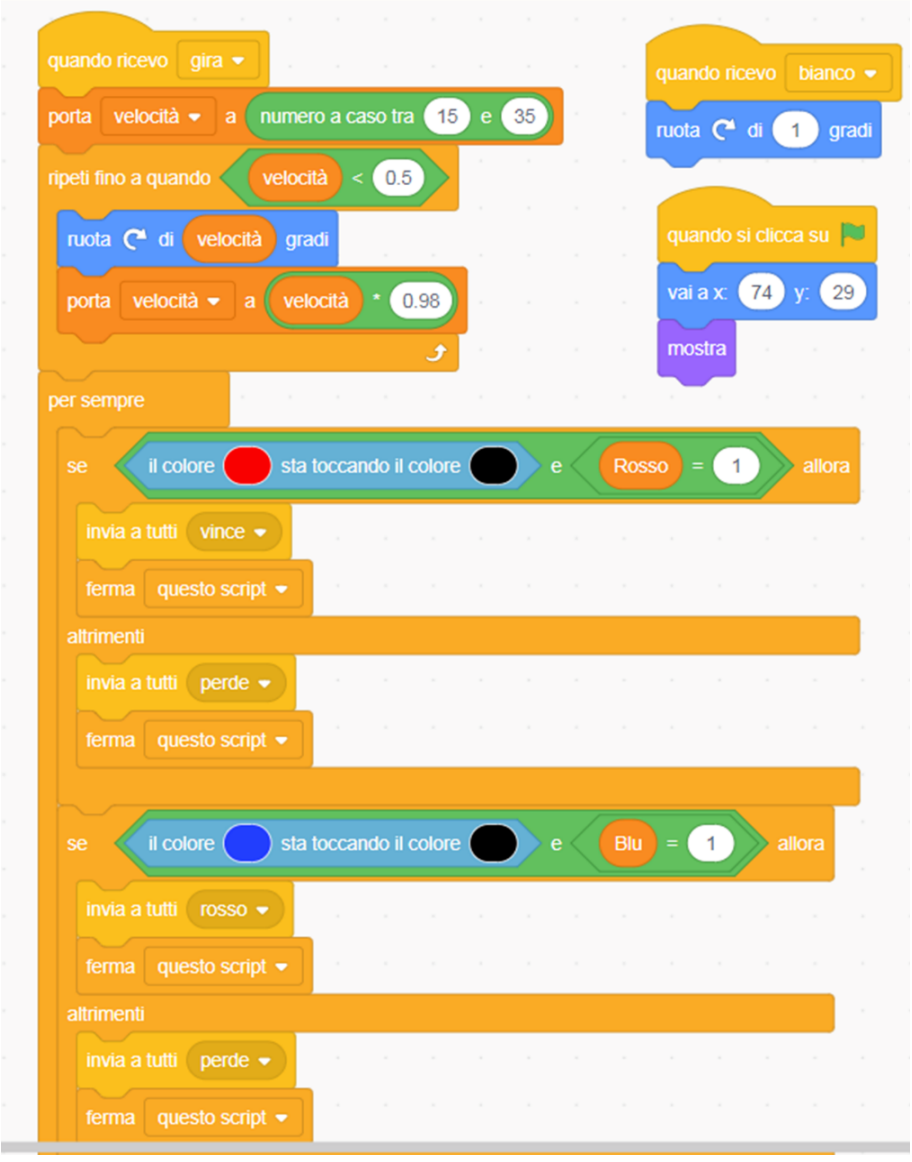


Figura 2. Parte dello script del progetto "La ruota della fortuna".

Analizzando la relazione e la guida all'uso del progetto sviluppato con Scratch che ogni gruppo ha consegnato alla fine del laboratorio, abbiamo raccolto evidenze utili sul processo di ideazione e sviluppo del progetto. Abbiamo scoperto, ad esempio, che tutti i team hanno iniziato il lavoro identificando chiaramente il "Problema da risolvere" (l'argomento che ogni progetto approfondiva), mentre il 97,5% ha dettagliato l'obiettivo del progetto. Tutti i gruppi di lavoro hanno preso in considerazione i requisiti dell'utente. Hanno raccolto immagini, suoni, registrazioni vocali, immagini personalizzate e tutti i materiali digitali utili per assemblare il progetto finale. L'80% dei team ha sviluppato un progetto descrivendo chiaramente le moda-

lità di funzionamento del gioco, specificando e spiegando tutte le funzionalità degli script. Infine, i risultati evidenziano che il 95% dei progetti è stato dotato di istruzioni chiare.

3.1. *Concetti computazionali acquisiti*

I 40 progetti implementati sono stati analizzati per rilevare i concetti computazionali acquisiti secondo la metodologia di Wilson et al. (2013) (Tabella 8). In particolare, per quanto riguarda i concetti di programmazione, nel 100% dei progetti sono state utilizzate le Sequenze, nel 78% i Cicli e nell'88% l'Interazione definita dall'utente (per esempio, cambio di uno sfondo determinato da un input della tastiera, quindi abbinato alla pressone di uno specifico tasto). I progetti sviluppati dai diversi team presentavano diversi livelli di complessità a livello di programmazione. I comandi di comunicazione e sincronizzazione sono stati utilizzati in modo prevalente (83% dei casi). Questi comandi hanno un ruolo fondamentale quando gli utenti costruiscono progetti più strutturati e complessi in Scratch poiché consentono di passare da una funzionalità ad un'altra. Inoltre, le operazioni booleane (13%), le Variabili (20%) e i Numeri casuali (8%) sono risultati essere dei concetti particolarmente difficili da apprendere. Il 63% dei progetti presentava dichiarazioni condizionali, per esempio "Se", oppure "Se - Altrimenti" (menu "Controllo"). Questi blocchi vengono utilizzati, ad esempio, per programmare uno sprite che deve spostarsi lungo un perimetro evitando gli oggetti oppure per farlo tornare indietro se tocca il bordo.

VALUTAZIONE DEI PROGETTI SCRATCH	
Concetti di programmazione	%
Sequenza	100
Interazione dell'utente (es. Input della tastiera)	88
Iterazione (cicli)	78
Variabili	20
Istruzioni condizionali	63
Elenco (array)	0
Coordinamento e sincronizzazione (parallelismo)	83
Numeri casuali	8
Logica booleana	13
Organizzazione del codice	
Blocchi estranei	5
Nomi Sprite (il valore predefinito è sovrascritto)	25
Nomi di variabili	20
Usabilità	
Funzionalità	97,5
Personalizzazione Sprite	72,5
Personalizzazione dello stage	82,5
Istruzioni chiare	95
Originalità del progetto	97,5

Tabella 8. Concetti di programmazione appresi dai programmatori principianti (concetti di programmazione, progettazione organizzativa del codice e usabilità).

Tutti i progetti sviluppati sono funzionanti; solo il 5% include blocchi estranei, ossia blocchi di codice non collegati agli altri, la cui presenza non influisce sul funzionamento generale del progetto. Nel 25% dei casi è stato modificato il nome predefinito dello sprite; il 20% includeva variabili e tutti avevano nomi significativi collegati alla funzionalità. Gli sprite (72,5%) e gli stage (82,5%) sono stati personalizzati; quindi, i programmatori non hanno usato sprite o stage predefiniti, ma hanno accuratamente disegnato/realizzato questi elementi.
Nei loro report, i programmatori principianti hanno dichiarato di aver testato più volte il proprio progetto con i bambini, al fine di sviluppare un progetto in Scratch centrato sull'utente e che rispecchiasse le loro reali esigenze. Questa affermazione è supportata dall'analisi dei risultati relativi alla categoria Usabilità. Infine, i risultati mostrano che il 97,5% dei progetti presenta un alto livello di funzionalità che sono state realizzate tenendo conto dell'obiettivo educativo definito dallo sviluppatore. Questo criterio è indice dell'originalità di un progetto.

3.2. *Livello di competenza*

Il livello di competenza è stato misurato analizzando i 40 progetti Scratch attraverso l'utilizzo del software Dr. Scratch (Moreno-León et al., 2015). Dr. Scratch assegna un punteggio a ciascuno dei sette concetti presenti nel progetto e ciascun programmatore può ottenere da 0 a 21 punti. Come mostrato nella Tabella 9, i concetti più utilizzati sono i blocchi di Controllo del flusso (61,67%) e l'interattività dell'utente (60,83%). Ciò implica che i progetti hanno sequenze di istruzioni, prevedono l'uso di condizioni specifiche e l'interazione degli utenti. Il parallelismo è stato utilizzato nel 48,33% dei progetti, a significare che diversi sprite (personaggi) sono stati attivati contemporaneamente. Gli altri concetti usati sono: la sincronizzazione (45,83%); i concetti logici (38,33%) (AND/OR/NOT), compresi i comandi condizionali (Se, Se - altrimenti); la rappresentazione dei dati (37,50%), e l'astrazione (33,33%).

CONCETTI COMPUTAZIONALI	%
Controllo del flusso	61,67
Rappresentazione dei dati	37,50
Astrazione	33,33
Interattività dell'utente	60,83
Sincronizzazione	45,83
Parallelismo	48,33
Logica	38,33

Tabella 9. Livello di competenza dei progetti.

Il software Dr. Scratch tiene conto anche di alcune "cattive abitudini" o di possibili errori, come nomi di sprite non significativi, ripetizione di codice, codice mai eseguito o inizializzazione errata degli attributi dell'oggetto. Il livello raggiunto da un programmatore (Base, Intermedio, Avanzato) rispecchia i concetti utilizzati dall'utente nel progetto (Tabella 5). Quindi, gli sviluppatori possono provare a migliorare i propri progetti usando le linee guida e i suggerimenti offerti dallo strumento Dr. Scratch. Dall'analisi eseguita è emerso che il 30% dei progetti sviluppati rientra nel livello "Base" di competenza, il 60% in quello "Intermedio" e il 10% nel livello "Avanzato".

3.3. *Livello di auto-efficacia*

Alla fine del percorso laboratoriale, l'82% degli studenti percepiva se stesso come "esperto" nell'uso di Scratch. Solo un 18% degli studenti si riteneva "principiante". I partecipanti hanno giudicato la loro esperienza con Scratch eccellente o buona (91,67%), in grado di stimolare la creatività (83,33%) e di migliorare le loro competenze lavorative (79,17%). Gli insegnanti si sono dichiarati soddisfatti o molto soddisfatti delle linee guida fornite per sviluppare i propri progetti (81,82%). Ogni team di lavoro ha ritenuto l'esperienza utile per la futura carriera. In particolare, l'85% ha giudicato "Molto buone" le Conoscenze pedagogiche acquisite, così come la capacità di progettare materiale educativo relativamente al coding. Il 97,5% dei partecipanti ha confermato che l'interazione avuta con i bambini durante il test del progetto ha avuto effetti altamente positivi sulla loro motivazione nell'acquisire conoscenze di coding.

In generale, i futuri insegnanti si sono dichiarati molto soddisfatti dei risultati raggiunti e di aver completato il progetto. Dalle risposte aperte è emerso quanto segue:

"Ci siamo resi conto che è molto importante sperimentare l'uso di nuovi strumenti didattici, abbiamo arricchito il nostro bagaglio culturale e capito che lo scetticismo iniziale mostrato verso questo tipo di attività era totalmente infondato".

"Dobbiamo essere più flessibili ed elastici, non limitare l'immaginazione dei nostri studenti e fornire loro più strumenti utili per il loro processo di apprendimento".

"Ci è stata data l'opportunità di interagire con i colleghi. Questa opportunità ha stimolato e motivato a riflettere sui vantaggi e sugli svantaggi del lavoro di gruppo. Ogni riunione produceva uniformità di pensiero o divergenza di opinioni; tuttavia, è stata un'esperienza innovativa".

"Scratch offre l'opportunità di creare progetti in modo che adulti e bambini non siano solo utenti ma anche produttori digitali".

"Programmare con Scratch è stata un'esperienza interessante e divertente ... Questa applicazione è adatta e utile per imparare le nozioni di programmazione e realizzare prodotti coinvolgenti per bambini di diverse fasce di età".

3.4. *Valutazione finale del progetto*

Per superare il corso è stato assegnato ai futuri insegnanti il compito di progettare e sviluppare un progetto in Scratch con un obiettivo educativo seguendo specifiche linee guide e di elaborare un report. Alla fine del laboratorio, i voti assegnati in base ai criteri definiti in Tabella 7 rispecchiano il livello di conoscenze e competenze acquisite. In particolare, è emerso che la performance generale degli studenti insegnanti è stata di alta qualità. Infatti, il 21% ha ricevuto un punteggio di "Eccellente", il 43% "Molto buono", il 22% "Buono"; il 6% "Discreto" e solo l'8% ha ricevuto un voto "Sufficiente" all'esame finale.

4. DISCUSSIONE

Curzon, Dorling, Ng, Selby e Woollard (2014) hanno sottolineato l'importanza di valutare la crescente competenza degli studenti nelle attività di coding e di mappare i risultati dell'apprendimento. Pertanto, abbiamo selezionato in letteratura due metodologie di valutazione: quella di Wilson et al. (2013) e quella di Moreno-León et al. (2015). Mentre la prima metodologia permette di individuare i concetti di programmazione acquisiti dagli insegnanti in formazione iniziale attraverso la progettazione e l'implementazione delle loro applicazioni, la seconda consente di valutare con precisione il livello di competenza raggiunto nel padroneggiare i concetti. Il campione composto dai 141 futuri insegnanti è stato in grado di implementare 40 progetti, acquisendo in breve tempo concetti di programmazione sia di base che avanzati.

In riferimento alla prima domanda di ricerca, "Quali sono i concetti computazionali acquisiti dagli insegnanti in formazione iniziale?", attraverso la metodologia di Wilson et al. (2013) è emerso che i partecipanti hanno scoperto e utilizzato sequenza, interazione, cicli, condizionali e infine comunicazione e sincronizzazione in diversi script. Il fatto che certi blocchi siano stati usati correttamente in un progetto significa che un concetto di programmazione è stato compreso e appreso. Inoltre, nei progetti è stato riscontrato un uso ridotto di variabili, operatori logici, mentre non sono stati mai utilizzate liste, valore assoluto e radice quadrata. Come discusso da Sajaniemi (2005), ciò è probabilmente dovuto al fatto che questi sono concetti di coding avanzati che richiedono alta capacità di astrazione. Infatti, piuttosto che usare dei cicli o delle variabili, i partecipanti hanno ripetuto alcuni script. In alcuni casi hanno trovato molto utile adottare l'approccio remix per implementare il proprio progetto, scaricando dalla comunità Scratch online altri progetti, modificando gli script e integrandoli nel proprio. Tuttavia, l'opportunità di trovare altri progetti e verificare se alcuni script possono essere riutilizzati richiede una grande abilità. In effetti, gli studenti hanno utilizzato inconsciamente l'approccio dal basso verso l'alto (bottom-up) nelle attività di coding, scomponendo l'intero progetto in piccoli script e analizzandone le funzionalità.

I risultati mostrano che i progetti di Scratch sono stati implementati rispettando il concetto di "usabilità", ossia gli studenti hanno personalizzato gli sprite in base all'obiettivo formativo e alle esigenze degli utenti; hanno personalizzato suoni, immagini, file audio e così via. Solo il 29% dei progetti utilizzava immagini di sfondi predefiniti, mentre gli altri team hanno creato i propri materiali digitali. Infine, ogni gruppo di lavoro ha testato il proprio progetto con gli utenti finali. Questo metodo consente ai team di sviluppare un progetto centrato sull'utente con un alto livello di usabilità. Abbiamo spiegato agli studenti l'importanza di coinvolgere gli utenti finali nel processo di sviluppo del progetto e di tenere conto delle esigenze degli utenti e verificare se il loro progetto fosse efficace, efficiente, coinvolgente, tollerante agli errori, facile da imparare. I progetti implementati hanno permesso di gestire facilmente la modularità, grazie alla facilità d'uso di Scratch. Tenendo conto di quanto affermato da Wing (2006), la modularità consente ai programmatori di scomporre l'intero programma in parti più piccole di codice (script); quindi, i novizi programmatori possono facilmente gestire le interdipendenze tra le parti del progetto e assemblare applicazioni molto complesse in modo affidabile. In effetti, gli studenti creavano, condividevano e riutilizzavano oggetti e componenti; hanno decomposto il problema principale in componenti o moduli più piccoli e sviluppato gli script per ciascuno sprite in modo indipendente e in modo asincrono prima di assemblarli, in linea con i risultati di Brennan e Resnick (2012).

Per quanto riguarda la seconda domanda di ricerca, "Qual è il livello di competenza acquisito?", il quale è stato misurato attraverso l'utilizzo di Dr. Scratch, il punteggio totale di ciascun progetto riflette il livello di competenza cognitiva raggiunto dai programmatori. Di conseguenza, i risultati mostrano che gli insegnanti in formazione hanno raggiunto un livello medio-alto di competenze: il 60% dei progetti è stato classificato "Intermedio" e il 10% "Avanzato". Solo il 30% dei progetti è risultato di base (gli studenti hanno utilizzato solo sequenze di blocchi, il comando "Attendi" per la sincronizzazione e solo la "Bandierina verde" per implementare l'interattività). Per quanto riguarda il livello medio-alto, il 60% dei progetti etichettati come "Intermedi" presenta una buona definizione di blocchi e condizionali come "se altrimenti" e cicli come "ripeti" e "ripeti per sempre". L'interattività è stata applicata utilizzando i tasti "premuto" o gli sprite cliccati. Infine, solo pochi hanno utilizzato variabili. Alcuni team sono stati in grado di gestire script complessi utilizzando, ad esempio, funzioni di interattività più complesse rispetto all'uso della "Bandierina verde". Infine, il 10% dei progetti che hanno ottenuto un livello di competenza avanzato ha utilizzato tutti i concetti di programmazione spiegati durante le lezioni, ad eccezione delle liste. I progetti incorporavano registrazioni audio, operazioni logiche e codice per diversi sprite.

Riassumendo, le metodologie utilizzate per valutare i progetti degli studenti insegnanti sono risultate facili

da applicare e utili per valutare le conoscenze di CT acquisite. Questi criteri possono essere riutilizzati anche dagli stessi insegnanti per i prodotti dell'apprendimento, ossia i progetti sviluppati. Pertanto, questi strumenti diventano utili principalmente durante il processo di apprendimento per fornire una fotografia dei concetti computazionali presenti in un progetto Scratch ordinandoli per livello di complessità. I futuri insegnanti potranno garantire il criterio di obiettività per valutare la qualità dei progetti sviluppati, riducendo sforzo e soggettività a vantaggio di imparzialità (Brennan, 2013), facendo riferimento alle indicazioni fornite da Román-González, Moreno-León e Robles (2019) che propongono un modello completo per valutare il CT partendo dai livelli cognitivi contemplati nella tassonomia di Bloom. Il modello indica esplicitamente come combinare in modo armonioso i diversi tipi di strumenti di valutazione del CT al fine di dare risposta alle domande di ricerca più comuni nel campo dell'istruzione e ad eseguire accurate valutazioni in base agli obiettivi di ricerca.

Le opinioni riportate dai futuri insegnanti dimostrano come la metodologia didattica adottata sia servita a motivare i partecipanti e ha permesso di ottenere un ottimo risultato generale.

5. CONCLUSIONI E PROSPETTIVE FUTURE

Secondo McComas (2013), un'esigenza imperativa della nostra società è quella di preparare gli insegnanti a fornire competenze e contenuti efficaci, e ciò richiede un'esperienza di curriculum adeguata e un ragionevole periodo di pratica. A nostro parere, questo è anche particolarmente importante per gli insegnanti in formazione iniziale, per prepararli ad insegnare le abilità del 21° secolo attraverso un'adeguata metodologia.

I nostri risultati suggeriscono che le attività di laboratorio hanno sortito diversi effetti positivi, tra cui un pieno coinvolgimento dei programmatori principianti nelle attività di coding. Ciò è probabilmente dovuto alla facilità d'uso che caratterizza il software Scratch e alla possibilità di verificare in tempo reale l'output degli script sullo schermo. Quest'ultima funzione è particolarmente utile per tutti i tipi di progetti didattici (Koh, 2013).

Per quanto riguarda l'importanza pratica di questa indagine, in poche settimane gli studenti sono stati in grado di acquisire gli elementi chiave del coding utili per sviluppare progetti con Scratch. Questo gruppo di futuri insegnanti è stato in grado di acquisire concetti di programmazione, sperimentando il problem solving: hanno utilizzato nuovi strumenti tecnologici, sono stati in grado di applicare diversi processi cognitivi, creatività, pensiero critico e processo decisionale, condividendo allo stesso tempo le conoscenze e responsabilità nel gruppo.

La metodologia didattica presenta diversi vantaggi. Innanzitutto, rappresenta un utile e ripetibile esempio che i futuri insegnanti potrebbero applicare nella pratica scolastica. In effetti, abbiamo mostrato loro come integrare le attività di coding nella didattica, fornendo loro precise raccomandazioni metodologiche e didattiche. In secondo luogo, l'approccio utilizzato è "friendly", in quanto, grazie all'apprendimento collaborativo e all'approccio cooperativo adottato, in prospettiva costruttivista, e promuovendo quindi l'acquisizione di competenze attraverso il learning by doing, in breve tempo i programmatori principianti hanno raggiunto un livello di competenza medio-alto nelle attività di coding. Infine, l'approccio ha puntato molto sulla motivazione degli studenti per affrontare le sfide e il cambiamento generale. Porter, Lee, Simon e Guzdial (2017) sottolineano la necessità e l'importanza di enfatizzare le "best practice". Il nostro metodo parte dal basso perché tiene conto delle reali esigenze degli studenti che devono imparare a programmare. Questi ultimi non hanno un background di base in informatica e in alcuni casi sono studenti adulti che devono essere fortemente motivati. A nostro avviso, i risultati presentati in questo documento consentono di raccogliere informazioni importanti da un punto di vista qualitativo. Infine, secondo noi, sia i risultati positivi che negativi del laboratorio potrebbero fornire utili indicazioni per ripensare ad alcuni curricula degli insegnanti

in formazione italiani e centrarli sulle esigenze pratiche della futura professione.
Tuttavia, è bene qui evidenziare come l'aspetto del lavoro di gruppo, che ha avuto un ruolo determinante a livello motivazionale, in realtà costituisca anche un limite. Gli studenti hanno sviluppato il proprio progetto in gruppo, quindi i dati raccolti rispecchiavano il lavoro di più soggetti. Se gli studenti lavorassero individualmente, i dati potrebbero riflettere in modo più accurato la performance di ogni studente. Quindi, nel proseguo del lavoro abbiamo pensato che sarebbe interessante utilizzare le linee guida implementate per sviluppare dei percorsi individuali, coinvolgere più gruppi di ricerca e indagare la componente motivazionale con un pre e un post test. Una siffatta impostazione permetterebbe di raccogliere anche dati di tipo quantitativo e di effettuare, ad esempio, delle correlazioni statistiche tra le diverse variabili in gioco.

6. RINGRAZIAMENTI

Gli autori hanno partecipato egualmente alle attività di a. formulazione dell'ipotesi sottostante la ricerca; b. progettazione della ricerca e della metodologia; c. raccolta dei dati; d. elaborazione e analisi dei dati; e. interpretazione dei risultati; f. redazione di parti significative del testo e revisione del testo per la sottomissione finale.

7. BIBLIOGRAFIA

An, S. & Lee, Y. (2014). Development of Pre-service Teacher Education Program for Computational Thinking. In M. Searson & M. Ochoa (Eds.), *Proceedings of SITE 2014--Society for Information Technology & Teacher Education International Conference* (pp. 2055-2059). Jacksonville, FL, USA: Association for the Advancement of Computing in Education (AACE).

Armoni, M., Meerbaum-Salant, O., & Ben-Ari, M. (2015). From scratch to "real" programming. *ACM Transactions on Computing Education (TOCE)*, *14*(4), 25. doi:10.1145/2677087

Barr, D., Harrison, J., & Conery, L. (2011). Computational Thinking: A Digital Age Skill for Everyone. *Learning & Leading with Technology*, *38*(6), 20-23.

Baytak, A., & Land, S. M. (2011). An investigation of the artifacts and process of constructing computers games about environmental science in a fifth grade classroom. *Educational Technology Research and Development*, *59*(6), 765-782. doi:10.1007/s11423-010-9184-z

Bell, S., Frey, T., & Vasserman, E. (2014). Spreading the word: introducing pre-service teachers to programming in the K12 classroom. In *Proceedings of the 45*th *ACM Technical Symposium on Computer Science Education* (pp. 187-192). New York, NY, USA: ACM. doi:10.1145/2538862.2538963

Bertacchini, F., Bilotta, E., Carini, M., Gabriele, L., Pantano, P., & Tavernise, A. (2012). Learning in the smart city: A virtual and augmented museum devoted to chaos theory. In *International Conference on Web-Based Learning* (pp. 261-270). Heidelberg, DE: Springer. doi:10.1007/978-3-662-43454-3_27

Bertacchini, F., Bilotta, E., & Pantano, P. (2017). Shopping with a robotic companion. *Computers in Human Behavior*, *77*, 382-395. doi:10.1016/j.chb.2017.02.064

Bilotta, E., Pantano, P., Bertacchini, F., Gabriele, L., Longo, G., Mazzeo, V., … Vena, S. (2007). ImaginationTOOLS (TM). A 3D Environment for Learning and Playing Music. In *Eurographics Italian Chapter Conference 1*, 139-144. Trento, IT.

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., & Engelhardt, K. (2016). *Developing computational thinking in compulsory education – Implications for policy and practice*, JRC Working Papers JRC104188, Joint Research Centre (Seville site).

Brennan, K. (2013). Learning computing through creating and connecting. *Computer*, *46*(9), 52-59. doi:10.1109/MC.2013.229

Brennan, K. A., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American Educational Research Association* (Vol. 1, p. 25). Vancouver, CA.
Retrieved from http://scratched.gse.harvard.edu/ct/files/AERA2012.pdf

Chiu, C. F. (2014, March). Use of problem-solving approach to teach scratch programming for adult novice programmers. In *Proceedings of the 45th ACM technical symposium on Computer science education* (pp. 710-711). New York, NY, USA: ACM. doi:10.1145/2538862.2544284

Curzon, P., Dorling, M., Ng, T., Selby, C., & Woollard, J. (2014). *Developing computational thinking in the classroom: a framework*. Swindon, UK: Computing at School.

Denner, J., Werner, L., & Ortiz, E. (2012). Computer games created by middle school girls: Can they be used to measure understanding of computer science concepts? *Computers & Education*, *58*(1), 240-249. doi:10.1016/j.compedu.2011.08.006

Faraco, G., & Gabriele, L. (2007). Using LabVIEW for applying mathematical models in representing phenomena. *Computers & Education*, *49*(3), 856-872. doi:10.1016/j.compedu.2005.11.025

Federici, S., Gola, E., Brau, D., & Zuncheddu, A. (2015). Are Educators Ready for Coding?. In *Proceedings of the 7th International Conference on Computer Supported Education*-Volume 1 (pp. 494-500). Lisbon, Portugal: SCITEPRESS-Science and Technology Publications, Lda. doi:10.5220/0005491604940500

Foerster, K. T. (2016). Integrating programming into the mathematics curriculum: Combining scratch and geometry in grades 6 and 7. In *Proceedings of the 17th annual conference on information technology education* (pp. 91-96). New York, NY, USA: ACM. doi:10.1145/2978192.2978222

Gabriele, L., Marocco, D., Bertacchini, F., Pantano, P., & Bilotta, E. (2017). An educational robotics lab to investigate cognitive strategies and to foster learning in an arts and humanities course degree. *International Journal of Online Engineering (iJOE)*, *13*(04), 7-19. doi:10.3991/ijoe.v13i04.6962

Giglio, S., Gabriele, L., Tavernise, A., Pantano, P., Bilotta, E., & Bertacchini, F. (2015). Virtual museums and Calabrian Cultural Heritage: projects and challenges. In J.C. Torres et al. (Eds). *2015 Digital Heritage* (Vol. 2, pp. 703-708), Granada, IEEE. doi:10.1109/DigitalHeritage.2015.7419603

Kobsiripat, W. (2015). Effects of the Media to Promote the Scratch Programming Capabilities Creativity of Elementary School Students. *Procedia-Social and Behavioral Sciences*, *174*, 227-232. doi:10.1016/j.sbspro.2015.01.651

Koh, K. (2013). Adolescents' information-creating behavior embedded in digital Media practice using scratch. *Journal of the American Society for Information Science and Technology*, *64*(9), 1826-1841. doi:10.1002/asi.22878

Korkmaz, Ö. (2016). The Effects of Scratch-Based Game Activities on Students' Attitudes, Self-Efficacy and Academic Achievement. *International Journal of Modern Education and Computer Science*, *1*, 16-23. doi:10.5815/ijmecs.2016.01.03

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*, *41*, 51-61. doi:10.1016/j.chb.2014.09.012

Maloney, J. H., Peppler, K., Kafai, Y., Resnick, M., & Rusk, N. (2008). Programming by choice: urban youth learning programming with scratch. *ACM SIGCSE Bulletin*, *40*(1), 367-371. doi:10.1145/1352322.1352260

Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The scratch programming language and environment. *ACM Transactions on Computing Education* (TOCE), *10*(4), 16. doi:10.1145/1868358.1868363

McComas, W. F. (Ed.). (2013). *The Language of Science Education: An Expanded Glossary of Key Terms and Concepts in Science Teaching and Learning.* Rotterdam, NL: Springer Science & Business Media.

Monroy-Hernández, A. (2012). *Designing for remixing: Supporting an online community of amateur creators* (Doctoral dissertation, Massachusetts Institute of Technology). Retrieved from https://llk.media.mit.edu/papers/andres-dissertation.pdf

Moreno, J., & Robles, G. (2016). Code to learn with Scratch. A systematic literature review. In *2016 IEEE Global Engineering Education Conference, EDUCON* (pp. 150-156). Abu Dhabi, UAE: IEEE. doi:10.1109/EDUCON.2016.7474546

Moreno-León, J., Robles, G., & Román-González, M. (2015). Dr. Scratch: Análisis Automático de Proyectos Scratch para Evaluar y Fomentar el Pensamiento Computacional. *Revista de Educación a Distancia*, *46*(10), 1-23. doi:10.6018/red/46/10

Porter, L., Lee, C., Simon, B., & Guzdial, M. (2017). Preparing tomorrow's faculty to address challenges in teaching computer science. *Communications of the ACM*, *60*(5), 25-27. doi:10.1145/3068791

Papert, S. (1984). *Mindstorms, Bambini computers e creatività.* Milano, IT: Emme Edizioni.

Papert, S. (1986). *Constructionism: A new opportunity for elementary science education*. Cambridge, MA, USA: MIT, Media Laboratory, Epistemology and Learning Group.

Piaget, J. (1967). *Lo sviluppo mentale del bambino*. Torino, IT: Einaudi.

Repenning, A., Webb, D. C., Koh, K. H., Nickerson, H., Miller, S. B., Brand, C., …Repenning, N. (2015). Scalable game design: A strategy to bring systemic computer science education to schools through game design and simulation creation. *ACM Transactions on Computing Education (TOCE)*, *15*(2), 1-31. doi:10.1145/2700517

Quan, C. G. (2015). Student teachers evaluating and assessing SCRATCH in the Applied Linguistics classroom. *Procedia-Social and Behavioral Sciences*, *174*, 1450-1456. doi:10.1016/j.sbspro.2015.01.774

Román-González, M., Moreno-León, J., & Robles, G. (2019). Combining Assessment Tools for a Comprehensive Evaluation of Computational Thinking Interventions. In S. C. Kong, & H. Abelson (Eds.), *Computational Thinking Education* (pp. 79–98). Singapore, SG: Springer. doi:10.1007/978-981-13-6528-7_6

Sajaniemi, J. (2005). Roles of variables and learning to program. In A. Jimoyiannis (Ed.), *Proceedings of the 3rd Panhellenic Conference on Didactics of Informatics*. University of Peloponnese, Korinthos, Greece.

Seiter, L., & Foreman, B. (2013). Modeling the learning progressions of computational thinking of primary grade students. In *Proceedings of the ninth annual international ACM Conference on International computing education research* (pp. 59-66). New York, NY, USA: ACM. doi:10.1145/2493394.2493403

Tan, L., & Kim, B. (2015). Learning by Doing in the Digital Media Age. In Lin, T. B., Chen, V., & Chai, C. (Eds.), *New Media and Learning in the 21st Century* (pp. 181-197). Singapore, SG: Springer. doi:10.1007/978-981-287-326-2_12

Vaca-Cárdenas, L. A., Bertacchini, F., Tavernise, A., Gabriele, L., Valenti, A., Olmedo, …Bilotta, E. (2015). Coding with Scratch: The design of an educational setting for Elementary pre-service teachers. In 2015 *International Conference on Interactive Collaborative Learning* (ICL) (pp. 1171-1177). Florence, IT: IEEE. doi:10.1109/ICL.2015.7318200

Vaca-Cárdenas, L., Tavernise, A., Bertacchini, F., Gabriele, L., Valenti, A., Pantano, P., & Bilotta, E. (2016). An Educational Coding Laboratory for Elementary Pre-service Teachers: A Qualitative Approach. *International Journal of Engineering Pedagogy (iJEP)*, *6*(1), 11-17. doi:10.3991/ijep.v6i1.5146

Wilson, A., Hainey, T., & Connolly, T. M. (2013). Using Scratch with primary school children: an evaluation of games constructed to gauge understanding of programming concepts. *International Journal of Game-Based Learning (IJGBL)*, *3*(1), 93-109. doi:10.4018/ijgbl.2013010107

Wilson, A., & Moffat, D. C. (2010). Evaluating Scratch to introduce younger school children to programming. In *Proceedings of the Psychology of Programming Interest Group Workshop*, (p. 64-75). Madrid, SP. Retrieved from http://scratched.gse.harvard.edu/sites/default/files/wilson-moffat-ppig2010-final.pdf

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33-35. doi:10.1145/1118178.1118215

Wing, J. M. (2011). Research notebook: Computational thinking. What and why. *The Link.*, 20-23. Retrieved from https://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why

Wing, J. M. (2017). Computational thinking's influence on research and education for all. *Italian Journal of Educational Technology*, *25*(2), 7-14. doi:10.17471/2499-4324/922

Yin, R. K. (1994). *Case study research: Design and methods (2nd ed.).* Newbury Park, CA, USA: Sage.

8. APPENDICE

Cognome_____Nome_____età_____

1) Indicare la/e scuola/e superiore/i frequentata/e

_____

1) Sai usare il computer? SI NO
2) Valuta la tua "capacità" di usare il computer
   1=insufficiente; 2= appena sufficiente; 3= sufficiente; 4=buono; 5=ottimo
3) Quali tra questi programmi conosci?
   Word - Write - Pages (creare/modificare testi)
   Power Point - Impress - Keynote (creare/modificare presentazioni)
   Excel - Calc - Numbers (creare/modificare fogli elettronici)
   Access - Base (creare/modificare i database)
4) Utilizzi Internet per
   Navigare in rete
   Lavoro
   Usare i social network
   Altro
5) Hai partecipato ad un corso sulle TIC? SI NO
6) Conosci un linguaggio di programmazione? SI NO
   Se sì, specificare_____
7) Valuta il tuo livello di conoscenza del linguaggio di programmazione
   1=insufficiente; 2= appena sufficiente; 3= sufficiente; 4=buono; 5=ottimo
8) Indica il tuo livello di alfabetizzazione digitale generale
   1=insufficiente; 2= appena sufficiente; 3= sufficiente; 4=buono; 5=ottimo